On Approximability of Bounded Degree Instances of Selected Optimization Problems

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von
Richard Schmied
aus
Mährisch Ostrau

Bonn, 2013

Angefertigt mit Genehmigung
der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter: Prof. Dr. Marek Karpinski
2. Gutachter: Prof. Dr. Andrzej Lingas

Tag der Promotion: 25.07.2013
Erscheinungsjahr: 2013

Abstract

In order to cope with the approximation hardness of an underlying optimization problem, it is advantageous to consider specific families of instances with properties that can be exploited to obtain efficient approximation algorithms for the restricted version of the problem with improved performance guarantees. In this thesis, we investigate the approximation complexity of selected **NP**-hard optimization problems restricted to instances with bounded degree, occurrence or weight parameter. Specifically, we consider the family of *dense* instances, where typically the average degree is bounded from below by some function of the size of the instance. Complementarily, we examine the family of *sparse* instances, in which the average degree is bounded from above by some fixed constant. We focus on developing new methods for proving explicit approximation hardness results for general as well as for restricted instances.

The fist part of the thesis contributes to the systematic investigation of the VERTEX COVER problem in k-hypergraphs and k-partite k-hypergraphs with density and regularity constraints. We design efficient approximation algorithms for the problems with improved performance guarantees as compared to the general case. On the other hand, we prove the optimality of our approximation upper bounds under the Unique Games Conjecture or a variant.

In the second part of the thesis, we study mainly the approximation hardness of restricted instances of selected global optimization problems. We establish improved or in some cases the first inapproximability thresholds for the problems considered in this thesis such as the METRIC DIMENSION problem restricted to graphs with maximum degree 3 and the (1,2)-STEINER TREE problem. We introduce a new reductions method for proving explicit approximation lower bounds for problems that are related to the TRAVELING SALESPERSON (TSP) problem. In particular, we prove the best up to now inapproximability thresholds for the general METRIC TSP problem, the ASYMMETRIC TSP problem, the SHORTEST SUPERSTRING problem, the MAXIMUM TSP problem and TSP problems with bounded metrics.

Acknowledgement

I am deeply indebted to my advisor Marek Karpinski, in particular, for his invaluable guidance on my research, his everlasting willingness to spend time discussing new ideas with me, and also for creating the prefect working conditions at the Computer Science Department of the University of Bonn. His dedication to research and scientific curiosity were a great source of inspiration for me.

In addition, I would like to express my gratitude to the coauthors with whom I had the great pleasure to collaborate with: Jean Cardinal, Mathias Hauptmann, Marek Karpinski, Michael Lampis and Claus Viehmann. I enjoyed working with them very much and I have learned a lot from them.

I am also very grateful to Marek Karpinski, Andrzej Lingas, Heiko Röglin and Jens Franke to agree to be in my Thesis Committee.

I thank the whole team of the Algorithms and Distributed Systems group at the University of Bonn. Special thanks to my colleagues and friends who made my time in Bonn joyful and productive: Manuel Arora, Mathias Hauptmann, Matthias Kretschmer, Johannes Mittmann and Claus Viehmann.

Many thanks to the helpful and competent administrative staff at the Computer Science Department and BIGS, especially Christiane Andrade and Karen Bingel.

I gratefully acknowledge the Hausdorff Center for Mathematics for their financial support.

I am forever indebted to my father for his everlasting encouragement and support. This thesis is dedicated to my fiancee without whose love and support I would not be where I am today.

Contents

CHAPTER 1

Introduction

In order to study the hardness of computational problems, the notion of **NP**-hardness was introduced independently by Cook [C71] and Levin [L73]. Nowadays, it is widely believed that it is not possible to solve efficiently a **NP**-hard problem. As demonstrated in the early 1970's, a large class of natural combinatorial problems arising in practice, which had by then withstood all efforts of computer scientists all over the world to design efficient algorithms for those problems, were proved to be **NP**-hard.

Fortunately, it is sufficient for many applications to find an approximate solution that is very close to the optimum. Algorithms that construct a feasible solution for a given problem are called approximation algorithms and their quality is measured by the approximation ratio, which is the worst possible relative discrepancy between the value of an optimum solution and the solution produced by the algorithm.

The discovery of the PCP-Theorem [AS98, ALM$^+$98] and its consequences lead to the fact that there are problems, for which it is as hard to compute an approximate solution to within a certain precision as solving it exactly. Moreover, the existence of a sharp inapproximability threshold was proved for some problems, that is, there is an approximation ratio, for which it is feasible to approximate the problem, but on the other hand, computing an solution with an even slightly smaller factor becomes **NP**-hard. Nevertheless, for many problems, there still exists a gap between the approximation ratio of the best known efficient algorithm and the best inapproximability threshold.

It reflects the research in this area, in which we have given an optimization problem and we do not only want to know the approximation ratio of the best known efficient algorithm, but we also want to know what is the best approximation that can be achieved efficiently. Is it worthwhile to seek for algorithms with improved approximation ratios or have we already found the best possible approximation? Moreover, it is known that optimization problems have different behavior with respect to inapproximability. In particular, there are problems, which are **NP**-hard to approximate to within every constant factor. For other problems, it is possible to find a solution with some constant approximation ratio. On the other hand, there are **NP**-hard problems, for which it is possible to compute efficiently a solution within arbitrary precision.

In order to cope with the approximation hardness of an underlying optimization problem, it is advantageous to consider specific families of instances with properties that can be exploited to design efficient algorithms with improved approximation ratios as compared to general instances of the problem.

In this thesis, we investigate the approximation complexity of selected **NP**-hard optimization problems restricted to instances with bounded degree, occurrence or weight parameter. Specifically, we consider the family of *dense* instances, where typically the average degree is bounded from below by some function of the size of the instance. Complementarily, we examine the family of *sparse* instances, in which the average degree is bounded from above by some fixed constant. Moreover, we focus on developing new methods for proving explicit approximation hardness results for general as well as for restricted instances.

Dense Instances

Density is a property of the instance of the underlying optimization problem. A graph, for example, is called *dense*, whenever its average degree is linear in the number of vertices. From the probabilistic point of view, almost all graphs are dense and therefore, dense graphs obviously constitute an important family of instances. In 1995, Arora, Karger and Karpinski [AKK95] proved that a wide range of **NP**-hard optimization problems restricted to dense instances can be approximated within arbitrary precision. Later, Karpinski and Zelikovsky [KZ97a] defined and studied dense cases of covering problems. In particular, they investigated the approximation complexity of dense instances of the SET COVER problem, the VERTEX COVER problem and the STEINER TREE problem.

In the following years, researchers exhibited other dense covering problems that admit approximation algorithms with improved approximation ratios as compared to general instances. A prominent member of this category of covering problems is the EDGE DOMINATING SET problem. Cardinal et al. [CLL+05] studied the EDGE DOMINATING SET problem restricted to dense graphs and gave an efficient algorithm for the problem with approximation ratio parametrized by the density of the input graph. If the underlying graph is dense enough, the algorithm due to Cardinal et al. [CLL+05] outperforms the best known approximation algorithm for the general problem (cf. [GLR08]). The approximation upper bound for the problem was improved by Cardinal, Langerman and Levy [CLL09] and later, by Schmied and Viehmann [SV11]. Another important optimization problem with this property is the VERTEX COVER problem restricted to dense k-hypergraphs, which was studied by Bar-Yehuda and Zehavit [BK04]. They extended the greedy approach due to Karpinski and Zelikovsky [KZ97a] to k-hypergraphs.

The study of subdense instances of the VERTEX COVER problem in graphs was initiated by Imamura and Imava [II05]. After that, Cardinal, Karpinski, Schmied and Viehmann [CKSV11] extended the study of subdense instances of covering problems. In particular, they investigated the approximability of subdense instances of the CONNECTED VERTEX COVER problem, the SET COVER problem, the STEINER TREE problem and the VERTEX COVER problem in graphs.

The first part of this work is dedicated to the study of the approximation complexity of the VERTEX COVER problem in dense and subdense k-hypergraphs. Furthermore, we extend our investigations and framework for the special case of the problem when the underlying k-hypergraph is even k-partite.

Sparse Instances

Another interesting family of instances are sparse instances, as many optimization problems restricted to sparse instances are known to admit efficient approximation algorithms with improved approximation ratios as compared to the general case. A graph, for example, is called sparse whenever the size of the edge set is at most linear in the number of vertices. On the other hand, the PCP Theorem [ALM+98, AS98] combined with the approximation preserving reductions due to Papadimitriou and Yanakakis [PY91] implies that bounded degree instances of several optimization problems are **NP**-hard to approximate to within some constant factor. Moreover, Berman and Karpinski [BK99] proved inapproximability thresholds for various bounded degree optimization problems with a very small bound.

Explicit approximation lower bounds for bounded degree instances with a very small bound on the degrees have emerged to be important in proving inapproximability results for global optimization problems, for which direct PCP constructions leading to tight inapproximability results are not known. Prominent members of this category of global problems are the TRAVELING SALESPERSON (TSP) problem (cf. [L12], [PV06]), SHORTEST SUPERSTRING problem (cf. [V05]), the asymmetric and symmetric TSP problem restricted to distances one and two (cf. [EK06]) and the STEINER TREE problem with weights one and two (cf. [H07]).

The second part of this work contributes to the study of the approximation hardness of bounded occurrence, degree and weight optimization problems. Moreover, we focus on new methods for proving explicit approximation lower bounds for restricted and also for general instances of selected problems by constructing reductions from well-suited bounded occurrence CSPs. In particular, we obtain the best up to now explicit approximation lower bounds for the METRIC and ASYMMETRIC TSP problem, the TSP problem with bounded metrics and some other restrictions on the underlying metric space, the SHORTEST SUPERSTRING problem restricted to instances with bounded occurrences of characters, the METRIC DIMENSION problem in graphs with maximum degree 3 and the STEINER TREE problem with weights one and two.

1.1 Outline of the Thesis

The thesis divides into three relatively independent parts. In Part I, we provide some relevant background knowledge and introduce preliminaries from complexity theory.

In particular, in Chapter 2, we fix the notation used in the thesis. In Chapter 3, we define the basic complexity classes. In Chapter 4, we give a brief introduction in the topic of approximation algorithms, optimization problems and lower bound techniques.

In Part II, our work is mainly on understanding the approximability of the VERTEX COVER problem with density and regularity constraints.

In Chapter 5, we study the VERTEX COVER problem in k-hypergraphs. In Chapter 6, we consider the VERTEX COVER problem in k-partite k-hypergraphs with given k-partition.

In Part III, we investigate the approximation hardness of selected bounded occurrence, degree and weight optimization problems. In addition, we develop new approaches for proving explicit approximation lower bounds by using and extending sparse instance methods.

Specifically, we examine the $(1,2)$-STEINER TREE problem in Chapter 7, the METRIC DIMENSION problem restricted to graphs with maximum degree 3 in Chapter 8, the SHORTEST SUPERSTRING problem and related problems in Chapter 9, and TSP problems in Chapter 10.

1.2 Contributions

In **Chapter 5**, we study the approximability of the VERTEX COVER problem in (ε, ℓ)-dense k-hypergraphs and design an efficient approximation algorithm the problem with an improved approximation ratio for all $k > 2$ and $\ell > 0$. On the other hand, we give an optimal inapproximability result the problem. Then, we consider a more general class of k-hypergraphs, which are called nearly regular. We give a randomized approximation algorithm for the VERTEX COVER problem in nearly regular k-hypergraphs with approximation ratio strictly less than k and running time depending on the density of the underlying k-hypergraph. It entails the existence of quasi-polynomial and polynomial time randomized approximation algorithms with approximation ratio less than k for mildly sparse and subdense instances, respectively. Furthermore, we obtain tight approximation lower bounds for the problem. In particular, we prove the best known approximation lower bounds for the

VERTEX COVER problem in regular k-hypergraphs. This chapter is based on the work [CKSV12].

In **Chapter 6**, we investigate the approximability of the VERTEX COVER problem in dense and nearly regular k-partite k-hypergraphs. We first consider the VERTEX COVER problem in dense k-balanced hypergraphs and prove that the problem is efficiently approximable within an approximation ratio better than $k/2$. After that, we develop an improved technique for the extraction of a minimum vertex cover of a dense k-partite k-partite by which we obtain an efficient approximation algorithm for the VERTEX COVER problem in dense k-partite k-hypergraphs with a better approximation ratio. On the approximation hardness side, we propose a conjecture about the UG-hardness of the VERTEX COVER problem in k-partite k-hypergraphs. Assuming this conjecture, we prove an optimal inapproximability result for the dense version of the problem. This part of the chapter is based on the work [KSV11].

By combining the framework developed for the VERTEX COVER problem in nearly regular k-hypergraphs with a new method called randomized bucketing extraction, we prove the existence of quasi-polynomial and polynomial time randomized approximation algorithms with approximation ratio less than $k/2$ for mildly sparse and subdense instances, respectively. On the other hand, we prove the optimality of the approximation ratio achieved by our algorithm based on the conjecture mentioned above.

In **Chapter 8**, we study the approximation complexity of the METRIC DIMENSION problem restricted to graphs with maximum degree 3 and prove that the problem is **APX**-hard. In addition, we establish the inapproximability threshold of 353/352 for the problem. This part of the chapter is based on the work [HSV12].

Afterwards, we construct a reduction implying that it is **NP**-hard to approximate the problem to within any constant factor less than 153/152.

In **Chapter 7**, we give an improved approximation lower bound for the $(1,2)$-STEINER TREE problem and prove that the problem is **NP**-hard to approximate to within any constant factor less than 221/220.

In **Chapter 9**, we introduce a new reduction method for proving explicit approximation lower bounds for optimization problems that are related to the SHORTEST SUPERSTRING problem. Based on this reduction method, we are able to improve the best up to now known approximation lower bounds for the SHORTEST SUPERSTRING problem and the MAXIMUM COMPRESSION problem by an order of magnitude. The inapproximability results holds for strongly restricted instances of the SHORTEST SUPERSTRING problem, in which no character appears more than eight times and all given strings have length at most four. It also implies an improved approximation lower bound for the MAX-ATSP problem. This chapter is based on the work [KS11].

In **Chapter 10**, we investigate the approximation hardness of TSP problems with bounded metrics. By extending our method from Chapter 9, we give improved approximation lower bounds for the $(1,2)$-ATSP problem, the $(1,2)$-TSP problem, MAX-$(0,1)$-ATSP problem, the $(1,4)$-ATSP problem, the $(1,2)$-TSP problem restricted to cubic and subcubic instances, and the GRAPHIC-TSP problem in cubic and subcubic graphs. This part is based on the works [KS12] and [KS13].

By constructing a new bounded degree wheel amplifier and exploiting the special properties of a well-suited bounded occurrence CSP, we prove the best up to now inapproximability thresholds for the general METRIC and ASYMMETRIC TSP problem improving upon results of Lampis [L12], and Papadimitriou and Vempala [PV06]. This part is based on the work [KLS13].

Part I

Foundations

CHAPTER 2

Background

This chapter serves as a foundation of notation for subsequent chapters. We fix the notation that we will use in this work and provide some relevant background knowledge. In particular, we review topics that are related to hypergraphs, symmetric metric spaces, strings, finite probability spaces and asymptotical behaviour of functions.

2.1 Standard Notation

We let $\mathbb{Z} = \{0, 1, -1, 2, -2, \ldots\}$ denote the set of integers, $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$ the set of non-negative integers, $\mathbb{N} = \mathbb{N}_0 \backslash \{0\}$ the set of natural numbers and $\mathbb{R}$ the set of real numbers. For a natural number $n \in \mathbb{N}$, we define $[n] = \{1, 2, \ldots, n\}$ and $[n]_0 = \{0\} \cup [n]$. Furthermore, we use the abbreviation $[0] = \varnothing$. Given a set S, we let $2^S = \{S' \mid S' \subseteq S\}$ denote the power set of S and

$$\binom{S}{k} = \{\, S' \subseteq S \mid |S'| = k \,\}$$

the collection of subsets of S with cardinality exactly k.

A *multiset* is defined as a pair (A, m), where A is a set and $m\colon A \to \mathbb{N}$. The set A is called the underlying set of elements. For each $a \in A$, the multiplicity, that is, the number of occurrences of a is $m(a)$.

For a real number x, we denote by $\lceil x \rceil$ the smallest integer n with $n \geq x$, whereas $\lfloor x \rfloor$ is the largest integer such that $n \leq x$. Whenever we use a real number x in a context requiring an integer, $\lceil x \rceil$ is implied. By $\log x$, we denote the logarithm of a positive real x to the base 2 and $\exp[x] = e^x$.

Strings

Let Σ be a finite alphabet. A string s of length n over Σ is a mapping $s : [n] \to \Sigma$. For notational simplicity, we identify a string $s : [n] \to \Sigma$ with the n-tuple $s_1 s_2 \cdots s_n$ of elements from Σ. The empty string is denoted by Λ_0. For $n \in \mathbb{N}$, we define Σ^n as the set of all strings over Σ having length n, $\Sigma^0 = \{\Lambda_0\}$ and $\Sigma^* = \bigcup_{n \geq 0} \Sigma^n$. If x and y are strings, then, we denote their concatenation simply by xy. The length of a string x is denoted by $|x|$.

2.2 Random Variables and Expectation

A *finite probability space* is a finite set $\Omega \neq \emptyset$ along with a function $\mathsf{Pr} : \Omega \to \mathbb{R}_{>0}$ such that

$$\text{for all } \omega \in \Omega, \mathsf{Pr}(\omega) > 0 \text{ and } \sum_{\omega \in \Omega} \mathsf{Pr}(\omega) = 1.$$

The set Ω is the *sample space* and the function Pr is the *probability distribution*. The elements $\omega \in \Omega$ are called *elementary events*. An event E is a subset of Ω, for which we define the *probability* of E by $\mathsf{Pr}(E) = \sum_{\omega \in E} \mathsf{Pr}(\omega)$.

The *uniform distribution* over the sample space is specified by setting $\mathsf{Pr}(\omega) = 1/|\Omega|$ for every $\omega \in \Omega$.

Given a finite probability space (Ω, Pr), a *random variable* X is a function $X : \Omega \to \mathbb{R}$. The *expectation* of a random variable X, denoted by $\mathbb{E}[X]$, is defined as follows.

$$\mathbb{E}[X] = \sum_{\omega \in \Omega} \mathsf{Pr}(\omega) \cdot X(\omega)$$

Chernoff Bounds

We are going to state the most commonly used version of the Chernoff bound applicable for the tail distribution of a sum of independent 0/1 valued random variables.

Poisson trials are repeated independent trials with two possible outcomes called success and failure. In general, the success probability is allowed to change with each trial. Let us represent n Poisson trials by indicator random variables $X_1, \dots, X_n$ that can take values 1 and 0 representing success and failure, respectively. The following statement is a simplified version of the Chernoff bounds.

Theorem 2.2.1
Let $X_1, X_2, \dots, X_n$ be independent Poisson trials with $\mathsf{Pr}[X_i = 1] = p_i$ for all $i \in [n]$. Let $X = \sum_{i \in [n]} X_i$ and $\mu = \mathbb{E}[X]$. Then, the following Chernoff bounds hold:

(*i*) *For any* $\delta \in (0,1]$, *we have*

$$\mathsf{Pr}\left[X \geq (1+\delta)\mu\right] \;\leq\; \exp\left[\frac{-\mu\delta^2}{3}\right].$$

(*ii*) *For any* $\delta \in (0,1)$, *we have*

$$\mathsf{Pr}\left[X \leq (1-\delta)\mu\right] \;\leq\; \exp\left[\frac{-\mu\delta^2}{2}\right].$$

We omit the proof of Theorem 2.2.1 and refer to the textbook by Mitzenmacher and Upfal [MU05].

2.3 Asymptotic Notation

In the remainder, we will typically measure the computational efficiency of an algorithm by the number of performed basic operations as a function of its input length. In other words, the efficiency of an algorithm is captured by a function $T : \mathbb{N} \to \mathbb{N}$ with $T(n)$ being the maximum number of basic operations that the algorithm performs on inputs of length n. However, this function can be overly dependent on the details of our definition of a basic operation. In order to help us ignoring low-level details and focus on the big picture, we define the following well-known notation introduced by P. Bachmann [B92] and E. Landau [L09].

Definition 2.3.1 (Bachmann–Landau Notation)
Let f *and* g *be two functions mapping from* $\mathbb{N}$ *to* $\mathbb{N}$. *We say that*

- $f = \mathrm{O}(g)$ *if there is a constant* $c > 0$ *and* $n_0 \in \mathbb{N}$ *such that* $f(n) \leq c \cdot g(n)$ *for every* $n \geq n_0$,
- $f = \Omega(g)$ *if there is a constant* $c > 0$ *such that for infinitely many* $n \in \mathbb{N}$, $c \cdot g(n) \leq f(n)$ *holds,*
- $f = \Theta(g)$ *if* $f = \mathrm{O}(g)$ *and* $g = \mathrm{O}(f)$,
- $f = o(g)$ *if for every* $\epsilon > 0$, *there is a natural number* n_ε *such that* $f(n) \;\leq\; \varepsilon \cdot g(n)$ *for every* $n \geq n_\varepsilon$, *and*

- *$f = \omega(g)$ if $g = o(f)$.*

In order to emphasize the input parameter, we write $f(n) = \mathrm{O}(g(n))$ instead of $f = O(g)$. Analogously, we use a similar notation for $o, \Omega, \omega, \Theta$.

For a function $f : \mathbb{R}^n \to \mathbb{R}$, we write $f(x_1, x_2, \ldots, x_n) = \mathrm{poly}(x_1, x_2, \ldots, x_n)$ if $f(x_1, x_2, \ldots, x_n) = (x_1 \cdot x_2 \cdots x_n)^{\mathrm{O}(1)}$.

2.4 Hypergraphs

A *hypergraph* $\mathcal{H}$ is a pair $(V(\mathcal{H}), E(\mathcal{H}))$, where $V(\mathcal{H})$ is the vertex set of $\mathcal{H}$ and $E(\mathcal{H})$ the edge set of $\mathcal{H}$ satisfying $E(\mathcal{H}) \subseteq 2^{V(\mathcal{H})}$. For every vertex $v \in V(\mathcal{H})$ in a given hypergraph $\mathcal{H}$, we define its *degree* in $\mathcal{H}$, denoted $d_{\mathcal{H}}(v)$, by $|\{e \in E(\mathcal{H}) \mid v \in e\}|$. Analogously, we define the degree of a set $S \subseteq V(\mathcal{H})$ in $\mathcal{H}$ to be $d_{\mathcal{H}}(S) = |\{e \in E(\mathcal{H}) \mid S \subseteq e\}|$. The *neighborhood* $N_{\mathcal{H}}(v)$ of a vertex v in $\mathcal{H}$ is defined as $N_{\mathcal{H}}(v) = \bigcup_{e \in E : v \in e} e \backslash \{v\}$. The *maximum degree* of a given hypergraph $\mathcal{H}$ is denoted by $\Delta_{\mathcal{H}}$, where $\Delta_{\mathcal{H}} = \max_{v \in V(\mathcal{H})} \{d_{\mathcal{H}}(v)\}$. We introduce the *average degree* $\overline{d}_{\mathcal{H}}$ of a hypergraph $\mathcal{H}$ defined as follows.

$$\overline{d}_{\mathcal{H}} = \frac{\sum\limits_{v \in V(\mathcal{H})} d_{\mathcal{H}}(v)}{|V(\mathcal{H})|}$$

A hypergraph $\mathcal{H}$ is said to be *d-regular* whenever $d_{\mathcal{H}}(v) = d \in \mathbb{N}$ for all $v \in V(\mathcal{H})$ and *regular* whenever there exists a $d \in \mathbb{N}$ such that $\mathcal{H}$ is d-regular. We refer to $\mathcal{H}$ as *r-nearly regular* if there exists a constant $r \in (0,1]$ with $\overline{d}_{\mathcal{H}} \geq r \cdot \Delta_{\mathcal{H}}$ and *nearly regular* if there exists a constant $r \in (0,1]$ such that $\mathcal{H}$ is r-nearly regular.

A *k-uniform hypergraph* or simply *k-hypergraph* is a hypergraph $\mathcal{H}$ with the restriction that every edge $e \in E(\mathcal{H})$ has size $|e| = k$. In particular, we refer to a 2-hypergraph simply as *graph* denoted by $\mathcal{G}$. In the remainder, unless stated explicitly, we will suppose that $k = \mathrm{O}(1)$.

A k-hypergraph $\mathcal{H}$ is said to be *k-partite* whenever $V(\mathcal{H})$ can be partitioned in k classes $V_1(\mathcal{H}), \ldots, V_k(\mathcal{H})$ such that for all $e \in E(\mathcal{H})$ and all $i \in [k]$, we have $|e \cap V_i(\mathcal{H})| = 1$. If the vertex partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$ of a k-partite k-hypergraph $\mathcal{H}$ is given as a part of the input, we assume that $|V_{i+1}(\mathcal{H})| \leq |V_i(\mathcal{H})|$ for all $i \in [k-1]$.

Matchings, Vertex Cover and Independent Sets

A *vertex cover* of a k-hypergraph $\mathcal{H}$ is a set $C \subseteq V(\mathcal{H})$ such that for all edges $e \in E(\mathcal{H})$, we have $e \cap C \neq \emptyset$. An *independent set* of a k-hypergraph $\mathcal{H}$ is a set $I \subseteq V(\mathcal{H})$ having the property that $V(\mathcal{H}) \backslash I$ is a vertex cover of $\mathcal{H}$. A clique of $\mathcal{H}$ is a set $K \subseteq V(\mathcal{H})$ with $|K| \geq k$ such that for all distinct $v_1, \ldots, v_k \in K$, we have $\{v_1, \ldots, v_k\} \in E(\mathcal{H})$. A *matching* of a k-hypergraph $\mathcal{H}$ is a set $M \subseteq E(\mathcal{H})$ such that for all distinct edges $e_1, e_2 \in M$, we have $e_1 \cap e_2 = \emptyset$. A matching $M \subseteq E(\mathcal{H})$ of $\mathcal{H}$ is called *maximal* if for all $e \in E(\mathcal{H}) \backslash M$, there is a $e_0 \in M$ with $e_0 \cap e \neq \emptyset$. A *maximum matching* of $\mathcal{H}$ is a maximal matching of $\mathcal{H}$ with maximum cardinality. For a set $E \subseteq E(\mathcal{H})$ of edges, we define $V(E) = \bigcup\limits_{e \in E} e$. A matching M of a hypergraph $\mathcal{H}$ is called perfect if $V(M) = V(\mathcal{H})$.

Induced Subgraphs and Basic Operations

Given a k-hypergraph $\mathcal{H}$, we call $\mathcal{H}'$ a subhypergraph of $\mathcal{H}$ if $V(\mathcal{H}') \subseteq V(\mathcal{H})$ and $E(\mathcal{H}') \subseteq E(\mathcal{H})$. $\mathcal{H}'$ is said to be an induced subhypergraph of $\mathcal{H}$ if $\mathcal{H}'$ contains all edges $e \in E(\mathcal{H})$ with $e \subseteq V(\mathcal{H}')$. The subhypergraph of $\mathcal{H}$ induced by a vertex set $U \subseteq V(\mathcal{H})$ is denoted by $\mathcal{H}[U]$.

Directed Graphs and Multi-graphs

A *directed graph* $\mathcal{G}$ or simply *digraph* is a pair $(V(\mathcal{G}), E(\mathcal{G}))$ with vertex set $V(\mathcal{G})$ and arc set $E(\mathcal{G}) \subseteq V(\mathcal{G}) \times V(\mathcal{G}) \backslash \{(v,v) \mid v \in V(\mathcal{G})\}$. A (directed) *multigraph* $\mathcal{G}$ is a pair $(V(\mathcal{G}), E(\mathcal{G}))$, where $E(\mathcal{G})$ is a multiset of edges (arcs).

Given a directed graph G and $E' \subseteq E(G)$, for $e = (x,y) \in E(G)$, we define $V(e) = \{x,y\}$ and $V(E') = \bigcup_{e \in E'} V(e)$. For convenience, we abbreviate a sequence of arcs (x_1,x_2), $(x_2,x_3), \ldots, (x_{n-1},x_n)$ by $x_1 \to x_2 \to x_3 \to \ldots \to x_{n-1} \to x_n$. In the undirected case, we use sometimes $x_1 - x_2 - x_3 - \ldots - x_{n-1} - x_n$ instead of $\{x_1,x_2\}$, $\{x_2,x_3\}, \ldots, \{x_{n-1},x_n\}$. Given a directed (multi)graph $\mathcal{G}$, an *Eulerian* cycle in $\mathcal{G}$ is a directed cycle that traverses all edges of $\mathcal{G}$ exactly once. We refer to $\mathcal{G}$ as *Eulerian*, if there exists an Eulerian cycle in $\mathcal{G}$.

2.5 Symmetric and Asymmetric Metric Spaces

A asymmetric metric space is an ordered pair (V, d), where V is a set and d an asymmetric metric on V, that is, a function $d: V \times V \to \mathbb{R}$ such that for all $x, y, z \in V$, the following holds.

$$
\begin{aligned}
(i) \quad & d(x,y) \ \geq \ 0 \\
(ii) \quad & d(x,y) = 0 \quad \text{if and only if } x = y \\
(iii) \quad & d(x,z) \ \leq \ d(x,y) + d(y,z)
\end{aligned}
$$

A asymmetric metric space (V, d) is called metric space if for all $x, y \in V$, we have that $d(x, y) = d(y, x)$.

CHAPTER 3

Basic Complexity Classes

One subject of computational complexity theory is the classification of computational problems by their intrinsic difficulty. In this chapter, we only give a brief introduction and define the basic concepts of the computational model, complexity classes and probabilistic computational model. Readers asking for a more profound introduction into this important field of computer science are referred to the text book by Arora and Barak [AB09].

3.1 The Computational Model

Since we are interested in issues of computational efficiency, we need to define a mathematical model that is sufficient for studying questions about computation and its efficiency. For this reason, Turing [T36] introduced the model of *Turing machines* (TM for short) in 1936 in order to formalize the intuitive notion of an algorithm. A TM $\mathcal{M}$ consists of a *finite control* and a finite number of *tapes*. $\mathcal{M}$ has read/write access to its tapes by means of read/write-heads. The finite control contains a finite program coordinating the movement of its read/write-heads, the write operations in dependence of the symbols of the input at the current tape positions and the actual state of the program stored in the finite control. Let us give the formal definition of a TM.

Definition 3.1.1 (Turing Machine $\mathcal{M}$)
A k-tape TM *is a tuple $\mathcal{M} = (Q, \delta, q_0, F, \Gamma)$ with*

- *Q is a finite set of states*
- *$q_0 \in Q$ is the starting state,*
- *$F \subseteq Q$ is the set of accepting states,*
- *Γ is a finite set of tape symbols containing at least two elements,*
- *δ is the transition function of the finite control*

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, N, R\}$$

$\delta(q, a_1, \dots, a_k) = (q', a'_1, \dots, a'_k, \vec{d})$ with $\vec{d} \in \{L, N, R\}^k$ means that the machine is initially in the state q and the i-th head over the tape symbol a_i, the machine writes the symbol a'_i replacing a_i, proceeds to state q' and moves the head one cell in direction $\vec{d}_i$ for all $i \in [k]$, where L, N and R corresponds to the directions left, neutral and right, respectively.

After having fixed a model of computation, we are going to formalize the notion of running time and computation of a function.

Definition 3.1.2 (Computation of a Function and Running Time)
Let $f : \{0,1\}^* \to \{0,1\}^*$ and $T : \mathbb{N} \to \mathbb{N}$ be some functions. Given a Turing machine $\mathcal{M}$, we say that $\mathcal{M}$ computes f if for every $x \in \{0,1\}^*$, $\mathcal{M}$ initialized to the start configuration on input x reaches an accepting state with $f(x)$ written on its output tape. In addition, we say that $\mathcal{M}$ computes f in $T(n)$-time if $\mathcal{M}$ computes f and for every $x \in \{0,1\}^*$, $\mathcal{M}$ on input x requires at most $T(|x|)$ steps for its computation.

We are going to define our first complexity class. A *complexity class* is a set of functions for each of which there exist a Turing machine computing the function within a specified resource bound. Boolean functions $f : \{0,1\}^* \to \{0,1\}$ specify *decision problems* or *languages*. In the remainder, we say that a Turing machine $\mathcal{M}$ *decides* a language $L \subseteq \{0,1\}^*$ if $\mathcal{M}$ computes the function $f_L : \{0,1\}^* \to \{0,1\}$, where $f_L(x) = 1 \Leftrightarrow x \in L$.

Let us now introduce the complexity class **DTIME** that was defined by Hartmanis and Stearns [HS65] in 1965.

Definition 3.1.3 (The Class **DTIME**)
Let $T : \mathbb{N} \to \mathbb{N}$ be some function. A language $L \subseteq \{0,1\}^*$ is in $\mathbf{DTIME}(T(n))$ iff there exists a TM that decides L in $c \cdot T(n)$ time for some constant $c > 0$.

3.2 The Complexity Classes P and NP

In order to formalize the term efficient computation, Cobham [C64] defined the class **P**. In the context of presenting a polynomial time algorithm for finding a maximum matching in general graphs, Edmonds [E65] made a similar suggestion. Let us give the definition of the complexity class **P**.

Definition 3.2.1 (The Class **P**)
The class **P** *contains all languages* $L \subseteq \{0,1\}^*$ *that can be decided by a Turing machine in polynomial time or equivalently,*

$$\mathbf{P} \quad = \bigcup_{p=\text{poly}(n)} \mathbf{DTIME}(p).$$

The class **P** contains all decision problems that can be solved efficiently. In contrast, we are going to define the class **NP** of decision problems that can be verified efficiently.

Definition 3.2.2 (The Class **NP**)
A language $L \subseteq \{0,1\}^*$ *is contained in* **NP** *if there exists a polynomial* $p = \text{poly}(n)$ *and a polynomial time Turing machine* $\mathcal{M}$ *such that for every* $x \in \{0,1\}^*$,

$$x \in L \Longleftrightarrow \exists c \in \{0,1\}^{p(|x|)} \textit{ such that } \mathcal{M}(x,c) = 1$$

If $x \in L$ *and* $c \in \{0,1\}^{p(|x|)}$ *satisfy* $\mathcal{M}(x,c) = 1$, *then, we call* c *a certificate for* x.

Let us give an example of a decision problem that belongs to the class **NP**. We denote by the SAT problem the language of all satisfiable Boolean formulae in CNF form (shorthand for Conjunctive Normal Form) and by the kSAT problem the language of all satisfiable Boolean formulae in CNF form, in which all clauses contain at most k literals.

Around 1971, Cook and Levin independently developed the notion of **NP**-completeness and proved that the SAT problem is **NP**-complete. Soon after that, Karp [K75] showed that the decision version of 21 important problems in combinatorial optimization are in fact **NP**-complete. In order to define **NP**-completeness, we need to introduce the notion of Karp-reductions.

Definition 3.2.3 (Karp-Reductions, **C**-Hardness and **C**-Completeness)
We say that a language $A \subseteq \{0,1\}^*$ *is polynomial time Karp reducible to a language* $B \subseteq \{0,1\}^*$ (*short: polynomial time reducible*), *denoted* $A \leq_p B$, *if*

there is a polynomial time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that

$$\textit{for every } x \in \{0,1\}^*,\ x \in A \textit{ if and only if } f(x) \in B.$$

Let $\mathbf{C}$ be a complexity class. We say that B is $\mathbf{C}$-hard if $A \leq_p B$ for every $A \in \mathbf{C}$. We say that B is $\mathbf{C}$-complete if B is $\mathbf{C}$-hard and $B \in \mathbf{C}$.

The following theorem is due to Cook and Levin providing us with our first **NP**-complete problems.

Theorem 3.2.1 (Cook-Levin Theorem [C71],[L73])
*The SAT problem and the 3SAT problem are both **NP**-complete.*

3.3 Randomized Computation

In order to formalize the notion of probabilistic computation, we extend our mathematical model of computation and introduce the *probabilistic Turing machine.* The probabilistic Turing machine was firstly defined by de Leeuw, Moore, Shannon and Shapiro [LMSS55]. Gill [G77] gave the definitions of the classes **BPP** (bounded error probabilistic polynomial time), **RP** (randomized polynomial time) and **ZPP** (zero error probabilistic polynomial time).

Let us start with the definition of a probabilistic Turing machine.

Definition 3.3.1 (Probabilistic Turing Machine (PTM))
A probabilistic Turing machine $\mathcal{M}$ is a Turing machine with two transition functions δ_0 and δ_1. Given an input $x \in \{0,1\}^*$, $\mathcal{M}$ chooses in each step to apply the transition function δ_i with $i \in \{0,1\}$ each with probability $1/2$. In every step, the choice is made independently of all previous choices. The machine only outputs 1 (YES) or 0 (NO). We denote by $\mathcal{M}(x)$ the random variable corresponding to the value that $\mathcal{M}$ writes on input x. Let $T : \mathbb{N} \to \mathbb{N}$ be a function. We say that $\mathcal{M}$ runs in $T(n)$ time if for any input $x \in \{0,1\}^*$, $\mathcal{M}$ halts on x requiring at most $T(|x|)$ steps regardless of the random choices.

We are going to introduce the class **BPP** that implements the term efficient probabilistic computation. The class **BPP** captures the nature of probabilistic algorithms with two sided error. It means that the machine $\mathcal{M}$ is allowed to output both 0 when $x \in L$ and 1 when $x \notin L$.

Definition 3.3.2 (The Classes **BPTIME** and **BPP**)
Let $T : \mathbb{N} \to \mathbb{N}$ be a function, $L \subseteq \{0,1\}^*$ a language and f_L its corresponding Boolean function. We say that a PTM *$\mathcal{M}$ decides L in $T(n)$ time if for every $x \in \{0,1\}^*$, $\mathcal{M}$ halts within $T(|x|)$ steps regardless of its random choices and*

$$\mathsf{Pr}[\mathcal{M}(x) = f_L(x)] \geq \frac{2}{3}$$

We let **BPTIME**$(T(n))$ *be the class of languages decided by* PTM*s in* $\mathrm{O}(T(n))$ *time and define* **BPP**$= \bigcup_{k \in \mathbb{N}}$ **BPTIME**(n^k).

As mentioned above, the class **BPP** captures the nature of probabilistic algorithms with two sided error. However, as many probabilistic algorithms have the property of one sided error, we introduce the class **RP** to capture this behavior.

Definition 3.3.3 (The Classes **RTIME** and **RP**)
Let $T : \mathbb{N} \to \mathbb{N}$ be a function. **RTIME**$(T(n))$ *contains the languages $L \subseteq \{0,1\}^*$, for which there is a is a* PTM *$\mathcal{M}$ running in $T(n)$ time and satisfying the following conditions.*

$$x \in L \Rightarrow \mathsf{Pr}[\mathcal{M}(x) = 1] \geq \frac{2}{3}$$
$$x \notin L \Rightarrow \mathsf{Pr}[\mathcal{M}(x) = 1] = 0$$

We define **RP** $= \bigcup_{k \in \mathbb{N}}$ **RTIME**(n^k).

For a PTM $\mathcal{M}$, a function $T : \mathbb{N} \to \mathbb{N}$ and input $x \in \{0,1\}^*$, we define the random variable $T_{\mathcal{M},x}$ to be the running time of $\mathcal{M}$ on input x. In particular, we set $\mathsf{Pr}[T_{M,x} = T(|x|)] = p$ if $\mathcal{M}$ on input x halts in at most $T(|x|)$ steps with probability p over all random choices of $\mathcal{M}$. We say that $\mathcal{M}$ has *expected running time* $T(n)$ if for every $x \in \{0,1\}^*$, the expectation $\mathbb{E}[T_{M,x}]$ is at most $T(|x|)$. We are going to define *zero error machines* which are PTMs with the property that they never err.

Definition 3.3.4 (The Classes **ZTIME** and **ZPP**)
Let $T : \mathbb{N} \to \mathbb{N}$ be a function. The class $\mathbf{ZTIME}(T(n))$ contains all languages $L \subseteq \{0,1\}^*$ for which there is a PTM $\mathcal{M}$ *with $\mathcal{M}(x) = f_L(x)$ for every input $x \in \{0,1\}^*$ and expected running time* $\mathrm{O}(T(n))$.
We define $\mathbf{ZPP} = \bigcup_{k \in \mathbb{N}} \mathbf{ZTIME}(n^k)$.

CHAPTER 4

Complexity of Optimization Problems

In this chapter, we are concerned with the computational complexity of optimization problems. Most of this material deals with approximation algorithms, hardness of approximation and lower bound techniques. For a broader overview on this topic, we recommend the text books by Ausiello et al. [ACG+99] and by Arora and Barak [AB09].

This chapter is organized as follows. In Section 4.1, we define optimization problems. In Section 4.2, we formalize the notion of approximation algorithms and approximation classes. In Section 4.3, we review basic facts on gap and promise problems. In Section 4.4, we focus on approximation preserving reductions. In Section 4.5, we give a brief introduction to probabilistically checkable proofs. In Section 4.6 and 4.7, we consider the LABEL COVER and the MAX-E3LIN2 problem, respectively. In Section 4.8, we formulate the Unique Games Conjecture. In Section 4.9, we survey some results on the approximation hardness of bounded occurrence CSPs.

4.1 Optimization Problems

As we want to extend complexity theory from decision problems to optimization problems, we need to introduce appropriate definitions. In addition, we will discuss the relationships between the complexity of optimization problems and the complexity of decision problems. Let us first give the formal definition of an optimization problem.

Definition 4.1.1 (Optimization Problem)
An optimization problem Π *is specified by a 4-tuple* $(\mathcal{I}_\Pi, \mathcal{S}_\Pi, \mathrm{m}_\Pi, goal_\Pi)$*, where*

- $\mathcal{I}_\Pi \subseteq \{0,1\}^*$ *is the set of valid instances of* Π*,*
- $\mathcal{S}_\mathsf{P}$ *is a function that associates to each valid instance* I *the set of feasible solutions* $\mathcal{S}_\Pi(I)$ *of* I *with* $\mathcal{S}_\Pi(I) \subseteq \{0,1\}^*$*,*
- $\mathrm{m}_\Pi : \{(I,S) \mid I \in \mathcal{I}_\Pi \text{ and } S \in \mathcal{S}_\Pi(I)\} \to \mathbb{N}$ *is the objective function and* $\mathrm{m}_\Pi(I,S)$ *is the objective value of the feasible solution* S *with respect to* I*,*

- *$goal_\Pi \in \{\mathsf{MIN}, \mathsf{MAX}\}$ characterizes the type of optimization problem.*

In the remainder, we will drop the subscript Π, when the context is clear. Given an instance I of an optimization problem Π, we denote by $\mathcal{S}^*_\Pi(I)$ the set of all optimal solutions of I. More formally, it is the set of all $S^* \in \mathcal{S}_\Pi(I)$ such that

$$\mathrm{m}_\Pi(I, S^*) = goal_\Pi\{\mathrm{m}_\Pi(I, S') \mid S' \in \mathcal{S}_\Pi(I)\}.$$

In the following, we will assume that there exists always optimal solutions provided that $\mathcal{S}_\Pi(I) \neq \varnothing$. The corresponding objective value of any $S^* \in \mathcal{S}^*_\Pi(x)$ is denoted by $\mathrm{opt}_\Pi(I)$.

The Classes PO and NPO

We now define the optimization counterparts of the classes **P** and **NP**. Let us start with the definition of **NPO**.

Definition 4.1.2 (The class **NPO**)
NPO *is the class of all optimization problems* $\Pi = (\mathcal{I}_\Pi, \mathcal{S}_\Pi, \mathrm{m}_\Pi, goal_\Pi)$ *such that*

- $\mathcal{I}_\Pi \in \mathbf{P}$ *meaning the set of valid instances is decidable in polynomial time.*

- *there is a polynomial p such that given $I \in \mathcal{I}_\Pi$, for all $S \in \mathcal{S}_\Pi(I)$, we have $|S| \leq p(|I|)$. Furthermore, for all S' with $|S'| \leq p(|I|)$, we can decide in* $\mathrm{poly}(|I|)$ *time whether $S \in \mathcal{S}_\Pi(I)$.*

- m_Π *is computable in polynomial time.*

Based on the previous definition, we define the class of optimization problems that are solvable in polynomial time.

Definition 4.1.3 (The class **PO**)
PO *is the class of all optimization problems* $\Pi \in$ **NPO**, *for which there exists a polynomial time algorithm that, for any valid instance $I \in \mathcal{I}_\Pi$, returns an optimal solution $S \in \mathcal{S}^*_\Pi(I)$.*

Next, we are going to define approximation algorithms for optimization problems.

4.2 Approximation Classes

In the most general sense, an approximation algorithm computes on input of a valid instance I of an optimization problem a feasible solution.

Definition 4.2.1 (Approximation Algorithm)
Let Π be an optimization problem. An algorithm $\mathcal{A}$ is an approximation algorithm for Π if for all instances $I \in \mathcal{I}_\Pi$, we have $\mathcal{A}(I) \in \mathcal{S}_\Pi(I)$.

In order to classify the performance of approximation algorithms, we are going to develop a framework to measure the quality of a produced solution.

Definition 4.2.2 (Approximation Ratio)
Let Π be an optimization problem, $I \in \mathcal{I}_\Pi$ and $S \in \mathcal{S}_\Pi(I)$. The approximation ratio of S with respect to I, denoted $R(I,S)$, is defined as follows.

$$R(I,S) = \max\left\{\frac{\mathrm{m}_\Pi(I,S)}{\mathrm{opt}(I)}, \frac{\mathrm{opt}(I)}{\mathrm{m}_\Pi(I,S)}\right\}$$

Let $\alpha : \mathbb{N} \to \mathbb{Q}_+$ be a function. An approximation algorithm $\mathcal{A}$ for Π has approximation ratio α, if for all $I \in \mathcal{I}_\Pi$, we have that $R(I, \mathcal{A}(I)) \leq \alpha(|I|)$.

We are going to introduce the class **APX**. It is the set of **NPO** optimization problems having a polynomial time approximation algorithm with approximation ratio bounded by a constant.

Definition 4.2.3 (The class **APX**)
APX *is the class of all* **NPO** *problems Π, for which there exist a constant $\alpha \geq 1$ and a polynomial time approximation algorithm for Π with approximation ratio α.*

Some optimization problems in the class **APX** even admit a polynomial time approximation scheme, that is roughly speaking, a family of approximation algorithms that achieve an arbitrarily good approximation ratio $\alpha > 1$

for the cost of a longer runtime. Let us give the precise definition.

Definition 4.2.4 (PTAS, EPTAS, FPTAS)
Let Π be a **NPO** *problem. A family $\{\mathcal{A}_\varepsilon\}_{\varepsilon>0}$ of approximation algorithms for Π is called a polynomial time approximation scheme* (PTAS) *if for any instance $I \in \mathcal{I}_\Pi$ and every fixed $\varepsilon > 0$, the algorithm $\mathcal{A}_\varepsilon$ returns a solution with approximation ratio at most $(1+\varepsilon)$ in* $\text{poly}(|I|)$ *time.*
We call a PTAS *$\{\mathcal{A}_\varepsilon\}_{\varepsilon>0}$ for a problem Π an efficient polynomial time approximation scheme* (EPTAS) *if for any instance $I \in \mathcal{I}_\Pi$ and every fixed $\varepsilon > 0$, the running time of $\mathcal{A}_\varepsilon$ is $f(1/\varepsilon) \cdot \text{poly}(|I|)$, where $f : \mathbb{Q}_+ \to \mathbb{Q}_+$ is some function.*
A PTAS *$\{\mathcal{A}_\varepsilon\}_{\varepsilon>0}$ for Π is said to be a fully polynomial time approximation scheme* (FPTAS) *if for any instance $I \in \mathcal{I}_\Pi$ and every ε, the running time of $\mathcal{A}_\varepsilon$ is* $\text{poly}(|I|, 1/\varepsilon)$.

Accordingly, we define the classes **PTAS**, **FPTAS** and **EPTAS** , which by definition are subclasses of **APX**.

Definition 4.2.5 (The classes **PTAS**, **FPTAS** and **EPTAS**)
PTAS, **FPTAS** *and* **EPTAS** *are the classes of all optimization problems in* **NPO** *that admit a* PTAS, FPTAS *and* EPTAS, *respectively.*

In the next section, we are going to define gap problems and promise problems

4.3 Gap and Promise Problems

In order to relate the approximation hardness of optimization problems to the hardness of decision problems, we are going to introduce the notion of promise problems.

Definition 4.3.1 (Promise Problem)
A promise problem Λ is a pair of non-intersecting sets denoted by $(\Lambda_{\mathsf{YES}}, \Lambda_{\mathsf{NO}})$ with $\Lambda_{\mathsf{YES}}, \Lambda_{\mathsf{NO}} \subseteq \{0,1\}^*$. The set $\Lambda_{\mathsf{NO}} \cup \Lambda_{\mathsf{YES}}$ is called the promise.

We refer to elements in Λ_{YES} as YES-instances, whereas elements in Λ_{NO}

are called NO-instances. The remaining elements in $\{0,1\}^* \setminus (\Lambda_{\mathsf{NO}} \cup \Lambda_{\mathsf{YES}})$ are called *don't care*-instances. Decision problems are a special case of promise problems, in which we have $\Lambda_{\mathsf{NO}} \cup \Lambda_{\mathsf{YES}} = \{0,1\}^*$.

Polynomial time reductions can be extended to promise problems in a very natural way. Let $\Lambda^1 = (\Lambda^1_{\mathsf{YES}}, \Lambda^1_{\mathsf{NO}})$ and $\Lambda^2 = (\Lambda^2_{\mathsf{YES}}, \Lambda^2_{\mathsf{NO}})$ be two promise problems. We say that Λ^1 is polynomial time reducible to Λ^2 if there is a polynomial time computable function $f : \Lambda^1_{\mathsf{YES}} \cup \Lambda^1_{\mathsf{YES}} \to \Lambda^2_{\mathsf{YES}} \cup \Lambda^2_{\mathsf{NO}}$ such that the following holds.

$$I \in \Lambda^1_{\mathsf{YES}} \implies f(x) \in \Lambda^2_{\mathsf{YES}}$$

and

$$I \in \Lambda^1_{\mathsf{NO}} \implies f(x) \in \Lambda^2_{\mathsf{NO}}$$

It simply means that YES-instances are mapped to YES-instances and NO-instances to NO-instances.

A promise problem Λ is $\mathbf{C}$-hard for some class $\mathbf{C}$ of decision problems, if every problem in $\mathbf{C}$ is polynomial time reducible to Λ.

Definition 4.3.2 (Gap Problem)
For an optimization problem $\Pi = (\mathcal{I}_\Pi, \mathcal{S}_\Pi, \mathrm{m}_\Pi, goal_\Pi)$ and some constants $A, B \in \mathbb{Q}^+$ with $A < B$, the corresponding promise problem $[A,B]$-GAP-Π is defined as follows.

$$\mathsf{YES}\text{-}instances \;:\; \begin{cases} \{I \in \mathcal{I}_\Pi \mid \mathrm{opt}(I) \geq B\} & \text{if} \quad goal_\Pi = \mathsf{MAX} \\ \{I \in \mathcal{I}_\Pi \mid \mathrm{opt}(I) \leq A\} & \text{if} \quad goal_\Pi = \mathsf{MIN} \end{cases}$$

$$\mathsf{NO}\text{-}instances \;:\; \begin{cases} \{I \in \mathcal{I}_\Pi \mid \mathrm{opt}(I) \leq A\} & \text{if} \quad goal_\Pi = \mathsf{MAX} \\ \{I \in \mathcal{I}_\Pi \mid \mathrm{opt}(I) \geq B\} & \text{if} \quad goal_\Pi = \mathsf{MIN} \end{cases}$$

By definition of gap problems, we obtain immediately the following lemma that relates the hardness of approximating an optimization problem to the hardness of a gap problem.

Lemma 4.3.1
Let $[A,B]$-GAP-Π be an $\mathbf{NP}$-hard gap problem of an optimization problem $\Pi \in \mathbf{NPO}$ for some constants $A, B \in \mathbb{Q}^+$ with $A < B$. Then, it is $\mathbf{NP}$-hard to

approximate the optimization problem Π *to within any constant approximation ratio less than* (B/A).

In order to prove hardness of approximation to within a particular approximation ratio, it suffices to prove hardness of the corresponding gap problem.

4.4 Approximation Preserving Reductions

There have been defined different kind of reducibilities in the literature. The AP-reducibility, which we are going to define, is sufficiently general to incorporate almost all known reducibilities while preserving a linear relation between approximation ratios. Let us give the precise definition of the AP-Reduction.

Definition 4.4.1 (AP-Reduction)
Let Π_1 *and* Π_2 *be optimization problems in* **NPO**. *We say that* Π_1 *is* AP-*reducible to* Π_2 *(in symbols:* $\Pi_1 \leq_{\mathsf{AP}} \Pi_2$*) if there is a function* $f : \{0,1\}^* \times \mathbb{Q}_+ \to \{0,1\}^*$, *a constant* $\alpha \geq 1$ *and a function* $g : \{0,1\}^* \times \{0,1\}^* \times \mathbb{Q}_+ \to \{0,1\}^*$ *such that*

- *for all* $I \in \mathcal{I}_{\Pi_1}$ *and all rational* $\beta > 1$, *we have* $f(I,\beta) \in \mathcal{I}_{\Pi_2}$,
- *for all* $I \in \mathcal{I}_{\Pi_1}$ *and all rational* $\beta > 1$, *if* $\mathcal{S}_{\Pi_1}(I) \neq \varnothing$, *then, we have that* $\mathcal{S}_{\Pi_2}(f(I,\beta)) \neq \varnothing$,
- *for all* $I \in \mathcal{I}_{\Pi_1}$, $S \in \mathcal{S}_{\Pi_2}(f(I,\beta))$, *and* $\beta > 1$, *we have* $g(I,S,\beta) \in \mathcal{S}_{\Pi_1}(I)$,
- f *and* g *are polynomial time computable for fixed* $\beta > 1$,
- *for all* $I \in \mathcal{I}_{\Pi_1}$ *and all* $S \in \mathcal{S}_{\Pi_2}(f(I,\beta))$, *if* S *is a solution to* $f(I,\beta)$ *with approximation ratio* β, *then,* $g(I,S,\beta)$ *is a solution to* I *with approximation ratio at most* $(1+\alpha(\beta-1))$.

We call the triple (f,g,α) *an* AP-*reduction from* Π_1 *to* Π_2.

We are going to state some properties of the AP-reduction. The proofs of the statements can be found in the textbook by Ausiello et al. [ACG+99].

Lemma 4.4.1
Let Π_1 and Π_2 be optimization problems.

- *If $\Pi_1 \leq_{\mathsf{AP}} \Pi_2$ and $\Pi_2 \in$ **APX**, then, we have $\Pi_1 \in$ **APX**.*
- *If $\Pi_1 \leq_{\mathsf{AP}} \Pi_2$ and $\Pi_2 \in$ **PTAS**, then, $\Pi_1 \in$ **PTAS**.*
- *The relation $\leq_{\mathsf{AP}}$ is transitive.*

Next, we are going to define the notion of completeness for optimization problems.

Definition 4.4.2 (Complete Optimization Problems)
*Let $\boldsymbol{C}$ be a class of optimization problems in **NPO**. An optimization problem Π is $\boldsymbol{C}$-hard (under AP-reductions) if for all $\Pi_2 \in \boldsymbol{C}$, we have $\Pi_2 \leq_{\mathsf{AP}} \Pi$. The optimization problem Π is said to be $\boldsymbol{C}$-complete if it is contained in $\boldsymbol{C}$ and $\boldsymbol{C}$-hard.*

As an implication of Lemma 4.4.1, we obtain the following statement.

Corollary 4.4.1
*Let $\boldsymbol{C}$ be a class of optimization problems in **NPO** and Π, Π' be optimization problems. If $\Pi \leq_{\mathsf{AP}} \Pi'$ and Π is $\boldsymbol{C}$-hard, then, Π' is also $\boldsymbol{C}$-hard.*

4.5 Probabilistically Checkable Proofs

A *polynomial time probabilistic verifier* $\mathcal{V}$ is a polynomial time probabilistic Turing machine with *oracle access* to a proof $\pi \in \{0,1\}^*$. That means each bit of π can be independently queried by $\mathcal{V}$ by means of a special address tape. Let us say that the verifier queries the i-th bit in the proof π, then, it writes i on the address tape and receives the bit $\pi[i]$.

In addition, we restrict verifiers to be *non-adaptive* meaning the verifier may write several positions of π at one time. If it enters the query state, it receives the values of all queried positions. But, the verifier is allowed to enter the query state only once. Therefore, the verifier has to decide in advance which bits it wants to query.

A non-adaptive probabilistic verifier $\mathcal{V}$ with access to π is called $(r(n), q(n))$*-restricted* if for all $n \in \mathbb{N}$ and on all inputs $x \in \{0,1\}^n$, the verifier $\mathcal{V}$ uses at most $r(n)$ random bits and queries at most $q(n)$ bits of π. We are ready to give the definition of the class $\mathbf{PCP}[r, q]$.

Definition 4.5.1 (The Class $\mathbf{PCP}[r, q]$)
Let $r, q : \mathbb{N} \to \mathbb{N}$ be functions and L a language. L is a member of the class $\mathbf{PCP}[r, q]$ if there exists a (r', q')-restricted non-adaptive polynomial time probabilistic verifier $\mathcal{V}$ with $r' = \mathrm{O}(r)$, $q' = \mathrm{O}(q)$ and the following properties.

- *For any $x \in L$, there is a proof π_x such that*

$$\Pr_{r \in \{0,1\}^{r(n)}} \Big[\mathcal{V}(x, r, \pi_x) = 1 \Big] \;=\; 1.$$

- *For any $x \notin L$ and for all proofs π, the following holds.*

$$\Pr_{r \in \{0,1\}^{r(n)}} \Big[\mathcal{V}(x, r, \pi) = 1 \Big] \;\le\; \frac{1}{2}$$

Following a long sequence of work, Arora and Safra [AS98] and Arora et al. [ALM+98] constructed polynomial time probabilistic verifier for the class **NP**. This result is also known as the PCP Theorem.

Theorem 4.5.1 (PCP Theorem [AS98], [ALM+98])
$\mathbf{NP} = \mathbf{PCP}\big[\log(n), 1\big]$.

The PCP Theorem implies that it is **NP**-hard to approximate the MAX-3SAT problem to within some constant $r > 1$. Moreover, Arora et al. [ALM+98] proved that these statements are equivalent.

Theorem 4.5.2 ([ALM+98])
The following two statements are equivalent:

- $\mathbf{NP}$= $\mathbf{PCP}\big[\log(n), 1\big]$,

- *There is a constant $\epsilon > 0$ such that $[1-\varepsilon, 1]$-GAP-MAX-3SAT is* **NP**-*hard.*

4.6 Label Cover Problem

In 1993, Arora, Babai, Stern and Sweedyk [ABSS93] introduced a new paradigm for proving hardness of approximation results. On the basis of their paradigm, Håstad [H01] gave tight hardness results for approximating the MAX-E3SAT problem and many other optimization problems. This paradigm is based on the hardness of approximating the LABEL COVER problem defined below.

Definition 4.6.1 (LABEL COVER Problem)
An instance $\mathcal{L}$ of the LABEL COVER *problem is defined by a tupel $(\mathcal{G}, V_1(\mathcal{G}), V_2(\mathcal{G}), [m], [n], \{\pi_e \mid e \in E(\mathcal{G})\})$ consisting of*

- *A bipartite graph $\mathcal{G}$ with bipartition $\{V_1(\mathcal{G}), V_2(\mathcal{G})\}$.*
- *Every vertex in $V_1(\mathcal{G})$ is supposed to get a label from a set $[m]$ and every vertex in $V_2(\mathcal{G})$ is supposed to get a label from a set $[n]$, where $n \geq m$.*
- *Every edge $e \in E(\mathcal{G})$ is associated with a projection $\pi_e : [n] \to [m]$.*

A labeling L for $\mathcal{L}$ is a mapping $L : V_1(\mathcal{G}) \to [m]$, $L : V_2(\mathcal{G}) \to [n]$. We say that L satisfies an edge $\{v, w\}$ if $\pi_{\{v,w\}}(L(w)) = L(v)$. The task is to find a labeling that maximizes the number of satisfied edges. Let us define $\text{opt}(\mathcal{L})$ *to be the maximum fraction of edges that are satisfied by any labeling meaning*

$$\text{opt}(\mathcal{L}) = \max_{\textit{all labeling } L} \{r \mid \exists\ L \textit{ for } \mathcal{L} \textit{ that satisfies } r \cdot |E(\mathcal{G})| \textit{ edges}\}.$$

There is a natural correspondence between PCP verifier and the LABEL COVER problem. Moreover, the PCP Theorem implies that it is **NP**-hard to approximate the LABEL COVER problem to within any constant approximation ratio less than r for some $r > 1$.

Theorem 4.6.1 ([AS98], [ALM⁺98])
*There exists a constant $\varepsilon \in (0,1)$ such that $[1, 1-\varepsilon]$-*GAP-LABEL COVER *is* **NP***-hard.*

Proof of Theorem 4.6.1.
We prove Theorem 4.6.1 by constructing a reduction from the MAX-3SAT problem to the LABEL COVER problem. Given an instance f of the MAX-3SAT problem, construct a bipartite graph $\mathcal{G}$ such that all vertices on the left side $V_1(\mathcal{G})$ correspond to clauses of f and the vertices on the right side $V_2(\mathcal{G})$ correspond to variables in f. Vertices $v_i^1 \in V_1(\mathcal{G})$ and $v_x^2 \in V_2(\mathcal{G})$ are connected by an edge if the variable x occurs in the clause C_i of f.

We are now going to describe the labels and projections. A label of a vertex v_i^1 decodes one of the 7 satisfying assignments to the variables of C_i and the label of a vertex v_x is supposed to be the value of the variable x. Therefore, the projection $\pi_{\{v_i^1, v_x^2\}}$ is just a consistency check which verifies that the label of v_x^1 is as claimed by the label of v_i^2. ∎

Thereafter, Raz [R98] gave a low error version of the previous theorem. It improves the inapproximability factor in Theorem 4.6.1 from some constant to any constant. This theorem is also known as the *Parallel Repetition Theorem.*

Theorem 4.6.2 (Parallel Repetition Theorem [R98])
For every $\delta > 0$, there exist label sizes m and n such that given an instance of the LABEL COVER *problem $\mathcal{L} = (\mathcal{G}, V_1(\mathcal{G}), V_2, [m], [n], \{\pi_e \mid e \in E(\mathcal{G})\})$, it is* **NP***-hard to tell whether $\operatorname{opt}(\mathcal{L}) = 1$ or $\operatorname{opt}(\mathcal{L}) \leq \delta$.*

More generally, the LABEL COVER problem is a *Constraint Satisfaction Problem* (CSP). An instance of a CSP consists of a set of variables, a set of values for the variables and a set of constraints that restrict the combinations of values for certain subsets of variables. Let us give the formal definition of a CSP.

Definition 4.6.2 (Constraint Satisfaction Problem (CSP))
An instance of a Constraint Satisfaction Problem is defined by a tupel

(X, D, Γ) *consisting of*

- *A set of* n *variables* X.
- *Every variable in* X *is supposed to get a value from the domain* D.
- Γ *is a collection of constraints* R_i*, where* R_i *is a relation* $R_i \subseteq D^k$ *with arity* k.

An assignment $d \in D^k$ *satisfies a constraint* R_i *if* $d \in R_i$*. The task is to find an assignment* $d \in D^n$ *to the variables in* X *that maximizes the number of satisfied constraints.*

4.7 Håstad's 3-Bit PCP

A sequence of developments in constructing PCP verifier with low query complexity culminated in the results due to Håstad proving that every language in **NP** has a PCP verifier querying 3 bits and having error probability arbitrarily close to 1/2. Furthermore, the verifier's decision process consists of checking the parity of the 3 bits. This characterization lead to tight inapproximability results for various problems such as the MAX-E3SAT problem and the MAX-E3LIN2 problem. Let us first give the definition of the MAX-E3LIN2 problem.

Definition 4.7.1 (MAX-E3LIN2 problem)

Instances: *A system of linear equations* $\mathscr{L}$ *with variables* $\{x_1, \dots, x_n\}$ *and all equations are of the form* $x_i \oplus x_j \oplus x_k = c_{ijk}$ *with* $c_{ijk} \in \{0, 1\}$

Solutions: *Assignment* $\phi : \{x_1, \dots, x_n\} \to \{0, 1\}$

Task: *Maximize the number of satisfied equations in* $\mathscr{L}$

Let us formulate the tight hardness result for the MAX-E3LIN2 problemdue to Håstad [H01].

Theorem 4.7.1 ([H01])

For every $\varepsilon > 0$*, the* $\left[1 - \varepsilon, \frac{1}{2} + \varepsilon\right]$-GAP-MAX-E3LIN2 *problem is* **NP**-*hard.*

Note that given an instance of the MAX-E3LIN2 problem, a random assignment satisfies half of the equations in expectation. On the other hand, it is possible to use Gaussian elimination to efficiently check whether any system of linear equations mod 2 can be completely satisfied. Hence, the result above is tight. By using a simple gadget construction, we obtain another tight inapproximability result for the MAX-3SAT problem.

Corollary 4.7.1
*For every $\varepsilon > 0$, the $\left[\frac{7}{8} + \varepsilon, 1 - \varepsilon\right]$-GAP-MAX-3SAT problem is **NP**-hard.*

4.8 Unique Games Conjecture

Håstad's work gives optimal inapproximability results for the MAX-E3SAT and the MAX-E3LIN2 problem. For many other problems, like the VERTEX COVER problem or the MAX-CUT problem, the status remained open. In 2002, Khot [K02] formulated the Unique Games Conjecture in order to prove hardness of approximation results for **NP**-hard problems that researchers have been unable to prove otherwise. This conjecture postulates the inapproximability of a restricted version of the LABEL COVER problem, where the projections π_e on the edges are permutations. This restricted version is called the UNIQUE GAMES problem.

Definition 4.8.1 (UNIQUE GAMES Problem)
An instance of the UNIQUE GAMES problem is a restricted instance of the LABEL COVER problem $\mathcal{L} = (V(\mathcal{G}), W(\mathcal{G}), E(\mathcal{G}), [m], [n], \{\pi_e \mid e \in E(\mathcal{G})\})$, in which $m = n$ and for all edges $e \in E(\mathcal{G})$, $\pi_e : [n] \to [n]$ is a bijection.

Based on the Unique Games Conjecture, it was possible to prove optimal inapproximability results for the MAX-CUT problem (cf. [KKMO07]) and the VERTEX COVER problem (cf. [KR08]).

Conjecture 4.8.1 (Unique Games Conjecture (UGC) [K02])
*For every constant $\varepsilon > 0$, there is an integer k_ε such that for all instances of the UNIQUE GAMES problem $\mathcal{L}$ with label size greater than k_ε, it is **NP**-hard*

to tell whether $\mathrm{opt}(\mathcal{L}) \geq 1 - \varepsilon$ *or* $\mathrm{opt}(\mathcal{L}) \leq \varepsilon$.

We will formulate an equivalent version of the UGC. For this, we need to define a special case of the UNIQUE GAMES problem called the MAX-E2LINq problem.

Definition 4.8.2 (MAX-E2LINq problem)

Instances: *A system of linear equations* $\mathscr{L}$ *with variables* $\{x_1, \dots, x_n\}$ *and all equations are of the form* $x_i + x_j = c_{ij} \pmod{q}$ *with* $c_{ij} \in \{0, \dots, q-1\}$

Solutions: *Assignment* $\phi : \{x_1, \dots, x_n\} \to \{0, \dots, q-1\}$

Task: *Maximize the number of satisfied equations in* $\mathscr{L}$

Clearly, the MAX-E2LINq problem is a special case of the Unique Games problem with alphabet size q. By the work of Khot, Kindler, Mossel and O'Donnell [KKMO07], it is also known that the UGC is equivalent to the following statement.

Conjecture 4.8.2 (Equivalent statement of the UGC)

For any constant $\varepsilon > 0$*, there exists large enough* $q \in \mathbb{N}$ *such that* $[1-\varepsilon, \varepsilon]$-GAP-MAX-E2LINq *is* **NP***-hard.*

We now formalize the statement that a gap version of an optimization problem is UG-hard.

Definition 4.8.3 (UG-Hardness)

Let Π *be an* **NPO** *optimization problem and* $0 < A < B$ *some constants. We say that* $[A, B]$-GAP-Π *is* UG*-hard if for some* $\gamma > 0$*, there is a polynomial time reduction from the* $[\gamma, 1-\gamma]$-GAP-UNIQUE GAMES *problem to* $[A, B]$-GAP-Π.

In this case, we also say that Π is UG-hard to approximate to within any approximation ratio less than (B/A). Furthermore, we note that if the UGC holds, then, for a gap problem being UG-hard is equivalent to the gap problem being **NP**-hard.

4.9 Bounded Occurrence CSPs

In order to prove hardness of approximation results for several optimization problems restricted to instances with bounded occurrences or bounded degree, Berman and Karpinski [BK99], see also [BK01] and [BK03], introduced the following CSP called the MAX-HYBRID-LIN2 problem. It will play a key role in the subsequent chapters, in which we construct several reductions from this problem.

Definition 4.9.1 (MAX-HYBRID-LIN2 problem)

Instances: *A system of linear equations $\mathscr{L}$ with variables $\{x_1, \dots, x_n\}$, m_2 equations of the form $x_i \oplus x_j = b_{ij}$ with $b_{ij} \in \{0,1\}$ and m_3 equations of the form $x_i \oplus x_j \oplus x_k = c_{ijk}$ with $c_{ijk} \in \{0,1\}$.*

Solutions: *Assignment $\phi : \{x_1, \dots, x_n\} \to \{0,1\}$*

Task: *Maximize the number of satisfied equations in $\mathscr{L}$*

In particular, Berman and Karpinski [BK99] constructed special instances of the MAX-HYBRID-LIN2 problem with bounded occurrences of variables, for which they proved the following inapproximability result.

Theorem 4.9.1 ([BK99])

For any constant $\epsilon > 0$, there exists instances of the MAX-HYBRID-LIN2 *problem with 42ν variables, 60ν equations with exactly two variables, and 2ν equations with exactly three variables such that:*

(i) *Each variable occurs exactly three times.*

(ii) *Either there is an assignment to the variables that leaves at most $\epsilon\nu$ equations unsatisfied, or else every assignment to the variables leaves at least $(1-\epsilon)\nu$ equations unsatisfied.*

(iii) *It is* **NP***-hard to decide which of the two cases in item (ii) above holds.*

The instances of the MAX-HYBRID-LIN2 problem produced in Theorem 4.9.1 have an even more special structure, which we are going to describe. The equations with three variables are of the form $x \oplus y \oplus z = b$ with $b \in \{0,1\}$ and stem from Theorem 4.7.1. We now formulate a slightly extended version of this theorem.

Theorem 4.9.2 ([H01])
For every $\delta \in \left(0, \frac{1}{2}\right)$, there exists a positive constant B_δ and instances $\mathscr{L}^3$ of the MAX-E3LIN2 *problem with 2ν equations such that each variable in $\mathscr{L}^3$ occurs in at most B_δ equations and it is* **NP**-*hard to tell whether there is an assignment satisfying all but at most $\delta \cdot \nu$ equations, or every assignment leaves at least $(1 - \delta)\nu$ equations unsatisfied.*

Let us describe the construction due to Berman and Karpinski [BK99] and give the proof of Theorem 4.9.1. In order to reduce the occurrences of the variables in Theorem 4.9.2, we are going to introduce a special class of graphs which are called *amplifier*. Amplifier graphs are useful in proving inapproximability results for bounded occurrence CSPs.

Definition 4.9.2 (Regular Amplifier Graph)
A graph $\mathcal{G}$ is called a r-regular amplifier for a given set $X \subseteq V(\mathcal{G})$ if

- *for all $A \subseteq V(\mathcal{G})$, we have $|\{e \in E(\mathcal{G}) \mid 1 = |e \cap A|\}| \geq \min\left\{|X \cap A|, |X \backslash A|\right\}$*
- *every vertex in X has $r - 1$ neighbors and every vertex in $V(\mathcal{G}) \backslash X$ has r neighbors.*

Given a r-regular amplifier $\mathcal{G}$ for $X \subseteq V(\mathcal{G})$, the vertices in X are called *contact vertices*, whereas all $v \in V(\mathcal{G}) \backslash X$ are called *checker vertices*.

The following result was proved on the existence of 3-regular amplifier.

Theorem 4.9.3 ([BK99])
For a set of ν vertices, there is a 3-regular amplifier with 7ν vertices and 10ν edges.

In the following, we will describe the structure of the 3-regular amplifier from Theorem 4.9.3 more in detail and introduce some terminology. Due to the special structure of the 3-regular amplifier constructed by Berman and Karpinski [BK99], they also refer to it as a *wheel amplifier*. In Figure 4.1, we displayed such a 3-regular amplifier. The contact vertices of a wheel

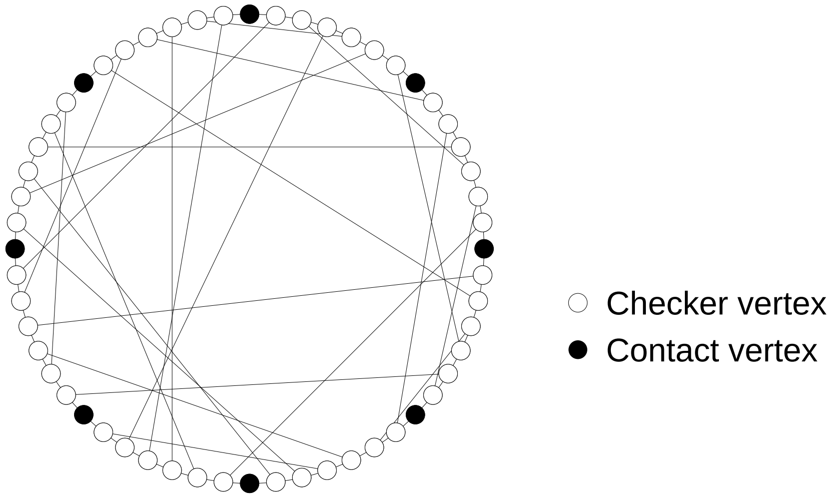


Figure 4.1: A 3-regular wheel amplifier, where checker vertices are indicated by circles (∘) and contact vertices by filled circles (•).

amplifier $\mathcal{W}$ with $B = 7\nu$ vertices $\{v_1, \dots, v_B\}$ is defined by the set $\{v_i \mid i \in [B], 0 = i \pmod 7\}$. The edges of $\mathcal{W}$ are defined by $E(\mathcal{W}) = C(\mathcal{W}) \cup M(\mathcal{W})$, where $C(\mathcal{W})$ induces a Hamiltonian cycle $C(\mathcal{W}) = \big\{\{v_i, v_{i+1}\} \mid i \in [B-1]\big\} \cup \big\{\{v_B, v_1\}\big\}$ and $M(\mathcal{W})$ is a perfect matching on the set of checker vertices.

We now give the proof of Theorem 4.9.1.

Proof of Theorem 4.9.1.
Let us fix a constant $\delta \in (0, \frac{1}{2})$. By Theorem 4.9.2, there exist instances $\mathscr{L}^3$ of the MAX-E3LIN2 problem with 2ν equations such that each variable in $\mathscr{L}^3$ occurs in at most B_δ equations and it is **NP**-hard to tell whether there is an assignment satisfying all but at most $\delta \cdot \nu$ equations, or every assignment leaves at least $(1 - \delta)\nu$ equations unsatisfied. For each variable x_i in the instance $\mathscr{L}^3$, we generate a corresponding wheel $\mathcal{W}_i$ of size $7t_i$, where x_i occurs exactly t_i times in $\mathscr{L}^3$. Since B_δ is a constant that depends only on δ, it can be accomplished in constant time. Then, we replace the occurrences

$x_i^{(1)}, \ldots, x_i^{(t_i)}$ of the variable x_i in $\mathscr{L}^3$ by newly created variables $X^i_{7\cdot 1}, \ldots, X^i_{7\cdot t_i}$ (the contact variables) in some order. Finally, we add all equations $X^i_j \oplus X^i_k = 0$ for all $\{v_j, v_k\} \in E(\mathcal{W}_i)$ to $\mathscr{L}^3$.

Let us denote this instance of the MAX-HYBRID-LIN2 problem by $\mathscr{L}$. According to Theorem 4.9.3, $\mathscr{L}$ has 2ν equations with three variables, 60ν equations with two variables and every variable in $\mathscr{L}$ occurs in exactly three equations due to the construction of a wheel.

Let ϕ denote an assignment to the variables of $\mathscr{L}$, $V_i = \{X^i_1, \ldots, X^i_{7t_i}\}$ be the variables corresponding to $\mathcal{W}_i$ and $V_i^0 = \{X^i_j \in V_i \mid \phi(X^i_j) = 0\}$. We assume that the number of contact vertices contained in V_i^0 is at most the number of contact variables contained in $V_i \backslash V_i^0$. By setting $\phi(X^i_j) = 1$ for all $X^i_j \in V_i^0$, the number of satisfied equations in $\mathscr{L}$ is not decreased due to the property of an amplifier. Notice that this fact holds for all variables associated to wheels in $\mathscr{L}$. Accordingly, we obtain that it is **NP**-hard to distinguish whether there is an assignment to the variables of $\mathscr{L}$ satisfying all but at most $\delta \cdot \nu$ equations, or every assignment leaves at least $(1-\delta)\nu$ equations unsatisfied. ■

In order to specify the equations in an instance of the MAX-HYBRID-LIN2 problem, we are going to introduce some conventions. Given a wheel $\mathcal{W}$ with $B = 7n$ vertices $V(\mathcal{W}) = \{v_1, \ldots, v_B\}$, we refer to the associated equations with two variables of the form $x_i \oplus x_{i+1} = 0$ with $i \in [B-1]$ as *cycle equations.* The equation $x_1 \oplus x_B = 0$ is called *wheel border equation.* Furthermore, we refer to equations of the form $x_i \oplus x_j = 0$ with $\{v_i, v_j\} \in M(\mathcal{W})$ as *matching equations.*

Part II

Dense Instances

CHAPTER 5

Vertex Cover of k-Hypergraphs

In this chapter, we study the approximability of the VERTEX COVER problem in dense, subdense, mildly sparse and nearly regular k-hypergraphs.

For the VERTEX COVER problem in (ε, ℓ)-dense k-hypergraphs, we design a polynomial time approximation algorithm with an improved approximation ratio for all $k > 2$ and $\ell > 0$. On the other hand, we give an optimal inapproximability result the problem. The approach designed for the (ε, ℓ)-dense case will also be applicable for the k-balanced case in Section 6.5.

For the VERTEX COVER problem in nearly regular hypergraphs, we give a randomized approximation algorithm with approximation ratio strictly less than k and running time depending on the density of the underlying k-hypergraph. More precisely, it entails the existence of quasi-polynomial and polynomial time approximation algorithms with approximation ratio less than k for mildly sparse and subdense instances, respectively. Furthermore, we obtain tight approximation lower bounds for this problem. In particular, we prove the best known approximation lower bounds for the VERTEX COVER problem in regular k-hypergraphs.

5.1 Introduction

The VERTEX COVER problem in k-uniform hypergraphs is among the most fundamental problem in combinatorial optimization. Especially, in the case $k = 2$, it is the classical VERTEX COVER problem in graphs. In 1972, Karp [K75] proved that the decision version of the problem is **NP**-complete. Accordingly, there is only little hope for efficient algorithms solving the VERTEX COVER problem to optimality. On the other hand, there exists a simple approximation algorithm for the problem with approximation ratio 2 by constructing greedily a maximal matching in the given graph. However, the currently best known approximation algorithm for the VERTEX COVER problem in graphs achieves only an approximation ratio $2-o(1)$ [K09]. For the case $k > 2$, the best known approximation algorithm is due to Halperin [H02] and yields an approximation ratio $k - o(1)$.

On the approximation hardness side, Papadimitriou and Yannakakis [PY91] proved that the VERTEX COVER problem in graphs is **APX**-hard. After that, Bellare, Goldreich and Sudan [BGS98] gave the first explicit **NP**-hardness of approximation result. In particular, they proved that it is **NP**-hard to approximate the problem to within any constant approximation ratio less than 233/218. Håstad gave an improved inapproximability result for the problem yielding an inapproximability factor of $(7/6 - \delta)$ for all $\delta > 0$. Finally, Dinur and Safra [DS05] proved that it is **NP**-hard to approximate the VERTEX COVER problem in graphs to within any constant approximation ratio less than 1.36. Assuming the UNIQUE GAMES CONJECTURE, Khot and Regev [KR08] gave an optimal inapproximability result for the problem. More precisely, they showed that it is UG-hard to approximate the VERTEX COVER problem in graphs to within any constant approximation ratio less than 2.

For larger k, the first explicit approximation hardness result proved for the VERTEX COVER problem in k-hypergraphs is due to Trevisan [T01] who considered the approximation hardness of bounded degree instances of several combinatorial problems. He achieved an inapproximability factor of $k^{1/19}$. Holmerin [H02a] studied the VERTEX COVER problem in 4-hypergraphs and proved that this restricted version of the problem is **NP**-hard to approximate to within any constant less than 2. Independently, inspired by the work of Feige et al. [FGL+96], Goldreich [G11] obtained the same inapproximability factor for the VERTEX COVER problem in k-hypergraphs for some constant k. After that, Holmerin [H02b] showed that the problem is **NP**-hard to approximate within any approximation ratio $k^{1-o(1)}$. By exploiting the notion of covering complexity introduced by Guruswami, Håstad and Sudan [GHS02] together with some ideas from [DS05], Dinur, Guruswami and Khot [DGK02] achieved a hardness factor of $(k - 3 - \delta)$ for all $\delta > 0$. Dinur, Guruswami, Khot and Regev [DGKR05] proved that the VERTEX COVER problem in k-hypergraphs is **NP**-hard to approximate to within any constant approximation ratio less than $(k - 1)$. The best known approximation hardness result is due to Khot and Regev [KR08], who proved that it is UG-hard to approximate the problem to within any constant approximation ratio less than k.

Dense k-Hypergraphs

Karpinski and Zelikovsky [KZ97a] studied the approximability of dense instances of various covering problems and designed an approximation algorithm for the VERTEX COVER problem in average and everywhere dense graphs with approximation ratio strictly less than 2. Exploiting a different approach, Nagamochi and Ibaraki [NI99] gave an approximation algorithm for VERTEX COVER problem using cycle decompositions of graphs with approximation ratio less than 2 in dense graphs, but the approximation ratio is weaker as compared to the previously mentioned algorithm. Bar-Yehuda and Kehat [BK04] studied the VERTEX COVER problem in dense k-hypergraphs. Using a greedy approach, they gave an approximation algorithm for this problem with a better approximation ratio than k. To the author's knowledge, this is the only result tackling the dense version of the VERTEX COVER problem in k-hypergraphs. On the other hand, Clementi and Trevisan [CT99] proved that the VERTEX COVER problem in average and everywhere dense graphs is **APX**-complete. Eremeev [E99] gave the first explicit inapproximability result and proved that it is **NP**-hard to approximate the VERTEX COVER problem in ε-dense graphs within any constant approximation ratio better than $(7 + \varepsilon)/(6 + 2 \cdot \varepsilon)$. Assuming that there is no polynomial time approximation algorithm for the VERTEX COVER problem in graphs with approximation ratio better than 2, Bar-Yehuda and Kehat [BK04] constructed a reduction implying that the approximation ratios achieved by the algorithm due to Karpinski and Zelikovsky [KZ97a] are optimal.

For the VERTEX COVER problem in (ε, ℓ)-dense in k-hypergraphs, we design an efficient approximation algorithm with improved approximation ratio for all $k > 0$ and $\ell > 0$. Furthermore, we give an optimal inapproximability result for the problem. The approach designed for the (ε, ℓ)-dense case will also be applicable for the k-balanced case in Section 6.5.

Nearly Regular k-Hypergraphs

The VERTEX COVER problem in nearly regular graphs, which is a generalization of the class of dense graphs, was first studied by Imamura and Iwama [II05]. They gave a polynomial time approximation algorithm that achieves with high probability an approximation ratio strictly smaller than 2 whenever the graph is nearly regular and has average degree $\Omega\big(n \cdot \log\log n / \log n\big)$. Assuming that there is no polynomial time approximation algorithm for the VERTEX COVER problem in graphs with approximation ratio better than 2, they proved that the approximation ratio of their algorithm is best possible.

By reducing the sample size of the algorithm due to Imamura and Iwama [II05], Cardinal, Karpinski, Schmied and Viehmann [CKSV11] extended the range of the approximation algorithm achieving an approximation ratio strictly smaller than 2 in nearly regular graphs having an average degree $\Omega\big(n \cdot \log\log\log n / \log n\big)$.

For the VERTEX COVER problem in nearly regular k-hypergraphs, we give a randomized approximation algorithm with approximation ratio strictly less than k and running time depending on the density of the underlying k-hypergraph. In particular, it entails the existence of quasi-polynomial and polynomial time approximation algorithms with approximation ratio less than k for mildly sparse and subdense instances, respectively. Especially, in the case $k = 2$, our approximation algorithm achieves the same approximation ratio as in [CKSV11], but for a larger class of graphs.

On the other hand, we give tight approximation lower bounds for this problem. Moreover, we prove the best known approximation lower bounds for the VERTEX COVER problem in regular k-hypergraphs.

5.2 Outline of this Chapter

This chapter is organized as follows. In Section 5.4, we survey some of the known approximation algorithms and approximation hardness results for the VERTEX COVER problem in k-hypergraphs. In Section 5.5, we investigate the approximability of the VERTEX COVER problem in dense k-hypergraphs. In Section 5.6, we study the approximability of the VERTEX COVER problem in a more general class of k-hypergraphs called nearly regular k-hypergraphs.

5.3 Preliminaries

We are going to introduce the notation used in this chapter. Let us first define the underlying problem according to Definition 4.1.1.

Definition 5.3.1 (VERTEX COVER problem in k-hypergraphs)
Instances: k-*hypergraphs* $\mathcal{H}$
Solutions: *Subsets* $C \subseteq V(\mathcal{H})$ *such that* $C \cap e \neq \varnothing$ *for all* $e \in E(\mathcal{H})$
Task: *Minimize the cardinality of* C

For a given k-hypergraph $\mathcal{H}$ with $k \geq 2$ and a vertex $v \in V(\mathcal{H})$, we introduce the *v-induced hypergraph* $\mathcal{H}(v)$ defined by $V\big(\mathcal{H}(v)\big) = V(\mathcal{H})\backslash\{v\}$ and $E\big(\mathcal{H}(v)\big) = \big\{ e\backslash\{v\} \mid v \in e \in E(\mathcal{H})\big\}$. Let $\mathcal{H}$ be a k-hypergraph with n vertices and $u \in [n]$. We refer to $A \subset B \subseteq V(\mathcal{H})$ with $|A| = u$ as a *set of u-heaviest vertices* in B if the vertices in A obey the property

$$\min_{v \in A} \big\{d_{\mathcal{H}}(v)\big\} \geq \max_{v \in B\backslash A} \big\{d_{\mathcal{H}}(v)\big\}.$$

Given a k-hypergraph $\mathcal{H}$ with n vertices, we are going to introduce the parameter $\Psi_{\mathcal{H}}(n)$ defined as follows.

$$\Psi_{\mathcal{H}}(n) = \binom{n}{k}\big(|E(\mathcal{H})|\big)^{-1}$$

By means of the parameter Ψ, we define classes of k-hypergraphs. Given a k-hypergraph $\mathcal{H}$ with n vertices, we refer to $\mathcal{H}$ as *dense*, *subdense*, *mildly sparse* and *non-dense* if we have $\Psi_{\mathcal{H}}(n) = \mathrm{O}(1)$, $\Psi_{\mathcal{H}}(n) = \mathrm{O}\big(\log n\big)$, $\Psi_{\mathcal{H}}(n) = \mathrm{poly}(\log n)$ and $\Psi_{\mathcal{H}}(n) = \omega(1)$, respectively.

For dense k-hypergraphs, we introduce even finer-grained classes. Given a k-hypergraph $\mathcal{H}$ with n vertices, we say that $\mathcal{H}$ is (ε, ℓ)*-dense* with $(\ell+1) \in [k]$ and $\varepsilon \in (0, 1]$ whenever for every subset

$$S \in \binom{V(\mathcal{H})}{\ell}, \text{ we have } d_{\mathcal{H}}(S) \geq \varepsilon \cdot \binom{n-\ell}{k-\ell}.$$

Thus, for instance, an $(\varepsilon, 0)$-dense k-hypergraph containing n vertices is a k-hypergraph with at least $\varepsilon\binom{n}{k}$ edges, whereas an $(\varepsilon, 1)$-dense k-hypergraph has the property that every vertex is contained in at least $\varepsilon\binom{n-1}{k-1}$ edges.

This definition naturally generalizes the notion of *everywhere density* ($(\varepsilon, 1)$-density) and *average density* ($(\varepsilon, 0)$-density) in graphs [KZ97a]. For notational simplicity, we refer to an $(\varepsilon, 0)$-dense k-hypergraph as an ε-dense k-hypergraph.

5.4 The General Problem

Before we study the VERTEX COVER problem in dense and nearly regular k-hypergraphs, we survey some of the known approximation algorithms and approximation hardness results for the VERTEX COVER problem in k-hypergraphs.

5.4.1 Approximation Algorithms

There is a well-known and simple approximation algorithm for the VERTEX COVER problem in k-hypergraphs. It constructs a maximal matching $M \subseteq E(\mathcal{H})$ in the given k-hypergraph $\mathcal{H}$ and returns the union of the edges in M. The constructed set $V(M)$ is a vertex cover of $\mathcal{H}$ since the remaining vertices contained in $V(\mathcal{H}) \backslash V(M)$ form an independent set in $\mathcal{H}$. The simple approximation algorithm is defined in Figure 5.1

```
Algorithm $\mathcal{A}_{5.1}$

    Input  : A $k$-hypergraph $\mathcal{H}$.
    Output: A vertex cover $S$ of $\mathcal{H}$.

    ① $S \leftarrow \emptyset$;
    while ($E(\mathcal{H}) \neq \emptyset$) do
        ② Let $e \in E(\mathcal{H})$ be an arbitrary edge in $\mathcal{H}$;
        ③ $S \leftarrow S \cup e$;
        ④ $\mathcal{H} \leftarrow \mathcal{H}[V(\mathcal{H}) \backslash e]$;
    end
    return $S$;
```

Figure 5.1: Algorithm $\mathcal{A}_{5.1}$

We are going to prove that algorithm $\mathcal{A}_{5.1}$ returns a vertex cover of the given k-hypergraph with approximation ratio at most k.

Lemma 5.4.1 (Folklore)
Algorithm $\mathcal{A}_{5.1}$ is a polynomial time approximation algorithm for the VERTEX COVER *problem in k-hypergraphs with approximation ratio k.*

Proof of Lemma 5.4.1.
Let $\mathcal{H}$ be a k-hypergraph. Furthermore, let $S = e_1 \cup e_2 \cup \ldots \cup e_l$ be the set constructed by Algorithm $\mathcal{A}_{5.1}$ on input $\mathcal{H}$, where $e_i \in E(\mathcal{H})$ denotes the edge that is taken by the algorithm in the i-th iteration. Let C be an minimum vertex cover of $\mathcal{H}$. By definition of C, we have $|C \cap e_i| \geq 1$ for every $i \in [l]$. Since the intersection of e_i with the vertex set of the k-hypergraph $\mathcal{H}[V(\mathcal{H}) \backslash e_i]$ is empty, we obtain $e_i \cap e_j = \varnothing$ for every distinct $j, i \in [l]$. Let us bound the cardinality of S from above. We see that

$$\begin{aligned} |S| &= \sum_{i \in [l]} |e_i| \leq \sum_{i \in [l]} |e_i \cap C||e_i| && (\text{by } |C \cap e_i| \geq 1) \\ &= \sum_{i \in [l]} |e_i \cap C| k && (\text{by } |e_i| = k) \\ &\leq \sum_{v \in C} k && (\text{by } e_i \cap e_i = \varnothing \text{ for all } i \neq j) \\ &\leq k \cdot |C|. \end{aligned}$$

It remains to be proved that S is indeed a vertex cover of $\mathcal{H}$. Suppose for the sake of contradiction that there is a edge e' in $\mathcal{H}$ with $e' \cap S = \varnothing$. Then, we get $e' \in E(\mathcal{H}[V(\mathcal{H}) \backslash S])$. Thus, by definition of the set S, e' must be contained in S contradicting our assumption. ■

The currently best known approximation algorithm for the VERTEX COVER problem in k-hypergraphs achieves an approximation ratio at most $k\big(1 - o(1)\big)$ and is due to Halperin [H02]. He designed an algorithm that uses the semidefinite programming relaxation combined with a randomized rounding procedure. Let us give the precise statement.

Theorem 5.4.1 ([H02])
Let $\mathcal{H}$ be a k-hypergraph with $V(\mathcal{H}) = n$ and $k \geq 2$. There is an efficient

randomized approximation algorithm for the VERTEX COVER *problem in k-hypergraphs with approximation ratio*

$$k - [1 - o(1)]k \cdot \frac{\ln \ln n}{\ln n}.$$

5.4.2 Approximation Hardness Results

As for approximation lower bounds, we present two approximation hardness results relying on different assumptions. The first result is due to Khot and Regev [KR08] and concerns the UG-hardness of approximation of the VERTEX COVER problem in k-hypergraphs.

Theorem 5.4.2 ([KR08])
Let $\mathcal{H}$ be a k-hypergraph with $k \geq 2$. For every $\delta > 0$, the following cases (i) and (ii) are UG*-hard to decide.*

(i) *Every vertex cover of $\mathcal{H}$ has size at least $|V(\mathcal{H})|\,(1-\delta)$.*

(ii) *The cardinality of a minimum vertex cover is at most*

$$|V(\mathcal{H})|\left(\frac{1}{k} + \delta\right).$$

In particular, this result implies that it is UG-hard to achieve a constant approximation ratio better than k for the VERTEX COVER problem in k-hypergraphs.

Corollary 5.4.1
For every $k \geq 2$, it is UG*-hard to approximate the* VERTEX COVER *problem in k-hypergraphs to within any constant approximation ratio less than k.*

Proof of Corollary 5.4.1.
Let $\mathcal{H}$ be a k-hypergraph with $k \geq 2$ and C an optimal vertex cover of $\mathcal{H}$. According to Theorem 5.4.2, it is UG-hard to tell

(i) whether $|V(\mathcal{H})|\,(1-\delta) \;\leq\; |C|$

(ii) or $|C| \;\leq\; |V(\mathcal{H})|\left(\frac{1}{k} + \delta\right)$ holds.

By approximating the VERTEX COVER problem in k-hypergraphs with an approximation ratio

$$k - \delta' \quad < \quad \frac{(1-\delta)}{\left(\frac{1}{k}+\delta\right)} \quad = \quad \frac{|V(\mathcal{H})|\,(1-\delta)}{|V(\mathcal{H})|\left(\frac{1}{k}+\delta\right)},$$

we are able to decide, which one of the cases (i) and (ii) applies. ■

On the other hand, Dinur, Guruswami, Khot and Regev [DGKR05] proved the following **NP**-hardness of approximation result. In particular, it implies that it is **NP**-hard to approximate the VERTEX COVER problem to within any constant approximation ratio less than $(k-1)$.

Theorem 5.4.3 ([DGKR05])
*Let $\mathcal{H}$ be a k-hypergraph with $k \geq 3$. For every $\delta > 0$, the following cases (i) and (ii) are **NP**-hard to decide.*

(i) *Every vertex cover of $\mathcal{H}$ has size at least $|V(\mathcal{H})|\,(1-\delta)$.*

(ii) *The size of a minimum vertex cover is at most*

$$|V(\mathcal{H})|\left(\frac{1}{k-1}+\delta\right).$$

Analogously, it yields the following inapproximability result for the VERTEX COVER problem in k-hypergraphs.

Corollary 5.4.2
*For every $k \geq 3$, it is **NP**-hard to approximate the* VERTEX COVER *problem in k-hypergraphs within any constant approximation ratio better than $(k-1)$.*

5.5 Dense k-Hypergraphs

As an important restriction of the general problem, we study the approximability of the VERTEX COVER problem in dense k-hypergraphs. We design an improved approximation algorithm and give tight approximation lower bounds. Before we present our results, we are going to review some previously known results for this restricted version of the problem.

5.5.1 Previously Known Results

The VERTEX COVER problem in dense graphs was studied by Karpinski an Zelikovsky [KZ97b]. They designed an approximation algorithm for this problem with an approximation ratio strictly less than 2 in dense and average dense graphs. In particular, they proved the following theorem.

Theorem 5.5.1 ([KZ97b])
The VERTEX COVER *problem can be approximated in polynomial time with an approximation ratio*

- $\frac{2}{1+\varepsilon}$ *in ε-everywhere dense graphs and*
- $\frac{2}{2-\sqrt{1-\varepsilon}}$ *in ε-average dense graphs.*

The approximation algorithm due to Karpinski and Zelikovsky [KZ97b] achieves the best known approximation ratio for the VERTEX COVER problem in average and everywhere dense graphs. The basic idea of their algorithm is to extract a large enough subset W of a minimum vertex cover of the underlying graph $\mathcal{G}$ by exhaustive search in the first phase. In the second phase, the algorithm computes a 2-approximate solution Y of the VERTEX COVER problem in the remaining graph defined by $\mathcal{G}' = \mathcal{G}[V(\mathcal{G})\backslash W]$. Then, the set $Y \cup W$ yields a vertex cover of the original graph $\mathcal{G}$.

Independently, Nagamochi and Ibaraki [NI99] gave an approximation algorithm for the VERTEX COVER problem with approximation ratio that is parametrized by the average density of the given graph. Their algorithm uses cycle decompositions of graphs leading to an approximation ratio below 2 if the given graph is dense. More precisely, they obtained the following result.

Theorem 5.5.2 ([NI99])
Given a graph $\mathcal{G}$ with n vertices and m edges, there is an approximation algorithm for the VERTEX COVER *problem in graphs that constructs a vertex cover for $\mathcal{G}$ with approximation ratio at most*

$$2 - \frac{8m}{13n^2 + 8m}$$

in $\mathrm{O}(n \cdot m)$ *time.*

Bar-Yehuda and Kehat [BK04] studied the VERTEX COVER problem in dense k-hypergraphs and proved that the problem is approximable with an approximation ratio strictly less than k.

Theorem 5.5.3 ([BK04])
There is a polynomial time approximation algorithm for the VERTEX COVER *problem with approximation ratio*

$$\frac{k}{k-(k-1)(1-\varepsilon)^{\frac{1}{k}}}$$

in ε-dense k-hypergraphs.

Trevisan and Clementi [CT99] investigated the approximation hardness of the VERTEX COVER problem in average and everywhere dense graphs and proved that both versions of the problem are **APX**-complete.

Theorem 5.5.4 ([CT99])
The VERTEX COVER *problem restricted to dense graphs is* **APX**-*complete. In particular, for any $\varepsilon > 0$, there exists a constant $\alpha > 1$ (depending on ε) such that it is* **NP**-*hard to approximate the* VERTEX COVER *problem restricted to graphs in which any node has degree at least $\varepsilon \cdot |V(\mathcal{G})|$ within any constant approximation ratio less than α.*

The idea of the reduction constructed in [CT99] is to join an **APX**-hard instance $\mathcal{G}$ of the VERTEX COVER problem in graphs with a clique of size $\varepsilon \cdot |V(\mathcal{G})|(1-\varepsilon)^{-1}$ together with all edges between $\mathcal{G}$ and the clique yielding an ε-everywhere dense graph $\mathcal{G}'$. The resulting graph is an **APX**-hard instance of the VERTEX COVER problem in dense graphs.

A similar construction was used by Eremeev [E99], who obtained an explicit approximation lower bound parametrized by the density of the resulting graph. The hard instance of the VERTEX COVER problem in graphs was given by Håstad [H01].

Theorem 5.5.5 ([E99])
It is **NP**-*hard to approximate the* VERTEX COVER *problem in ε-everywhere*

graphs to within any constant approximation ratio less than

$$\frac{7+\varepsilon}{6+2\cdot\varepsilon}.$$

Assuming that there is no polynomial time approximation algorithm for the VERTEX COVER problem in graphs with approximation ratio below 2, Bar-Yehuda and Kehat established the following hardness of approximation result.

Theorem 5.5.6 ([BK04])
There is no polynomial time algorithm with an approximation ratio less than

$$\frac{2}{1-\varepsilon} \qquad \textit{and} \qquad \frac{2}{1+\sqrt{1-\varepsilon}}$$

in ε-everywhere dense graphs and in ε-average dense graphs, respectively, unless there is a polynomial time approximation algorithm for the VERTEX COVER *problem in general graphs with an approximation ratio better than* 2.

In particular, it implies that under the assumption in Theorem 5.5.6, the approximation upper bounds achieved in Theorem 5.5.1 are best possible.

5.5.2 Our Contributions

For the VERTEX COVER problem in (ε, ℓ)-dense k-hypergraphs, we design an approximation algorithm with an improved approximation ratio for all $k > 2$ and $\ell > 0$. More precisely, we prove the following theorem.

Theorem 5.5.7
There is a polynomial time approximation algorithm for the VERTEX COVER *problem in (ε, ℓ)-dense k-hypergraphs with approximation ratio at most*

$$\frac{k}{k-(k-1)(1-\varepsilon)^{\frac{1}{k-\ell}}} - o(1).$$

Our algorithm matches the approximation ratio of the algorithms due to Bar-Yehuda and Kehat [BK04] for $\ell = 0$ and due to Karpinski and Zelikovsky [KZ97b] for $k = 2$ and $\ell \in \{0, 1\}$, but achieves a better result for all

$\ell > 0$ and $k > 2$. As a byproduct of our approach, we obtain three additional results. Firstly, we give a constructive proof for a lower bound on the size of a minimum vertex cover of (ε, ℓ)-dense k-hypergraphs. More precisely, we design an efficient algorithm that extracts a large subset of a minimum vertex cover of a given (ε, ℓ)-dense k-hypergraph.

Lemma 5.5.1 (Extraction Lemma)
Given an (ε, ℓ)-dense k-hypergraph $\mathcal{H}$, there is an algorithm that constructs on input $\mathcal{H}$ in polynomial time a set $\widetilde{W} = \{W_i \subseteq V(\mathcal{H}) \mid i \in [s]\}$ of size $s = \mathrm{O}(n^k)$ with the following properties.

(*i*) *There exists $i \in [s]$ such that W_i is a subset of a minimum vertex cover of $\mathcal{H}$.*

(*ii*) *For all $i \in [s]$, we have $|W_i| \geq \left[1 - (1-\varepsilon)^{\frac{1}{k-\ell}}\right][n - k + 1]$.*

Consequently, we obtain the following result.

Corollary 5.5.1
Given a (ε, ℓ)-dense k-hypergraph $\mathcal{H}$ containing n vertices, the size of a minimum vertex cover of $\mathcal{H}$ is bounded from below by $\left[1 - (1-\varepsilon)^{\frac{1}{k-\ell}}\right][n - k + 1]$.

Secondly, in section 5.6.3, we design a randomized version of the extraction algorithm with decreased number of candidate sets. It will play a crucial role in our improved approximation algorithm for the VERTEX COVER problem in nearly regular k-hypergraphs.

Thirdly, in section 6.5.2, we will see that this approach can also be successfully applied to the k-balanced case. In particular, we give an approximation algorithm for the VERTEX COVER problem in dense k-balanced hypergraphs with approximation ratio strictly less than $k/2$.

On the other hand, we obtain a tight approximation lower bound with respect to Theorem 5.5.7.

Theorem 5.5.8
For every $k \geq 2$, it is UG*-hard to approximate the* VERTEX COVER *problem in (ε, ℓ)-dense k-hypergraphs to within any constant approximation ratio less*

than

$$\frac{k}{k-(k-1)(1-\varepsilon)^{\frac{1}{k-\ell}}}.$$

Furthermore, we prove a **NP**-hardness of approximation result stated below.

Theorem 5.5.9
For every $k \geq 3$, it is **NP***-hard to approximate the* VERTEX COVER *problem in (ε, ℓ)-dense k-hypergraphs to within any constant approximation ratio less than*

$$\frac{k-1}{k-1-(k-2)(1-\varepsilon)^{\frac{1}{k-\ell}}}.$$

5.5.3 The Generalized Approximation Algorithm

We are going to define our generalized approximation algorithm with improved approximation ratio for the VERTEX COVER problem in (ε, ℓ)-dense k-hypergraphs. More precisely, we are going to prove Theorem 5.5.7 restated below.

Theorem 5.5.7
There is a polynomial time approximation algorithm for the VERTEX COVER *problem in (ε, ℓ)-dense k-hypergraphs with approximation ratio at most*

$$\frac{k}{k-(k-1)\left(1-\varepsilon\right)^{\frac{1}{k-\ell}}} - o(1).$$

In order to prove Theorem 5.5.7, we first design an approximation algorithm that on input of a k-hypergraph $\mathcal{H}$ and a subset W of an minimum vertex cover computes a vertex cover of $\mathcal{H}$ with approximation ratio depending on the size of W. Afterwards, we prove that we can extract a sufficiently large subset of an optimal vertex cover of a given dense k-hypergraph. In particular, we will give a proof of the Extraction Lemma (Lemma 5.5.1).

Approximating the Remaining Instance

Assuming we have given a k-hypergraph and a large subset W of a minimum vertex cover C of $\mathcal{H}$, the algorithm $\mathcal{A}_{5.2}$ defined in Figure 5.2 constructs a vertex cover of $\mathcal{H}$ with approximation ratio parametrized by the cardinality of $W \cap C$.

```
Algorithm $\mathcal{A}_{5.2}$
  Input  : $(\mathcal{H}, W, j)$, where $\mathcal{H}$ is a $k$-hypergraph, $W$ a subset of
           $V(\mathcal{H})$ and $j \geq k$ an integer.
  Output: A vertex cover $S$ of $\mathcal{H}$.
  begin
    ① $\mathcal{H}' \leftarrow \mathcal{H}[V(\mathcal{H}) \backslash W]$;
    ② $S' \leftarrow \mathcal{A}_{5.1}(\mathcal{H}')$;
    ③ $S \leftarrow W \cup S'$;
    foreach $Y \in \left\{ D \in \binom{V(\mathcal{H})}{n-i} \,\middle|\, i \in \{k-1, \ldots, j\} \right\}$ do
      if ($Y$ is a vertex cover of $\mathcal{H}$ and $|Y| < |S|$) then
        ④ $S \leftarrow Y$;
      end
    end
    return $S$;
  end
```

Figure 5.2: Algorithm $\mathcal{A}_{5.2}$

We are going to prove the following Lemma.

Lemma 5.5.2

Let $\mathcal{H}$ be a k-hypergraph containing n vertices, C a minimum vertex cover of $\mathcal{H}$ and $W \subseteq V(\mathcal{H})$ with $|W \cap C| \geq \delta \cdot |W|$ for a fixed constant $\delta \in (k^{-1}, 1]$. For every integer $j \geq k$, the algorithm $\mathcal{A}_{5.2}$ computes on input $(\mathcal{H}, W, j)$ in *polynomial time a vertex cover of $\mathcal{H}$ with size at most*

$$\max\left\{ |C|\,,\ \frac{k}{1+(\delta k-1)\dfrac{|W|}{n-j}}\cdot |C| \right\}.$$

Proof of Lemma 5.5.2.
Given a k-hypergraph $\mathcal{H}$ containing n vertices and a set W such that there is a minimum vertex cover C of $\mathcal{H}$ with the property

$$|W \cap C| \ \geq \ \delta \cdot |W| \text{ for a fixed constant } \delta \in \left(\frac{1}{k}\,,1\right], \tag{5.1}$$

the algorithm $\mathcal{A}_{5.1}$ computes a vertex cover Y of the k-hypergraph $\mathcal{H}'$ induced by the edges that are not covered by W. Let us denote by C' a minimum vertex cover of $\mathcal{H}'$. For every integer $j \geq k$, we introduce the set $V(j)$ given by

$$V(j) = \left\{ S \subseteq V(\mathcal{H}) \;\middle|\; S \in \binom{V(\mathcal{H})}{n-i} \text{ and } i \in \{k-1, ..., j\} \right\}. \tag{5.2}$$

Clearly, for every fixed integer $j \geq k$, we can find in polynomial time the smallest set X^j in $V(j)$ being a vertex cover of $\mathcal{H}$. Since the solution S, returned by the algorithm, is the smallest set among X^j and $Y \cup W$, we may assume that $|S| \leq n-j$ as otherwise, S is an optimal solution. We are going to deduce an upper bound on the size of S.

$$\begin{aligned}
|S| &= |W| + |Y| \\
&\leq |W| + k\cdot|C'| && \text{(by Theorem 5.4.1)} \\
&\leq |W| + k\cdot(|C| - |W \cap C|) \\
&\leq |W| + k\cdot(|C| - \delta|W|) && \text{(by (5.1))} \\
&\leq k|C| - (\delta k - 1)\frac{|W|}{n-j}\,|S| && \left(\text{by } \frac{|S|}{n-j} \leq 1 \text{ and } \delta \in \left(\frac{1}{k}\,,1\right]\right)
\end{aligned}$$

All in all, we obtain the following upper bound on the size of S

$$|S| \ \leq \ \frac{k}{1+(\delta k-1)\dfrac{|W|}{n-j}}\cdot|C|$$

completing the proof of Lemma 5.5.2. ■

A Deterministic Extraction Algorithm

In order to apply algorithm $\mathcal{A}_{5.2}$ successfully, we need to find a large subset W of a minimum vertex cover of a given k-hypergraph $\mathcal{H}$. The recursive algorithm $\mathcal{A}_{5.3}$ defined in Figure 5.3 constructs efficiently on input of a dense k-hypergraph $\mathcal{H}$ a collection $\widetilde{W}$ of subsets $W_i \subseteq V(\mathcal{H})$ such that at least one of them is contained in a minimum vertex cover of $\mathcal{H}$.

```
Algorithm $\mathcal{A}_{5.3}$
    Input  : $(\mathcal{H}, \varepsilon)$, where $\mathcal{H}$ is an $\varepsilon$-dense $k$-hypergraph.
    Output: A collection $\widetilde{W}$ of subsets $W_i \subseteq V(\mathcal{H})$.

    begin
        ① $\widetilde{W} \leftarrow \varnothing$;
        if $k = 1$ then
            ② $\widetilde{W} \leftarrow \{V(E(\mathcal{H}))\}$;
            return $\widetilde{W}$;
        else
            ③ Find a set $B$ consisting of some $\left(1 - (1-\varepsilon)^{\frac{k}{(k+1)}}\right)$-heaviest
            vertices in $V(\mathcal{H})$;
            ④ Add $B$ to $\widetilde{W}$;
            foreach $v \in B$ do
                ⑤ $\mathcal{H}' \leftarrow \mathcal{H}(v)$;
                ⑥ $\varepsilon' \leftarrow \left(1 - (1-\varepsilon)^{\frac{k}{(k+1)}}\right)$;
                ⑦ $W' \leftarrow \mathcal{A}_{5.3}(\mathcal{H}', \varepsilon')$;
                ⑧ $\widetilde{W} \leftarrow \widetilde{W} \cup W'$;
            end
        end
        return $\widetilde{W}$;
    end
```

Figure 5.3: Algorithm $\mathcal{A}_{5.3}$

We are going to prove the following lemma that enables us to extract efficiently a large part of a minimum vertex cover of a given dense k-hypergraph.

Lemma 5.5.3

Given an ε-dense k-hypergraph $\mathcal{H}$ containing n vertices, the algorithm $\mathcal{A}_{5.3}$ returns on input $(\mathcal{H}, \varepsilon)$ in polynomial time a set $\widetilde{W} = \{W_i \subseteq V(\mathcal{H}) \mid i \in [s]\}$ of size $s = \mathrm{O}(n^k)$ with the following properties.

(i) For all vertex covers C of $\mathcal{H}$, there exists a $j \in [s]$ such that W_j is a subset of C.

(ii) For all $i \in [s]$, we have $|W_i| \geq \left[1 - (1-\varepsilon)^{\frac{1}{k}}\right](n-k+1)$.

In order to give the proof of Lemma 5.5.3, we provide the following simple statement.

Lemma 5.5.4

Let $\mathcal{H}$ be an ε-dense k-hypergraph containing n vertices and $B \subseteq V(\mathcal{H})$ a set of $\left[\left(1-(1-\varepsilon)^{\frac{1}{k}}\right)n\right]$-heaviest vertices in $V(\mathcal{H})$. Then, for all $v \in B$, the degree of v can be bounded from below as follows.

$$d_{\mathcal{H}}(v) \;\geq\; \left(1-(1-\varepsilon)^{\frac{k-1}{k}}\right)\binom{n-1}{k-1}$$

Proof of Lemma 5.5.4.

Let us consider a k-hypergraph $\mathcal{H}$ containing n vertices and m edges, where

$$m \;\geq\; \varepsilon \cdot \binom{n}{k}. \tag{5.3}$$

We denote by $B \subseteq V(\mathcal{H})$ a set of $[(1-(1-\varepsilon)^{1/k})n]$-heaviest vertices in $V(\mathcal{H})$. Suppose the statement is not true. Then, the number m of edges in $\mathcal{H}$ is strictly smaller than the number of edges in a k-hypergraph in which all vertices of B have the maximum possible degree, whereas all the remaining vertices have degree exactly

$$\left(1-(1-\varepsilon)^{\frac{k-1}{k}}\right)\binom{n-1}{k-1}.$$

Therefore, we deduce the following bound on the number of edges in the k-hypergraph $\mathcal{H}$.

$$\begin{aligned} m &< \frac{1}{k}\left(|B|\cdot\binom{n-1}{k-1} + (n-|B|)\left[(1-(1-\varepsilon)^{\frac{k-1}{k}}\right]\binom{n-1}{k-1}\right) \\ &= \frac{1}{k}\left(\left[1-(1-\varepsilon)^{\frac{1}{k}}\right] n\cdot\binom{n-1}{k-1} \quad + \right. \\ &\qquad \left. \left[n-\left[1-(1-\varepsilon)^{\frac{1}{k}}\right]\cdot n\right]\left[1-(1-\varepsilon)^{\frac{k-1}{k}}\right]\binom{n-1}{k-1}\right) \\ &= \left(1-(1-\varepsilon)^{\frac{1}{k}}\right)\binom{n}{k} + (1-\varepsilon)^{\frac{1}{k}}\left(1-(1-\varepsilon)^{\frac{k-1}{k}}\right)\binom{n}{k} \\ &= \varepsilon\cdot\binom{n}{k} \end{aligned}$$

It contradicts our assumption (5.3), since $\mathcal{H}$ is ε-dense. ■

Let us proceed to give the proof of Lemma 5.5.3.

Proof of Lemma 5.5.3.

Let $\mathcal{H}$ be an ε-dense k-hypergraph containing n vertices and B be a set of $[(1-(1-\varepsilon)^{1/k})\,n]$-heaviest vertices of $\mathcal{H}$. On input $\mathcal{H}$, algorithm $\mathcal{A}_{5.3}$ returns a set $\widetilde{W}$ of size $\mathrm{O}(n^k)$ in time $\mathrm{O}(n^k)$, which is polynomial since we assumed $k=\mathrm{O}(1)$.

(i) Let C be a vertex cover of $\mathcal{H}$. We claim that there is a $W_j \in \widetilde{W}$ with $W_j \subseteq C$, which will be verified by induction. If all vertices in B belong to C, we are done.

Otherwise, there exists $v \in B$ that does not belong to C. But then, C must contain a vertex cover of the v-induced $(k-1)$-hypergraph $\mathcal{H}(v)$ as otherwise, some edges will not be covered. By induction, the recursive call returns one subset contained in C. The base case $(k=1)$ is trivial.

(ii) We prove the second property by induction as well. Suppose that we have $|W_i| \geq \left[(1-(1-\varepsilon)^{\frac{1}{k}}\right](n-k+1)$ for all $i \in [s]$ and for all k-hypergraphs with some fixed value of k. We now prove the property for $k+1$. According to Lemma 5.5.4, the recursive calls are performed on ε'-dense k-hypergraphs with $n-1$ vertices. By induction hypothesis the recursive call returns a collection of sets W_i of size

$$\begin{aligned} |W_i| &\geq \left[1-(1-\varepsilon')^{\frac{1}{k}}\right][(n-1)-k+1] \\ &= \left[1-\left(1-\left[1-(1-\varepsilon)^{\frac{k}{k+1}}\right]\right)^{\frac{1}{k}}\right][n-(k+1)+1] \\ &= \left[1-(1-\varepsilon)^{\frac{1}{k+1}}\right][n-(k+1)+1], \end{aligned}$$

as claimed. The base case $k = 1$ is verified, as in that case, the algorithm returns at least εn vertices. ■

We are going to tackle the case $\ell > 0$. Therefore, let us consider algorithm $\mathcal{A}_{5.4}$ defined in Figure 5.4. On input of an (ε, ℓ)-dense k-hypergraph $\mathcal{H}$ with n vertices, it constructs a collection of subsets of $V(\mathcal{H})$ such that one of them is contained in a minimum vertex cover of $\mathcal{H}$ and has size at least $\left[1-(1-\varepsilon)^{\frac{1}{k-\ell}}\right][n-k+1]$.

```
Algorithm $\mathcal{A}_{5.4}$

  Input  : $(\mathcal{H}, \varepsilon, \ell)$, where $\mathcal{H}$ is an $(\varepsilon, \ell)$-dense $k$-hypergraph.
  Output: A collection $\widetilde{W}$ of subsets $W_i \subseteq V(\mathcal{H})$.

  begin
      ① $\widetilde{W} \leftarrow \varnothing$;
      foreach $S \in \binom{V(\mathcal{H})}{\ell}$ do
          ② $\mathcal{H}' \leftarrow (V(\mathcal{H}) \backslash S, E(\mathcal{H}'))$ with
          $E(\mathcal{H}') = \{e \backslash S \mid S \subseteq e \in E(\mathcal{H})\}$;
          ③ $\widetilde{W}' \leftarrow \mathcal{A}_{5.3}(\mathcal{H}', \varepsilon)$;
          ④ $\widetilde{W} \leftarrow \widetilde{W} \cup \widetilde{W}'$;
      end
      return $\widetilde{W}$;
  end
```

Figure 5.4: Algorithm $\mathcal{A}_{5.4}$

Finally, we are going to prove the Extraction Lemma.

Lemma 5.5.1 (Extraction Lemma)
Given an (ε, ℓ)-dense k-hypergraph $\mathcal{H}$, there is an algorithm that constructs on input $\mathcal{H}$ in polynomial time a set $\widetilde{W} = \{W_i \subseteq V(\mathcal{H}) \mid i \in [s]\}$ of size $s = \mathrm{O}(n^k)$ with the following properties.

(i) There exists $i \in [s]$ such that W_i is a subset of a minimum vertex cover of $\mathcal{H}$.

(ii) For all $i \in [s]$, we have $|W_i| \geq \left[1 - (1-\varepsilon)^{\frac{1}{k-\ell}}\right][n-k+1]$.

Proof of Lemma 5.5.1.
Let $\mathcal{H}$ be an (ε, ℓ)-dense k-hypergraph and C a minimum vertex cover of $\mathcal{H}$. Consider a subset S of ℓ vertices with $S \cap C = \varnothing$. Then, we define the $(k-\ell)$-hypergraph $\mathcal{H}'$ by the vertex set $V(\mathcal{H}') = V(\mathcal{H})\backslash S$ and edge set $E(\mathcal{H}') = \left\{e \cap (V(\mathcal{H})\backslash S) \mid e \in E(\mathcal{H}), S \subseteq e\right\}$. Since S has an empty intersection with C, the set $C \cap V(\mathcal{H}')$ is a vertex cover of $\mathcal{H}'$. From the definition of (ε, ℓ)-density, the number of edges in $\mathcal{H}'$ is bounded from below by

$$|E(\mathcal{H}')| = d_{\mathcal{H}}(S) \geq \varepsilon \cdot \binom{n-\ell}{k-\ell}.$$

Hence, $\mathcal{H}'$ is an ε-dense $(k-\ell)$-hypergraph containing $n-\ell$ vertices. According to Lemma 5.5.3, algorithm $\mathcal{A}_{5.3}$ extracts $\mathrm{O}(n^{k-\ell})$ candidates W_i, which are subsets of $V(\mathcal{H}')$ of size at least

$$|W_i| \geq \left[1 - (1-\varepsilon)^{\frac{1}{k-\ell}}\right]\left[(n-\ell+1) - (k-\ell)\right].$$

Note that at least one of them is contained in the vertex cover $C \cap V(\mathcal{H}')$ and therefore, in a minimum vertex cover of $\mathcal{H}$. By enumerating all $\mathrm{O}(n^\ell)$ possibilities for S, we get the result in $\mathrm{O}(n^k)$ time. ■

Thus far, we are ready to give the proof of Theorem 5.5.7.

Proof of Theorem 5.5.7

Let us consider algorithm $\mathcal{A}_{5.5}$ defined in Figure 5.5. Let $\mathcal{H}$ be an (ε, ℓ)-dense k-hypergraph containing n vertices and C a minimum vertex cover of

$\mathcal{H}$. Then, by Lemma 5.5.1, algorithm $\mathcal{A}_{5.4}$ extracts efficiently a collection $\widetilde{C}$ of sets $C_i \subseteq V(\mathcal{H})$ such that there is a $C_p \in \widetilde{C}$ with

$$|C_p \cap C| \;=\; |C_p| \;\geq\; \left(1 - [1-\varepsilon]^{\frac{1}{k-\ell}}\right)(n-k+1)\,. \tag{5.4}$$

Furthermore, we know that $|\widetilde{C}| = \mathrm{O}(n^k)$.

```
Algorithm $\mathcal{A}_{5.5}$

    Input  : $(\mathcal{H}, \varepsilon, \ell, j)$, where $\mathcal{H}$ is an $(\varepsilon, \ell)$-dense $k$-hypergraph and
             $j \geq k$ an integer.
    Output: A vertex cover $S$ of $\mathcal{H}$.

    begin
        ① $\widetilde{W} \leftarrow \emptyset$;
        ② $\widetilde{C} \leftarrow \mathcal{A}_{5.4}(\mathcal{H}, \varepsilon, \ell)$;
        foreach  $S \in \widetilde{C}$ do
            ③ $Y \leftarrow \mathcal{A}_{5.2}(\mathcal{H}, S, j)$;
            ④ $\widetilde{W} \leftarrow \widetilde{W} \cup \{Y\}$;
        end
        return the smallest set $S$ among all sets in $\widetilde{W}$;
    end
```

Figure 5.5: Algorithm $\mathcal{A}_{5.5}$

Let S_p be the vertex cover of $\mathcal{H}$ that was constructed by algorithm $\mathcal{A}_{5.2}$ on input $(\mathcal{H}, C_p, j)$ with $j \geq k$. By testing all possible sets $W_i \in \widetilde{W}$ and choosing the smallest vertex cover of $\mathcal{H}$, denoted by S, we have that $|S| \leq |S_p|$. According to Lemma 5.5.2, we obtain a polynomial time approximation algorithm with approximation ratio at most

$$\begin{aligned}
\frac{|S|}{|C|} \;\leq\; \frac{|S_p|}{|C|} \;&\leq\; \frac{k}{1 + (k-1)\dfrac{|W_p|}{n-j}} \\
&\leq\; \frac{k}{1 + (k-1)\dfrac{\left[1 - (1-\varepsilon)^{\frac{1}{k-\ell}}\right](n-k+1)}{n-j}} && \text{(by 5.4)}
\end{aligned}$$

For every $j \geq k$, the approximation ratio of algorithm $\mathcal{A}_{5.5}$ becomes

$$\frac{k}{k-(k-1)(1-\varepsilon)^{\frac{1}{k-\ell}}} - o(1)$$

and the proof follows. ■

5.5.4 Tight Approximation Hardness Results

As for approximation lower bounds, we give an UG-hardness and a **NP**-hardness of approximation result for the VERTEX COVER problem in dense k-hypergraphs. Especially, the former approximation lower bound is tight with respect to the approximation ratio achieved by algorithm $\mathcal{A}_{5.5}$.

An Optimal Inapproximability Result

We are going to prove the following theorem, which implies that it is UG-hard to achieve a better approximation ratio than achieved by the approximation algorithm given in the previous section.

Theorem 5.5.8
For every $k \geq 2$, it is UG-hard to approximate the VERTEX COVER problem in (ε, ℓ)-dense k-hypergraphs to within any constant approximation ratio less than

$$\frac{k}{k-(k-1)(1-\varepsilon)^{\frac{1}{k-\ell}}}.$$

Proof of Theorem 5.5.8.
As starting point of the reduction, we consider the UG-hard instance $\mathcal{H}$ of the VERTEX COVER problem in k-hypergraphs from Theorem 5.4.2. We are going to densify $\mathcal{H}$ in order to obtain the (ε, ℓ)-dense k-hypergraph $\mathcal{H}'$. Recall that according to Theorem 5.4.2, the following is UG-hard to decide for every $\delta > 0$.

(*i*) Every vertex cover of $\mathcal{H}$ has size at least $|V(\mathcal{H})|(1-\delta)$.

(*ii*) The cardinality of a minimum vertex cover of $\mathcal{H}$ is at most

$$|V(\mathcal{H})|\left(\frac{1}{k}+\delta\right).$$

Let us fix a constant $\varepsilon \in (0,1)$ and $(\ell+1) \in [k]$. Furthermore, we introduce ε' defined by

$$\varepsilon' = \varepsilon + \frac{k(k+1)}{|V(\mathcal{H})|}.$$

For notational simplicity, we will use $n = |V(\mathcal{H})|$. Then, we join the clique

$$\mathcal{K} = \left(V(\mathcal{K}), \binom{V(\mathcal{K})}{k} \right) \text{ of size } \frac{1-(1-\varepsilon')^{\frac{1}{k-\ell}}}{(1-\varepsilon')^{\frac{1}{k-\ell}}} \cdot n \text{ to } \mathcal{H}.$$

In addition, we add all edges e of size k to $\mathcal{H}'$ having a non-empty intersection with $\mathcal{K}$. Thus, we obtain

$$E(\mathcal{H}') = E(\mathcal{H}) \cup \left\{ e \in \binom{V(\mathcal{K}) \cup V(\mathcal{H})}{k} \,\middle|\, e \cap V(\mathcal{K}) \neq \varnothing \right\}.$$

Since we need to cover all the edges of the clique $\mathcal{K}$, the cases (i) and (ii) from Theorem 5.4.2 being UG-hard to decide transform into the following.

(iii) Every vertex cover of $\mathcal{H}'$ has size at least

$$n\,(1-\delta) + \frac{[1-(1-\varepsilon')^{\frac{1}{k-\ell}}]n}{(1-\varepsilon')^{\frac{1}{k-\ell}}}.$$

(iv) The cardinality of a minimum vertex cover is at most

$$n\left(\frac{1}{k}+\delta\right) + \frac{[1-(1-\varepsilon')^{\frac{1}{k-\ell}}]n}{(1-\varepsilon')^{\frac{1}{k-\ell}}}.$$

It implies that for every $\delta > 0$, it is UG-hard to approximate the VERTEX COVER problem in (ε,ℓ)-dense hypergraphs to within any constant approximation ratio less than $R(\delta)$, where $R(\delta)$ is defined as follows.

$$R(\delta) \;=\; \frac{n(1-\delta) + \dfrac{[1-(1-\varepsilon')^{\frac{1}{k-\ell}}]n}{(1-\varepsilon')^{\frac{1}{k-\ell}}}}{n\left(\dfrac{1}{k}+\delta\right) + \dfrac{[1-(1-\varepsilon')^{\frac{1}{k-\ell}}]n}{(1-\varepsilon')^{\frac{1}{k-\ell}}}}$$

Before we prove that this construction yields an (ε,ℓ)-dense k-hypergraph, we first obtain a simpler term for $R(\delta)$.

Let us deduce a lower bound on the inapproximability factor $R(\delta)$.

$$\begin{aligned}
R(\delta) &= \frac{n(1-\delta) + \dfrac{[1-(1-\varepsilon')^{\frac{1}{k-\ell}}]n}{(1-\varepsilon')^{\frac{1}{k-\ell}}}}{n\left(\dfrac{1}{k}+\delta\right) + \dfrac{[1-(1-\varepsilon')^{\frac{1}{k-\ell}}]n}{(1-\varepsilon')^{\frac{1}{k-\ell}}}} \\
&= \frac{n(1-\delta)(1-\varepsilon')^{\frac{1}{k-\ell}} + [1-(1-\varepsilon')^{\frac{1}{k-\ell}}]n}{n\left(\dfrac{1}{k}+\delta\right)(1-\varepsilon')^{\frac{1}{k-\ell}} + [1-(1-\varepsilon')^{\frac{1}{k-\ell}}]n} \\
&= \frac{1-\delta(1-\varepsilon')^{\frac{1}{k-\ell}}}{\dfrac{(1-\varepsilon')^{\frac{1}{k-\ell}}}{k} + \delta(1-\varepsilon')^{\frac{1}{k-\ell}} + 1 - (1-\varepsilon')^{\frac{1}{k-\ell}}} \\
&= \frac{k - k\delta(1-\varepsilon')^{\frac{1}{k-\ell}}}{(1-\varepsilon')^{\frac{1}{k-\ell}} + k\delta(1-\varepsilon')^{\frac{1}{k-\ell}} + k - k(1-\varepsilon')^{\frac{1}{k-\ell}}} \\
&\geq \frac{k-k\delta}{k-(k-1)(1-\varepsilon')^{\frac{1}{k-\ell}} + k\delta} \\
&= \frac{k}{k-(k-1)(1-\varepsilon')^{\frac{1}{k-\ell}}} - \gamma(\delta,k).
\end{aligned}$$

Since we have $\varepsilon' = \varepsilon + \dfrac{k(k+1)}{n} = \varepsilon + o(1)$, we obtain the inapproximability factor

$$\frac{k}{k-(k-1)(1-\varepsilon')^{\frac{1}{k-\ell}}} - \gamma(\delta,k) = \frac{k}{k-(k-1)(1-\varepsilon)^{\frac{1}{k-\ell}}} - \mathrm{O}\left(\frac{1}{n^{\frac{1}{k}}}\right) - \gamma(\delta,k).$$

The remaining part to be proved is the density condition of $\mathcal{H}'$. Let us introduce the abbreviation $N = |V(\mathcal{H}')|$. Note that we have $n = (1-\varepsilon')^{\frac{1}{k-\ell}}N$. We will show that

$$d_{\mathcal{H}'}(S) \quad \geq \quad \varepsilon\binom{N-\ell}{k-\ell} \text{ is satisfied for all } S \in \binom{V(\mathcal{H}')}{\ell}. \tag{5.5}$$

Let us consider an arbitrary set $S \subseteq V(\mathcal{H}')$ with $|S| = \ell$. In order to derive a lower bound on $d_{\mathcal{H}'}(S)$, we may assume that $\{\, e \in E(\mathcal{H}') \mid e \subseteq V(\mathcal{H}) \,\} = \varnothing$

and get

$$\begin{aligned}
d_{\mathcal{H}'}(S) &\geq \binom{N-\ell}{k-\ell} - \binom{n-\ell}{k-\ell}\\
&= \binom{N-\ell}{k-\ell} - \binom{(1-\varepsilon')^{k-\ell}N-\ell}{k-\ell}\\
&\geq \frac{(N-k+1)^{k-\ell}}{(k-\ell)!} - \frac{\left((1-\varepsilon')^{\frac{1}{k-\ell}}N-\ell\right)^{k-\ell}}{(k-\ell)!}\\
&= \frac{N^{k-\ell}\left[\left(1-\frac{(k+1)}{N}\right)^{k-\ell} - \left((1-\varepsilon')^{\frac{1}{k-\ell}}-\frac{\ell}{N}\right)^{k-\ell}\right]}{(k-\ell)!}\\
&= \frac{N^{k-\ell}\left[\left(1-\frac{(k+1)}{N}\right)^{k-\ell} - \left(\left(1-\varepsilon-\frac{k(k+1)}{n}\right)^{\frac{1}{k-\ell}}-\frac{\ell}{N}\right)^{k-\ell}\right]}{(k-\ell)!}\\
&\geq \frac{N^{k-\ell}\left[1-\frac{(k-\ell)(k+1)}{N}-1+\varepsilon+\frac{k(k+1)}{n}\right]}{(k-\ell)!}\\
&\geq \varepsilon\,\frac{N^{k-\ell}}{(k-\ell)!} \quad\geq\quad \varepsilon\binom{N-\ell}{k-\ell}.
\end{aligned}$$

Since the density condition (5.5) is satisfied, the proof of Theorem 5.5.8 follows. ■

NP-Hardness of Approximation

The former construction combined with the hard instance of the VERTEX COVER problem in k-hypergraphs in Theorem 5.4.3 yields the following **NP**-hardness of approximation result.

Theorem 5.5.9

For every $k \geq 3$, it is **NP***-hard to approximate the* VERTEX COVER *problem in (ε,ℓ)-dense k-hypergraphs to within any constant approximation ratio less than*

$$\frac{k-1}{k-1-(k-2)(1-\varepsilon)^{\frac{1}{k-\ell}}}.$$

Proof of Theorem 5.5.9.

Let us consider the k-hypergraph $\mathcal{H}$ given in Theorem 5.4.3, for which the following is **NP**-hard to decide.

(i) Every vertex cover of $\mathcal{H}$ has size at least $|V(\mathcal{H})|\,(1-\delta)$.

(ii) The cardinality of a minimum vertex cover of $\mathcal{H}$ is at most

$$|V(\mathcal{H})|\left(\frac{1}{k-1}+\delta\right).$$

We are going to densify the k-hypergraph $\mathcal{H}$ similarly to the construction in Theorem 5.5.8. Let us introduce the abbreviation $n = |V(\mathcal{H})|$. For a fixed $\varepsilon \in (0,1)$ and $(\ell+1) \in [k]$, we add the clique $\mathcal{K}$ of size

$\frac{[1-(1-\varepsilon)^{\frac{1}{k-\ell}}]n}{(1-\varepsilon)^{\frac{1}{k-\ell}}}$ to $\mathcal{H}$ and all edges $e \in \binom{V(\mathcal{K}) \cup V(\mathcal{H})}{k}$ with $e \cap V(\mathcal{K}) \neq \varnothing$.

Since we need to cover all the edges of the clique $\mathcal{K}$, the cases (i) and (ii) from Theorem 5.4.3 being **NP**-hard to decide transform into the following.

(iii) Every vertex cover of $\mathcal{H}'$ has size at least

$$n\,(1-\delta)+\frac{[1-(1-\varepsilon)^{\frac{1}{k-\ell}}]}{(1-\varepsilon)^{\frac{1}{k-\ell}}}\cdot n.$$

(iv) The cardinality of a minimum vertex cover of $\mathcal{H}'$ is at most

$$n\left(\frac{1}{k-1}+\delta\right)+\frac{[1-(1-\varepsilon)^{\frac{1}{k-\ell}}]}{(1-\varepsilon)^{\frac{1}{k-\ell}}}\cdot n.$$

For every $\delta > 0$, it implies the **NP**-hardness of approximating the VERTEX COVER problem in (ε,ℓ)-dense hypergraphs to within any constant approximation ratio less than $R(\delta)$, where $R(\delta)$ is defined as follows.

$$R(\delta)=\frac{n(1-\delta)+\dfrac{[1-(1-\varepsilon')^{\frac{1}{k-\ell}}]n}{(1-\varepsilon')^{\frac{1}{k-\ell}}}}{n\left(\dfrac{1}{k-1}+\delta\right)+\dfrac{[1-(1-\varepsilon')^{\frac{1}{k-\ell}}]n}{(1-\varepsilon')^{\frac{1}{k-\ell}}}}$$

We are going to derive a lower bound on the term $R(\delta)$.

$$\begin{aligned}
R(\delta) &= \frac{n(1-\delta) + \dfrac{[1-(1-\varepsilon')^{\frac{1}{k-\ell}}]n}{(1-\varepsilon')^{\frac{1}{k-\ell}}}}{n\left(\dfrac{1}{k-1}+\delta\right) + \dfrac{[1-(1-\varepsilon')^{\frac{1}{k-\ell}}]n}{(1-\varepsilon')^{\frac{1}{k-\ell}}}} \\
&= \frac{(1-\delta)(1-\varepsilon')^{\frac{1}{k-\ell}} + [1-(1-\varepsilon')^{\frac{1}{k-\ell}}]}{\left(\dfrac{1}{k-1}+\delta\right)(1-\varepsilon')^{\frac{1}{k-\ell}} + [1-(1-\varepsilon')^{\frac{1}{k-\ell}}]} \\
&= \frac{1-\delta(1-\varepsilon')^{\frac{1}{k-\ell}}}{\dfrac{(1-\varepsilon')^{\frac{1}{k-\ell}}}{k-1} + \delta(1-\varepsilon')^{\frac{1}{k-\ell}} + 1 - (1-\varepsilon')^{\frac{1}{k-\ell}}} \\
&\geq \frac{(k-1)-(k-1)\delta}{(k-1)-(k-2)(1-\varepsilon')^{\frac{1}{k-\ell}} + (k-1)\delta} \\
&\geq \frac{k-1}{k-1-(k-2)(1-\varepsilon')^{\frac{1}{k-\ell}}} - \gamma(\delta,k).
\end{aligned}$$

Since we have $\varepsilon' = \varepsilon + \dfrac{k(k+1)}{n}$, we obtain the inapproximability factor

$$\frac{k-1}{k-1-(k-2)(1-\varepsilon)^{\frac{1}{k-\ell}}} - o(1) - \gamma(\delta,k).$$

Since we have already proved that the density condition (5.5) holds, the proof of Theorem 5.5.9 follows. ■

5.6 Nearly Regular k-Hypergraphs

As a generalization of the class of dense k-hypergraphs, we investigate the approximability of the VERTEX COVER problem in nearly regular k-hypergraphs. We design a randomized approximation algorithm for the problem with approximation ratio strictly less than k and running time depending on the density of the underlying k-hypergraph. In particular, it entails the existence of quasi-polynomial and polynomial time approximation algorithms with approximation ratio less than k for mildly sparse and for subdense instances, respectively. On the other hand, we give tight approximation hardness results. In particular, we prove the best known approximation lower bound for the VERTEX COVER problem in regular k-hypergraphs with $k > 3$. Before we start to describe our algorithm, we review known results concerning the approximability of the VERTEX COVER problem in subdense and regular k-hypergraphs.

5.6.1 Previously Known Results

Let us first survey some known results for the VERTEX COVER problem restricted to d-regular k-hypergraphs focusing on instances with $d = \omega(1)$.

Regular k-Hypergraphs

The best known approximation algorithm for this restricted version of the problem is due to Halperin [H02] achieving an approximation ratio $k(1 - o(1))$. The algorithm is based on randomized rounding of a semidefinite programming relaxation.

Theorem 5.6.1 ([H02])
There is an efficient randomized approximation algorithm for the VERTEX COVER *problem in k-hypergraphs that on input of a k-hypergraph $\mathcal{H}$ computes a solution with approximation ratio*

$$k - k(k-1)\frac{\log\log\Delta_{\mathcal{H}}}{\log\Delta_{\mathcal{H}}} + o\left(\frac{\log\log\Delta_{\mathcal{H}}}{\log\Delta_{\mathcal{H}}}\right)$$

for all $k \in \mathbb{N}$ with $k^3 = o\left(\frac{\log\log\Delta_{\mathcal{H}}}{\log\log\log\Delta_{\mathcal{H}}}\right)$.

On the approximation hardness side, Alimonti and Kann [AK00] proved that the VERTEX COVER problem restricted to 3-regular graphs is **APX**-hard. Based on the case $d = 3$, Manthey [M05] extended the range of d proving the **APX**-hardness of the VERTEX COVER problem in d-regular graphs for all integer $d \geq 3$.

Theorem 5.6.2 ([M05])
The VERTEX COVER *problem restricted to d-regular graphs is* **APX***-hard for all integer* $d \geq 3$.

Feige [F03] constructed an approximation preserving construction implying that the VERTEX COVER problem in regular graphs is as hard to approximate as in graphs without any restrictions.

Theorem 5.6.3 ([F03])
If there is an approximation algorithm for the VERTEX COVER *problem with approximation ratio* ρ *in regular graphs, then, there is an approximation algorithm for the* VERTEX COVER *problem with approximation ratio* ρ *in every graph.*

Combining this reduction with the result due to Khot and Regev [KR08], it implies the UG-hardness of approximating the VERTEX COVER problem in regular graphs within any constant approximation ratio better than 2.

Corollary 5.6.1
It is UG*-hard to approximate the* VERTEX COVER *problem in regular graphs to within any constant approximation ratio less than* 2.

Nearly Regular k-Hypergraphs

Imamura and Iwama [II05] studied the VERTEX COVER problem in subdense graphs. They designed a randomized approximation algorithm with approximation ratio and running time depending on the average degree and maximum degree of the underlying graph. In more detail, on input of a graph $\mathcal{G}$, their algorithm recursively extracts subsets of a minimum vertex cover of $\mathcal{G}$ until a sufficiently small subgraph of $\mathcal{G}$ remains. On the remaining instance, they apply the simple approximation algorithm $\mathcal{A}_{5.1}$ with approximation ratio 2. In each recursion level, they use a randomized version of the approximation algorithm for the VERTEX COVER problem in dense graphs due to Karpinski and Zelikovsky [KZ97b]. More precisely, they obtained the following result.

Theorem 5.6.4 ([II05])
For every $\varepsilon > 0$, there is a randomized approximation algorithm for the VERTEX COVER *problem in graphs that on input of a graph $\mathcal{G}$ containing n vertices computes in* $\mathrm{poly}(n)\exp\left[\frac{n}{\Delta_{\mathcal{G}}}\log\log n\right]$ *time a solution with approximation ratio at most*

$$\frac{2}{1+\gamma(\mathcal{G})}+\varepsilon,$$

where $\gamma(\mathcal{G})$ is defines as

$$\gamma(\mathcal{G}) = \begin{cases} \dfrac{\overline{d}_{\mathcal{G}}}{2\Delta_{\mathcal{G}}} & \text{if} \quad |E(\mathcal{G})| \le \Delta_{\mathcal{G}}(n-\Delta_{\mathcal{G}}) \\ \dfrac{n+\Delta_{\mathcal{G}}-\sqrt{(n+\Delta_{\mathcal{G}})^2-4\overline{d}_{\mathcal{G}}\cdot n}}{2n} & \text{if} \quad |E(\mathcal{G})| > \Delta_{\mathcal{G}}(n-\Delta_{\mathcal{G}}). \end{cases}$$

In particular, their approximation algorithm for the VERTEX COVER problem achieves in polynomial time an approximation ratio smaller than 2 if the given graph $\mathcal{G}$ obeys the property $\overline{d}_{\mathcal{G}}/\Delta_{\mathcal{G}} \ge c$ for some constant $c > 0$ and $\Delta_{\mathcal{G}} = \Omega\big(n\log\log n/\log n\big)$.

Cardinal, Karpinski, Schmied and Viehmann studied subdense instances of several covering problems. In particular, for the VERTEX COVER problem in graphs, they obtained an improved result when the underlying graph $\mathcal{G}$ satisfies $\Delta_{\mathcal{G}} \le |V(\mathcal{G})|/2$.

Theorem 5.6.5 ([CKSV11])
For every $\varepsilon > 0$, there is a randomized approximation algorithm with an approximation ratio

$$\frac{2}{1+\dfrac{\overline{d}_{\mathcal{G}}}{2\Delta_{\mathcal{G}}}}+\varepsilon$$

for the VERTEX COVER *problem in graphs $\mathcal{G}$ having n vertices, average degree $\overline{d}_{\mathcal{G}}$ and maximum degree $\Delta_{\mathcal{G}} \le n/\psi_{\mathcal{G}}(n)$ with $\psi_{\mathcal{G}}(n) \ge 2$ in time* $\mathrm{poly}(n)\exp\left[\mathrm{O}(\psi_{\mathcal{G}}(n)\log\log\psi_{\mathcal{G}}(n))\right]$.

It means that they achieve in polynomial time an approximation ratio below 2 for a wider range of graphs, in particular, for graphs $\mathcal{G}$ with $\overline{d}_{\mathcal{G}}/\Delta_{\mathcal{G}} \ge c$ for some constant $c \in (0,1]$ and $\Delta_{\mathcal{G}} = \Omega\big(n\log\log\log n/\log n\big)$.

On the other hand, Imamura and Imawa [II05] proved that the approximation ratio achieved by their algorithm for the VERTEX COVER problem in Theorem 5.6.4 is optimal assuming that the approximation ratio 2 is best possible for the general problem. More precisely, they gave the following hardness result.

Theorem 5.6.6 ([II05])
For every constant $\delta > 0$, suppose that there is a polynomial time algorithm for the VERTEX COVER *problem in graphs $\mathcal{G}$ having n vertices and $\overline{d}_{\mathcal{G}} = \Omega(n^c)$ for a constant $c \in (0,1)$ which achieves an approximation ratio*

$$\frac{2}{1+\gamma(\mathcal{G})} - \delta$$

where $\gamma(\mathcal{G})$ is defied as follows.

$$\gamma(\mathcal{G}) = \begin{cases} \dfrac{\overline{d}_{\mathcal{G}}}{2\Delta_{\mathcal{G}}} & \text{if } |E(\mathcal{G})| \le \Delta_{\mathcal{G}}(n-\Delta_{\mathcal{G}}) \\ \dfrac{n+\Delta_{\mathcal{G}} - \sqrt{(n+\Delta_{\mathcal{G}})^2 - 4\overline{d}_{\mathcal{G}} \cdot n}}{2n} & \text{if } |E(\mathcal{G})| > \Delta_{\mathcal{G}}(n-\Delta_{\mathcal{G}}). \end{cases}$$

Then, there is a polynomial time approximation algorithm for the VERTEX COVER *problem in graphs with approximation ratio $(2-\delta)$.*

5.6.2 Our Contributions

We give a randomized approximation algorithm for the VERTEX COVER problem in nearly regular hypergraphs with approximation ratio strictly less than k and running time depending on the density of the underlying k-hypergraph. In particular, it entails the existence of quasi-polynomial and polynomial time approximation algorithms with approximation ratio less than k for mildly sparse instances and for subdense instances, respectively. Let us give the precise statement.

Theorem 5.6.7
For every $\varepsilon > 0$ and $k \ge 2$, there is an approximation algorithm which com-

putes with probability at least $3/4$ *a vertex cover for a given* r*-nearly regular* k*-hypergraph* $\mathcal{H}$ *containing* n *vertices with approximation ratio*

$$\frac{k}{1+r-\frac{r}{k}}+\varepsilon$$

in $\mathrm{poly}(n)\exp\left[\mathrm{O}(\ \Psi_{\mathcal{H}}(n)\ )\right]$ *time.*

Let us first formulate several corollaries of the main theorem. By setting $\Psi_{\mathcal{H}}(n) = \mathrm{O}(\log(n))$ or $\Psi_{\mathcal{H}}(n) = \mathrm{poly}(\log(n))$, we obtain the following corollary.

Corollary 5.6.2
The VERTEX COVER *problem is approximable in subdense and mildly sparse* r*-nearly regular* k*-hypergraphs with approximation ratio arbitrarily close to*

$$\frac{k}{1+r\ -\frac{r}{k}}$$

in polynomial time and quasi-polynomial time, respectively.

If we additionally assume that the given k-hypergraph is regular, it yields the following corollary.

Corollary 5.6.3
The VERTEX COVER *problem is approximable in subdense and mildly sparse regular* k*-hypergraphs with approximation ratio arbitrarily close to*

$$\frac{k}{2\ -\ \frac{1}{k}}$$

in polynomial time and quasi-polynomial time, respectively.

By setting $k = 2$ in Theorem 5.6.7, our algorithm achieves the same approximation ratio as in Theorem 5.6.5, since a graph $\mathcal{G}$ with $\overline{d}_{\mathcal{G}}/\Delta_{\mathcal{G}} \geq c$ for some $c \in (0,1)$ is also c-nearly regular. Moreover, it is applicable on a wider range of graphs by a factor of $\mathrm{O}\big(\log\log\Psi_{\mathcal{G}}(n)\big)$ since $\overline{d}_{\mathcal{G}}/\Delta_{\mathcal{G}} = \Theta(1)$ implies that

$$\psi_{\mathcal{G}}(n) = \frac{n}{\Delta_{\mathcal{G}}} = \Theta\left(\frac{n}{\overline{d}_{\mathcal{G}}}\right) = \Theta\left(\frac{n^2}{|E(\mathcal{G})|}\right) = \Theta\left(\Psi_{\mathcal{G}}(n)\right) \qquad (5.6)$$

and the running time of the algorithm given in [CKSV11] can be written as

$$\mathrm{poly}(n)\exp\left[\mathrm{O}\big(\,\Psi_{\mathcal{H}}(n)\log\log\Psi(n)\,\big)\right].$$

On the one hand, our algorithm achieves an approximation ratio smaller than 2 for the whole class of subdense nearly regular graphs and on the other hand, it can be applied to k-hypergraphs with $k > 2$.

A crucial ingredient of our approximation algorithm for the VERTEX COVER problem in nearly regular k-hypergraphs is the Sampling Lemma given below.

Lemma 5.6.1 (Sampling Lemma)
Let $\mathcal{H}$ be a k-hypergraph containing n vertices, C a minimum vertex cover of $\mathcal{H}$. For every constant $\gamma \in (3/4, 1)$, there is a randomized algorithm that constructs in polynomial time a collection $\widehat{W}$ of subsets $W_i \subseteq V(\mathcal{H})$ with the following properties.

(*i*) *The size of $\widehat{W}$ is $|\widehat{W}| \le f(\gamma)$, where $f(\gamma) = \mathrm{O}(1)$.*

(*ii*) *Every set $W \in \widehat{W}$ has cardinality at least*

$$|W| \;\ge\; \left[1-\left(1-\frac{1}{\Psi_{\mathcal{H}}(n)}\right)^{\frac{1}{k}}\right](n-k+1). \tag{5.7}$$

(*iii*) *There is a set $W' \in \widehat{W}$ such that $|W' \cap C| \ge \gamma|W'|$ with probability at least γ.*

In contrast to the algorithm given in [CKSV11], we use a decreased sample size of $\mathrm{O}(1)$ compared to $\mathrm{O}(\,[\log(\Psi_{\mathcal{G}}(n))]^2\,)$ leading to an improved running time of our algorithm. Note that this is a randomized version of the extraction algorithm $\mathcal{A}_{5.3}$ defined in Section 5.5.3. In comparison to algorithm $\mathcal{A}_{5.3}$, the randomized version constructs a set $\widehat{W}$ of size $\mathrm{O}(1)$ instead of $\mathrm{O}(n^k)$.

On the approximation hardness side, we obtain two inapproximability results. The first approximation lower bound matches exactly the approximation ratio achieved in Theorem 5.6.7 in a specified range of the density of the given k-hypergraph.

Theorem 5.6.8
For every $k \geq 2$, it is UG*-hard to approximate the* VERTEX COVER *problem in r-nearly regular k-hypergraphs $\mathcal{H}$ having n vertices, maximum degree $\Delta_{\mathcal{H}} = \Omega(n^{\varepsilon})$ and $|E(\mathcal{H})| = o(n^k)$ to within any constant approximation ratio less than*

$$\frac{k}{1 + (k-1)\frac{r}{k}}$$

for every $\varepsilon > 0$.

In addition, we prove a **NP**-hardness of approximation result for the VERTEX COVER problem in nearly regular hypergraphs with density and regularity conditions.

Theorem 5.6.9
For every $k \geq 3$, it is **NP***-hard to approximate the* VERTEX COVER *problem in r-nearly regular k-hypergraphs $\mathcal{H}$ having n vertices, maximum degree $\Delta_{\mathcal{H}} = \Omega(n^{\varepsilon})$ and $|E(\mathcal{H})| = o(n^k)$ to within any constant approximation ratio less than*

$$\frac{k-1}{1 + (k-2)\frac{r}{k}}$$

for every $\varepsilon > 0$.

As a corollary, we obtain the best known approximation lower bounds for the VERTEX COVER problem in regular k-hypergraphs.

Corollary 5.6.4
It is UG*-hard and* **NP***-hard to approximate the* VERTEX COVER *problem in regular k-hypergraphs $\mathcal{H}$ to within any constant approximation ratio less than*

$$\frac{k}{2 - \frac{1}{k}} \text{ for every } k \geq 2 \text{ and } \frac{k}{2} \text{ for every } k \geq 3, \text{ respectively.}$$

5.6.3 The Randomized Approximation Algorithm

In the subsequent sections, we are going to prove Theorem 5.6.7 restated below.

Theorem 5.6.7

For every $\varepsilon > 0$ and $k \geq 2$, there is an approximation algorithm which computes with probability at least $3/4$ a vertex cover for a given r-nearly regular k-hypergraph $\mathcal{H}$ containing n vertices with approximation ratio

$$\frac{k}{1 + r - \frac{r}{k}} + \varepsilon$$

in $\operatorname{poly}(n) \exp\left[\mathrm{O}(\ \Psi_{\mathcal{H}}(n)\ )\right]$ *time.*

Let us first describe the main ideas of the proof and give an overview of the techniques.

Overview and Main Ideas

Given a r-nearly regular k-hypergraph $\mathcal{H}$, the algorithm first iteratively removes vertex subsets until a sufficiently small set of vertices remain. Then, we apply the simple approximation algorithm $\mathcal{A}_{5.1}$ with approximation ratio k on the remaining induced hypergraph. Suppose that at every step i of the algorithm, we are able to guess a sufficiently large subset of an optimal solution of the current hypergraph $\mathcal{H}_i$. This subset of vertices is removed together with the edges, that they cover, forming $\mathcal{H}_{i+1}$. In the next subsection, we will see how we can sample the set $\widetilde{W}$ computed by algorithm $\mathcal{A}_{5.4}$ to perform this guessing step efficiently. The union of the removed sets will form the set W allowing us to use Lemma 5.5.2. We aim at obtaining such a set W of size approximately

$$\beta_{\mathcal{H}}|V(\mathcal{H})|, \text{ where } \beta_{\mathcal{H}} = \frac{r}{k}\,.$$

Letting $\mathcal{H}_i$ be the hypergraph considered in the i-th step, we denote by n_i its number of vertices ($n_1 = n = |V(\mathcal{H})|$), by E_i its edge set and define

$$\Psi_i(n_i) = \binom{n_i}{k}(|E_i|)^{-1}.$$

In addition, we introduce $s_i = n_i - (1 - \beta_{\mathcal{H}})n$. Note that if $s_i = 0$, we have $n_i = (1 - \beta_{\mathcal{H}})n$ and thus, $n - n_i = \beta_{\mathcal{H}} n$. Since $n - n_i$ is the size of the extracted set W, s_i can serve as a measure of progress.

A key role will play the Sampling Lemma (Lemma 5.6.1). It enables us to efficiently guess a large subset of a minimum vertex cover of the actual hypergraph while introducing only a small number of candidate sets. Let us give the precise statement

Lemma 5.6.1 (Sampling Lemma)
Let $\mathcal{H}$ be a k-hypergraph containing n vertices, C a minimum vertex cover of $\mathcal{H}$. For every constant $\gamma \in (3/4, 1)$, there is a randomized algorithm that constructs in polynomial time a collection $\widehat{W}$ of subsets $W_i \subseteq V(\mathcal{H})$ with the following properties.

(*i*) *The size of $\widehat{W}$ is $|\widehat{W}| \leq f(\gamma)$, where $f(\gamma) = \mathrm{O}(1)$.*

(*ii*) *Every set $W \in \widehat{W}$ has cardinality at least*

$$|W| \ \geq \ \left[1 - \left(1 - \frac{1}{\Psi_{\mathcal{H}}(n)}\right)^{\frac{1}{k}}\right](n - k + 1). \tag{5.7}$$

(*iii*) *There is a set $W' \in \widehat{W}$ such that $|W' \cap C| \geq \gamma |W'|$ with probability at least γ.*

For technical reasons, we will only extract $(\beta_{\mathcal{H}} - \alpha)n$ vertices of the given k-hypergraph for a small constant $\alpha \in (0, rk^{-2})$. Due to the Sampling Lemma, we are able to extract $[1 - (1 - 1/\Psi_i(n_i))^{1/k}](n_i - k + 1)$ vertices of the actual hypergraph $\mathcal{H}_i$. In order to derive the number of iterations that are needed to extract the desired number of vertices, we have to obtain a lower bound on the number of extracted vertices in terms of $|V(\mathcal{H})|$ and the density of $\mathcal{H}$. In addition, we prove that the actual hypergraph $\mathcal{H}_i$ is still dense enough to extract sufficiently many vertices. More precisely, we will prove the following lemma.

Lemma 5.6.2
Let $\mathcal{H}$ be a r-nearly regular k-hypergraph and $\alpha \in (0, rk^{-2})$. Then, for every

k-hypergraph $\mathcal{H}_i = \mathcal{H}[V(\mathcal{H})\backslash V_i]$ with $V_i \subseteq V(\mathcal{H})$ and $|V_i| \le (\beta_{\mathcal{H}} - \alpha)n$, we have

$$\left[1 - \left(1 - \frac{1}{\Psi_i(n_i)}\right)^{\frac{1}{k}}\right](n_i - k + 1) \;\ge\; \alpha\frac{n}{\Psi_{\mathcal{H}}(n)}\,. \tag{5.8}$$

By combining the Sampling lemma and Lemma 5.6.2, we are able to remove in each step

$$\frac{\alpha \cdot n}{\Psi_{\mathcal{H}}(n)} \text{ vertices until } s_i \le \alpha n \text{ for a fixed constant } \alpha \in \left(0, \frac{1}{2k}\right).$$

Therefore, at the end of the extraction process, we obtain $s_i \le \alpha n$ implying $|W| \ge (\beta_{\mathcal{H}} - \alpha)n$. The number $T_{\mathcal{H}}(\alpha)$ of steps required to extract $(\beta_{\mathcal{H}} - \alpha)n$ vertices is at most

$$T_{\mathcal{H}}(\alpha) \;=\; \frac{(\beta_{\mathcal{H}} - \alpha)n}{b_{\mathcal{H}}(\alpha)} \;=\; \frac{(\beta_{\mathcal{H}} - \alpha)n}{\frac{\alpha \cdot n}{\Psi_{\mathcal{H}}(n)}} \;=\; \Psi_{\mathcal{H}}(n)\left(\frac{\beta_{\mathcal{H}}}{\alpha} - 1\right). \tag{5.9}$$

Finally, we define a collecting algorithm and give a lower bound on the expected number of collected vertices being contained in a minimum vertex cover of $\mathcal{H}$.

As for the running time, the algorithm generates a search tree of height $T_{\mathcal{H}}(\alpha)$ and fan-out less than $f(\gamma)$. At every node of the tree, it takes $\mathrm{poly}(n) + \mathrm{O}\big(f(\gamma)\big)$ time to generate at most $f(\gamma)$ candidate subsets. Accordingly, the overall running time of the algorithm is $\mathrm{poly}(n) \cdot (f(\gamma))^{\mathrm{O}(T_{\mathcal{H}}(\alpha))} = \mathrm{poly}(n)\exp\big[\mathrm{O}\big(\Psi_{\mathcal{H}}(n)\big)\big]$.

The Sampling Lemma

We are going to define a recursive sampling algorithm that allows us to efficiently guess subsets of a minimum vertex cover of a given k-hypergraph. In particular, it entails proving the Sampling Lemma (Lemma 5.6.1).

Before we define our recursive sampling algorithm, we introduce some notation to be used in the detailed description of the algorithm. Given a k-hypergraph $\mathcal{H}$ containing n vertices, we define the parameter $\widehat{h}(\mathcal{H})$ by

$$\widehat{h}(\mathcal{H}) \;=\; \left(1 - \left[1 - \frac{1}{\Psi_{\mathcal{H}}(n)}\right]^{1/k}\right)(n - k + 1). \tag{5.10}$$

In addition, we introduce the sample size $\widehat{s}(\gamma, k)$ in dependence of the parameter $\gamma \in (0,1)$ and $k \geq 2$ defined by

$$\widehat{s}(\gamma, k) \quad = \quad \left\lceil \log\left(1-\gamma^{\frac{1}{k-1}}\right)\left(\log(\gamma)\right)^{-1}\right\rceil . \tag{5.11}$$

The Algorithm $\mathcal{A}_{5.6}$ defined in Figure 5.6 returns a small set of candidate subsets. It is a randomized version of the Algorithm $\mathcal{A}_{5.3}$ given in Section 5.5.3.

```
Algorithm $\mathcal{A}_{5.6}$

  Input  : $(\mathcal{H}, t, l)$, where $\mathcal{H}$ is a $k$-hypergraph $\mathcal{H}$, $t \in \mathbb{N}$ with
           $t \leq \widehat{h}(\mathcal{H})$ and $l \in \mathbb{N}$ with $l \geq \widehat{s}(\gamma, k)$.
  Output: A collection $\widehat{W}$ of subsets $W_i \subseteq V(\mathcal{H})$.

  begin
     ① $\widehat{W} = \emptyset$;
     if $k = 1$ then
        ② $\widehat{W} \leftarrow \{A\}$, where $A$ is a set of $t$ vertices in $V(E(\mathcal{H}))$;
        return $\widehat{W}$;
     else
        ③ Let $B$ be a set of $t$-heaviest vertices in $V(\mathcal{H})$;
        ④ $\widehat{W} \leftarrow \widehat{W} \cup \{B\}$;
        ⑤ Let $B'$ be a set of $l$ uniformly at random chosen vertices
        from $B$;
        foreach $v \in B'$ do
           ⑥ $\widehat{W}' \leftarrow \mathcal{A}_{5.6}(\mathcal{H}(v), t, l)$;
           ⑦ $\widehat{W} \leftarrow \widehat{W} \cup \widehat{W}'$;
        end
     end
     return $\widehat{W}$;
  end
```

Figure 5.6: Algorithm $\mathcal{A}_{5.6}$

We are going to prove the following lemma which generalizes the Sampling Lemma.

Lemma 5.6.3

Let $\mathcal{H}$ be a k-hypergraph containing n vertices, C a minimum vertex cover of $\mathcal{H}$. Furthermore, let $\gamma \in (3/4, 1)$, $t \in \mathbb{N}$ with $t \le \widehat{h}(\mathcal{H})$ and $l \in \mathbb{N}$ with $l \ge \widehat{s}(\gamma, k)$. Then, on input $(\mathcal{H}, t, l)$, the algorithm $\mathcal{A}_{5.6}$ constructs in polynomial time a set $\widehat{W}$ containing subsets of $V(\mathcal{H})$ with the following properties.

(i) *The size of $\widehat{W}$ is at most $|\widehat{W}| \le (l+1)^{k-1}$.*

(ii) *Every set $W \in \widehat{W}$ has cardinality $|W| = t$.*

(iii) *There is a set $W' \in \widehat{W}$ such that $|W' \cap C| \ge \gamma|W'|$ with probability at least γ.*

Before we give the proof of Lemma 5.6.3, we argue that it implies the Sampling Lemma. By letting $t = \widehat{h}(\mathcal{H})$ and $l = (\widehat{s}(\gamma, k) + 1)^{k-1}$, we obtain $f(\gamma) = (\widehat{s}(\gamma, k) + 1)^{k-1} = \mathrm{O}(1)$ and all other properties described in the Sampling Lemma. We now give the proof of Lemma 5.6.3.

Proof of Lemma 5.6.3.
Let $\mathcal{H}$ be a k-hypergraph, C a vertex cover of $\mathcal{H}$, $\gamma \in (3/4, 1)$ and B a set consisting of $\widehat{h}(\mathcal{H})$-heaviest vertices of $\mathcal{H}$.
(i) Since we create at most $(l+1)$ sets in each recursion level, it yields at most $(l+1)^{k-1}$ subsets of $V(\mathcal{H})$ after $(k-1)$ recursions.
(ii) This property follows by definition of the set A and B in the description of algorithm $\mathcal{A}_{5.6}$.
(iii) According to Corollary 5.5.1, we have that $|C| \ge \widehat{h}(\mathcal{H})$ implying that it is possible to extract $\widehat{h}(\mathcal{H})$ vertices from C. We will distinguish two cases. If $|B \cap C| \ge \gamma|B|$, we have nothing to show.

Hence, we may assume that $|B \cap C| < \gamma|B|$ holds. The probability that a random vertex of B belongs to C is at most γ. Thus, with probability at least $1 - \gamma^{\widehat{s}(\gamma,k)}$, we get a vertex $v \notin C$ in the selected sample and C must contain a vertex cover of the v-induced hypergraph $\mathcal{H}(v)$. By Lemma 5.5.4, we know that $\mathcal{H}(v)$ is dense enough in order to extract the desired number of vertices. By iterating this step at most $(k-1)$-times, we obtain that the probability is at least

$$\left(1-\gamma^{l}\right)^{k-1} \;\geq\; \left(1-\gamma^{\widehat{s}(\gamma,k)}\right)^{k-1} \;\geq\; \left(1-\left[1-\gamma^{\frac{1}{k-1}}\right]\right)^{k-1} \;\geq\; \gamma,$$

by definition of $\widehat{s}(\gamma,k)$. Furthermore, we notice that this argumentation holds for all $t \leq \widehat{h}(\mathcal{H})$ and all vertex cover C of $\mathcal{H}$, and the proof follows. ■

Deriving a Lower Bound on the Size of the Extracted Set

In order to bound the number of iterations needed to extract a large enough subset of a minimum vertex cover of $\mathcal{H}$, we have to provide a lower bound on the number of vertices obtained in every iteration in terms of the density and the cardinality of the vertex set of $\mathcal{H}$. In particular, we are going to prove Lemma 5.6.2.

Lemma 5.6.2

Let $\mathcal{H}$ be a r-nearly regular k-hypergraph and $\alpha \in (0, rk^{-2})$. Then, for every k-hypergraph $\mathcal{H}_i = \mathcal{H}[V(\mathcal{H})\backslash V_i]$ with $V_i \subseteq V(\mathcal{H})$ and $|V_i| \leq (\beta_{\mathcal{H}} - \alpha)n$, we have

$$\left[1-\left(1-\frac{1}{\Psi_i(n_i)}\right)^{\frac{1}{k}}\right](n_i-k+1) \;\geq\; \alpha\frac{n}{\Psi_{\mathcal{H}}(n)}\,. \tag{5.8}$$

In order to give the proof of Lemma 5.6.2, we will provide three intermediate results.

Lemma 5.6.4

Let $\mathcal{H}$ be a k-hypergraph with n vertices, maximum degree $\Delta_{\mathcal{H}}$ and maximum independent set of size at least x. Then, $\mathcal{H}$ has at most $\Delta_{\mathcal{H}}(n-x)$ edges.

Proof of Lemma 5.6.4.

Let $\mathcal{H}$ be a k-hypergraph with n vertices, maximum degree $\Delta_{\mathcal{H}}$ and maximum independent set of size at least x. Every vertex covers at most $\Delta_{\mathcal{H}}$ edges. Hence, the size of a minimum vertex cover of $\mathcal{H}$, say τ, satisfies $|E(\mathcal{H})| \leq \Delta_{\mathcal{H}}\tau$. Also, by definition, the largest independent set of $\mathcal{H}$ has size $n-\tau \geq x$. Consequently, we obtain $|E(\mathcal{H})| \;\leq\; \tau\cdot\Delta_{\mathcal{H}} \;\leq\; (n-x)\Delta_{\mathcal{H}}$. ■

Next, we are going to prove a lower bound on the number of edges contained in the actual k-hypergraph $\mathcal{H}_i$.

Lemma 5.6.5

Let $\mathcal{H}$ be a r-nearly regular k-hypergraph with maximum degree $\Delta_{\mathcal{H}}$ and $V_i \subseteq V(\mathcal{H})$ with $|V_i| \leq \beta_{\mathcal{H}}|V(\mathcal{H})|$. Then, the size of the edge set E_i of the k-hypergraph $\mathcal{H}_i$ defined by $\mathcal{H}_i = \mathcal{H}[V(\mathcal{H})\backslash V_i]$ having $n_i = s_i + (1 - \beta_{\mathcal{H}})|V(\mathcal{H})|$ vertices can be bounded from below as follows.

$$|E_i| \;\geq\; \Delta_{\mathcal{H}} s_i$$

Proof of Lemma 5.6.5.

Let $\mathcal{H}$ be a r-nearly regular k-hypergraph having n vertices and maximum degree $\Delta_{\mathcal{H}}$. Furthermore, let $V_i \subseteq V(\mathcal{H})$ with $|V_i| \leq \beta_{\mathcal{H}} n$. We denote by E_i the edge set of the k-hypergraph $\mathcal{H}_i$ defined by $\mathcal{H}_i = \mathcal{H}[V(\mathcal{H})\backslash V_i]$ and by n_i the size of the vertex set of $\mathcal{H}_i$. Clearly, we have $s_i = n_i - (1 - \beta_{\mathcal{H}})n$. Let us introduce the k-hypergraph $\mathcal{H}'$ defined by $V(\mathcal{H}') = V(\mathcal{H})$ and $E(\mathcal{H}) = E(\mathcal{H})\backslash E_i$. Note that $\mathcal{H}'$ has an independent set of size at least $n_i = s_i + (1 - \beta_{\mathcal{H}})n$ as by definition, all the vertices of $\mathcal{H}_i$ form an independent set in $\mathcal{H}'$. Thus, by Lemma 5.6.4, $\mathcal{H}'$ can have at most

$$\Delta_{\mathcal{H}}[n - (s_i + [1 - \beta_{\mathcal{H}}]n)] \;=\; \Delta_{\mathcal{H}}(\beta_{\mathcal{H}} n - s_i) \tag{5.12}$$

edges. By the r-nearly regularity of $\mathcal{H}$ and (5.12), we obtain the following inequality.

$$|E_i| \;=\; |E(\mathcal{H})| - |E(\mathcal{H}')| \;\geq\; \frac{\Delta_{\mathcal{H}} \cdot r \cdot n}{k} - \Delta_{\mathcal{H}}(\beta_{\mathcal{H}} n - s_i)$$

Then, by definition of $\beta_{\mathcal{H}}$, we get

$$|E_i| \;\geq\; \Delta_{\mathcal{H}} \cdot \beta_{\mathcal{H}} n - \Delta_H(\beta_{\mathcal{H}} n - s_i) \;=\; \Delta_{\mathcal{H}} s_i$$

and the proof of Lemma 5.6.5 follows. ■

Finally, we will need the following simple inequality, which can be proved inductively.

Lemma 5.6.6

For all real $\varepsilon \in [0,1]$ and all integer $k \geq 1$, we have $1 - (1-\varepsilon)^{\frac{1}{k}} \;\geq\; \frac{\varepsilon}{k}$.

Thus far, we are ready to give the proof of Lemma 5.6.2.

Proof of Lemma 5.6.2.

Let $\mathcal{H}$ be a r-nearly regular k-hypergraph containing n vertices and $\alpha \in [0, rk^{-2})$. Furthermore, let $V_i \subseteq V(\mathcal{H})$ with $|V_i| \leq (\beta_{\mathcal{H}} - \alpha)n$ and $\mathcal{H}_i$ be the k-hypergraph given by $\mathcal{H}_i = \mathcal{H}[V(\mathcal{H})\backslash V_i]$. We denote by n_i the number of vertices contained in $\mathcal{H}_i$ and define $s_i = n_i - (1 - \beta_{\mathcal{H}})n$. We are going to consider two different cases. Let us start with $n_i = n$. Since we may assume that $n \geq k$ holds, we obtain by choice of α that

$$\begin{aligned}\left[1 - \left(1 - \frac{1}{\Psi_{\mathcal{H}}(n)}\right)^{\frac{1}{k}}\right](n-k+1) &\geq \frac{(n-k+1)}{\Psi_{\mathcal{H}}(n)k} && \text{(by Lemma 5.6.6)}\\ &\geq \frac{\alpha \cdot n}{\Psi_{\mathcal{H}}(n)} && \text{(by choice of } \alpha \text{)}\end{aligned}$$

Otherwise, we may assume that $n > n_i$ holds. Due to Lemma 5.6.6, we have

$$\left[1 - \left(1 - \frac{1}{\Psi_i(n_i)}\right)^{\frac{1}{k}}\right]\frac{n_i - k + 1}{s_i} \geq \frac{(n_i - k + 1)}{\Psi_i(n_i)k \cdot s_i}. \tag{5.13}$$

Then, the right hand side of (5.13) can be further simplified as follows.

$$\begin{aligned}\frac{(n_i-k+1)}{\Psi_i(n_i)ks_i} &\geq \frac{|E_i|(n_i-k+1)}{ks_i\binom{n_i}{k}}\\ &\geq \frac{\Delta_{\mathcal{H}}s_i(n_i-k+1)}{ks_i\binom{n_i}{k}} && \text{(by Lemma 5.6.5)}\\ &= \frac{\Delta_{\mathcal{H}}}{\binom{n_i}{k-1}} \geq \frac{k|E(\mathcal{H})|}{n\binom{n_i}{k-1}}\\ &\geq \frac{|E(\mathcal{H})|}{\binom{n}{k}} = \frac{1}{\Psi_{\mathcal{H}}(n)}\end{aligned}$$

Combining the deduced facts together with $s_i \geq \alpha \cdot n$, we have that

$$\left[1 - \left(1 - \frac{1}{\Psi_i(n_i)}\right)^{\frac{1}{k}}\right](n_i - k + 1) \geq \frac{s_i}{\Psi_{\mathcal{H}}(n)} \geq \frac{\alpha \cdot n}{\Psi_{\mathcal{H}}(n)}$$

and the proof follows. ■

The Collecting Algorithm

We are going to define an algorithm that extracts and collects all candidate sets in each iteration. Before we describe the algorithm, we need to introduce some notation.

Given a r-nearly regular k-hypergraph $\mathcal{H}$ with n vertices and a constant $\alpha \in (0, rk^{-2})$, we will use the abbreviation $\widehat{b}_{\mathcal{H}}(\alpha)$ defined by

$$\widehat{b}_{\mathcal{H}}(\alpha) \quad = \quad \frac{\alpha \cdot n}{\Psi_{\mathcal{H}}(n)}. \tag{5.14}$$

Recall that due to Lemma 5.6.2, we are able to extract $\widehat{b}_{\mathcal{H}}(\alpha)$ vertices of $\mathcal{H}$ in each iteration provided $s_i \leq \alpha n$ holds. The number $T_{\mathcal{H}}(\alpha)$ of required steps in order to extract $(\beta_{\mathcal{H}} - \alpha)n$ vertices is

$$T_{\mathcal{H}}(\alpha) \;=\; \frac{(\beta_{\mathcal{H}} - \alpha)n}{\widehat{b}_{\mathcal{H}}(\alpha)} \;=\; \Psi_{\mathcal{H}}(n)\left(\frac{\beta_{\mathcal{H}}}{\alpha} - 1\right). \tag{5.15}$$

The algorithm $\mathcal{A}_{5.7}$ defined in Figure 5.7 iterates this extraction until $s_i \leq \alpha n$ using exactly $T_{\mathcal{H}}(\alpha)$ recursion levels.

```
Algorithm $\mathcal{A}_{5.7}$
    Input  : $(\mathcal{H}, b, t, \gamma)$, where $\mathcal{H}$ is a $r$-nearly regular $k$-hypergraph,
             $b \in \mathbb{N}$ with $b \leq \widehat{b}_{\mathcal{H}}(rk^{-2})$, $t \in \mathbb{N}$ with $t \geq 0$ and
             $\gamma \in (1 - rk^{-2}, 1)$.
    Output: A collection $\widehat{C}$ of subsets $W_i \subseteq V(\mathcal{H})$.
    ① $\widehat{C} \leftarrow \varnothing$;
    if $(t > 0)$ then
        ② $\widehat{X} \leftarrow \mathcal{A}_{5.6}(\mathcal{H}, b, \widehat{s}(\gamma, k))$;
        ③ $\widehat{C} \leftarrow \{X \cup Y \mid X \in \widehat{X}, Y \in \widehat{C}' \leftarrow \mathcal{A}_{5.7}(\mathcal{H}[V(\mathcal{H}) \backslash X], b, t-1, \gamma)\}$;
    else
        return $\widehat{C}$;
    end
```

Figure 5.7: Algorithm $\mathcal{A}_{5.7}$

On input of a r-nearly regular k-hypergraph, algorithm $\mathcal{A}_{5.7}$ performs a recursive exploration of a search tree branching on every subset X in the set of candidates $\widehat{C}$. A root-to-leaf path in this tree yields a set W defined as the union of all the candidates X selected along the path. We now prove that this search tree contains a path yielding a suitable set W with high probability.

Lemma 5.6.7

Let $\mathcal{H}$ be a r-nearly regular k-hypergraph containing n vertices, C a minimum vertex cover of $\mathcal{H}$ and $\gamma \in (1 - rk^{-2}, 1)$. On input

$$(\mathcal{H}, \widehat{b}_{\mathcal{H}}(1-\gamma), T_{\mathcal{H}}(1-\gamma), \gamma),$$

the algorithm $\mathcal{A}_{5.7}$ constructs in $\operatorname{poly}(n)\exp[\mathrm{O}(\Psi_{\mathcal{H}}(n))]$ *time a collection $\widehat{C} = \{W_i \mid W_i \subseteq V(\mathcal{H})\}$ with the following properties.*

- (i) *The size of $\widehat{C}$ is at most $(\widehat{s}(\gamma, k) + 1)^{(k-1)T_{\mathcal{H}}(1-\gamma)}$.*
- (ii) *Every set $W \in \widehat{C}$ contains exactly $(\beta_{\mathcal{H}} - (1-\gamma))n$ vertices.*
- (iii) *For every $\delta \in (0,1)$, there is a $W' \in \widehat{C}$ such that $|W' \cap C| \geq (1-\delta)\gamma^2|W'|$ with probability at least*

$$1 - \exp\left[-\Psi_{\mathcal{H}}(n)\left(\frac{\beta_{\mathcal{H}}}{1-\gamma} - 1\right)\gamma \cdot \frac{\delta^2}{2}\right].$$

Proof of Lemma 5.6.7.

Let $\mathcal{H}$ be a r-nearly regular k-hypergraph containing n vertices and C a minimum vertex cover of $\mathcal{H}$.

(i) According to Lemma 5.6.3, the size of the actual set $\widehat{C}$ is increased by a multiplicative factor of at most $(\widehat{s}(\gamma, k)+1)^{k-1}$ in every iteration of algorithm $\mathcal{A}_{5.7}$. By iterating this extraction $T_{\mathcal{H}}(1-\gamma)$ times, we obtain a set with size at most $(\widehat{s}(\gamma, k) + 1)^{(k-1)\cdot T_{\mathcal{H}}(1-\gamma)}$.

(ii) By (5.15), the size of a set W in $\widehat{C}$ is exactly

$$T_{\mathcal{H}}(1-\gamma) \cdot \widehat{b}_{\mathcal{H}}(1-\gamma) \; = \; (\beta_{\mathcal{H}} - (1-\gamma))n,$$

as claimed.

(iii) For every $i \in [T_{\mathcal{H}}(1-\gamma)]$, we introduce the random variable $X_i \in \{0,1\}$

denoting the success in the i-th step. More precisely, $X_i = 1$ corresponds to the event of obtaining at least $\gamma \cdot \widehat{b}_{\mathcal{H}}(1-\gamma)$ vertices of a optimal vertex cover of $\mathcal{H}$ in the i-th iteration of algorithm $\mathcal{A}_{5.7}$. From Lemma 5.6.3, we know that this event happens with probability at least γ. The random variables define $T_{\mathcal{H}}(1-\gamma)$ independent Poisson trials. We introduce

$$X = \sum_{i \in [T_{\mathcal{H}}(1-\gamma)]} X_i. \tag{5.16}$$

Then, the expected value of the random variable X is given by

$$\mathbb{E}[X] \quad = \quad T_{\mathcal{H}}(1-\gamma)\cdot\gamma \quad = \quad \Psi_{\mathcal{H}}(n)\left(\frac{\beta_{\mathcal{H}}}{1-\gamma} - 1\right)\cdot\gamma. \tag{5.17}$$

Since we obtain at least $\gamma \cdot \widehat{b}_{\mathcal{H}}(1-\gamma)$ vertices in each step, the expected number of vertices of W' that are contained in C is at least

$$\begin{aligned} \mathbb{E}[X]\gamma \cdot \widehat{b}_{\mathcal{H}}(1-\gamma) &= \left[\Psi_{\mathcal{H}}(n)\left(\frac{\beta_{\mathcal{H}}}{1-\gamma} - 1\right)\gamma\right]\gamma\frac{(1-\gamma)\cdot n}{\Psi_{\mathcal{H}}(n)} \\ &\geq (\beta_{\mathcal{H}} - (1-\gamma))\gamma^2 n. \end{aligned}$$

By using Theorem 2.2.1, we conclude that for every $\delta \in (0,1)$, it yields at least $(1-\delta)(\beta_{\mathcal{H}} - (1-\gamma))\gamma^2 n$ vertices of C with probability at least

$$1 - \exp\left(-\Psi_{\mathcal{H}}(n)\left(\frac{\beta_{\mathcal{H}}}{1-\gamma} - 1\right)\gamma\frac{\delta^2}{2}\right),$$

as stated.

As for the running time, algorithm $\mathcal{A}_{5.7}$ generates a search tree of height $T_{\mathcal{H}}(1-\gamma)$ and fan-out less than $(\widehat{s}(\gamma,k)+1)^{k-1}$. At every node of the tree, the algorithm $\mathcal{A}_{5.6}$ is called taking

$$\mathrm{poly}(n) + \mathrm{O}([\widehat{s}(\gamma,k)]^{k-1})$$

time. Thus, the overall running time of the algorithm is $\mathrm{poly}(n) \cdot [\widehat{s}(\gamma,k)]^{\mathrm{O}((k-1)\cdot T_{\mathcal{H}}(1-\gamma))}$. Since we have

$$(k-1)T_{\mathcal{H}}(1-\gamma) \quad = \quad (k-1)\Psi_{\mathcal{H}}(n)\left(\frac{\beta_{\mathcal{H}}}{1-\gamma} - 1\right) \quad = \quad \mathrm{O}(\Psi_{\mathcal{H}}(n))$$

and $\widehat{s}(\gamma,k) = \mathrm{O}(1)$, the running time becomes

$$\mathrm{poly}(n)\exp\left[\mathrm{O}\left(\Psi_{\mathcal{H}}(n)\right)\right]$$

and the proof follows. ∎

Thus far, we are ready to give the proof of Theorem 5.6.7.

Proof of Theorem 5.6.7

In order to prove Theorem 5.6.7, we are going to analyze algorithm $\mathcal{A}_{5.8}$ given in Figure 5.8.

```
Algorithm $\mathcal{A}_{5.8}$
    Input  : $(\mathcal{H}, \varepsilon, r, k)$, where $\mathcal{H}$ is a $r$-nearly regular $k$-hypergraph
             and $\varepsilon > 0$ a constant.
    Output: A vertex cover $S$ of $\mathcal{H}$ .
    begin
        ① Set $\delta \leftarrow \min\left\{\frac{1}{4}, \frac{\varepsilon}{4k}\right\}$;
        ② Set $\gamma \leftarrow 1 - \frac{\beta_H}{8k \cdot \delta^{-2} + 1}$;
        ③ Let $\widehat{W} \leftarrow \mathcal{A}_{5.7}(\mathcal{H}, \widehat{b}_{\mathcal{H}}(1-\gamma), T_{\mathcal{H}}(1-\gamma), \gamma)$;
        foreach $W_i \in \widetilde{W}$ do
            ④ $Y \leftarrow \mathcal{A}_{5.2}(\mathcal{H}, W_i, k)$;
            ⑤ $\widehat{Y} \leftarrow \widehat{Y} \cup \{Y\}$;
        end
        return $S$ being the smallest set in $\widehat{Y}$;
    end
```

Figure 5.8: Algorithm $\mathcal{A}_{5.8}$

Let $\mathcal{H}$ be a r-nearly regular k-hypergraph containing n vertices and C a minimum vertex cover of $\mathcal{H}$. Furthermore, we choose a constant $\varepsilon > 0$. By letting

$$\delta = \min\left\{\frac{1}{4}, \frac{\varepsilon}{4k}\right\} \text{ and } \gamma = 1 - \frac{\beta_{\mathcal{H}}}{8k \cdot \delta^{-2} + 1}, \tag{5.18}$$

we have that $\gamma \in (1 - rk^{-2}, 1)$. Then, we run algorithm $\mathcal{A}_{5.7}$ on input $(\mathcal{H}, \widehat{b}_{\mathcal{H}}(1-\gamma), T_{\mathcal{H}}(1-\gamma), \gamma)$ with output $\widehat{W}$. According to Lemma 5.6.7,

$\widehat{W}$ contains a set W' such that

$$|W' \cap C| \;\geq\; (1-\delta)\gamma^2(\beta_{\mathcal{H}} - (1-\gamma))n \tag{5.19}$$

with probability at least

$$1 - \exp\left(-\Psi_{\mathcal{H}}(n)\left(\frac{\beta_{\mathcal{H}}}{1-\gamma} - 1\right)\gamma\frac{\delta^2}{2}\right) \;\geq\; \frac{3}{4}.$$

The right hand side in (5.19) can be bounded from below as follows.

$$\begin{aligned}
(1-\delta)\gamma^2(\beta_{\mathcal{H}} - (1-\gamma))n \;&\geq\; (1-\delta)\gamma^2\left(1 - \frac{1-\gamma}{\beta_{\mathcal{H}}}\right)\beta_{\mathcal{H}}n \\
&\geq\; (1-\delta)\gamma^2\left(1 - \frac{1}{8k\cdot\delta^{-2}+1}\right)\beta_{\mathcal{H}}n \\
&\geq\; (1-\delta)\left(1 - \frac{1}{8k\cdot\delta^{-2}+1}\right)^3\beta_{\mathcal{H}}n \\
&\geq\; (1-\delta)^4\beta_{\mathcal{H}}n \\
&\geq\; (1-4\delta)\beta_{\mathcal{H}}n
\end{aligned}$$

By Lemma 5.5.2, the algorithm $\mathcal{A}_{5.2}$ produces on input $(\mathcal{H}, W')$ a vertex cover S of $\mathcal{H}$ with approximation ratio at most

$$\begin{aligned}
\frac{k}{1 + \dfrac{|W'|}{n-k}} \;&\leq\; \frac{k}{1 + ((1-4\delta)k - 1)\beta_{\mathcal{H}}} \;&&\leq\; \frac{k}{1 + (k-1)\dfrac{r}{k} - 4\delta} \\
&\leq\; \frac{k}{1 + \dfrac{k-1}{k}r} + 4k\delta &&\leq\; \frac{k}{1 + \dfrac{k-1}{k}r} + \varepsilon.
\end{aligned}$$

Due to Lemma 5.6.7, we know that the running time of algorithm $\mathcal{A}_{5.8}$ is $\mathrm{poly}(n)\exp\big[\mathrm{O}\big(\Psi_{\mathcal{H}}(n)\big)\big]$. ■

5.6.4 Approximation Lower Bounds

In the previous section, we designed an approximation algorithm for the VERTEX COVER problem in r-nearly regular k-hypergraphs. The approximation ratio that was achieved by the algorithm is parametrized by the regularity of the input k-hypergraph, whereas its running time is parametrized by its density. In this section, we study the approximation hardness of the VERTEX COVER problem in nearly regular k-hypergraphs. In particular, we prove that it is UG-hard to obtain a better approximation upper bound than given in Theorem 5.6.7 for a large range of r-nearly regular k-hypergraphs. In addition, we give a **NP**-hardness of approximation result for the VERTEX COVER problem in nearly regular k-hypergraphs.

A Tight Inapproximability Result

We are going to prove the following hardness result.

Theorem 5.6.8
For every $k \geq 2$, it is UG*-hard to approximate the* VERTEX COVER *problem in r-nearly regular k-hypergraphs $\mathcal{H}$ having n vertices, maximum degree $\Delta_{\mathcal{H}} = \Omega(n^{\varepsilon})$ and $|E(\mathcal{H})| = o(n^k)$ to within any constant approximation ratio less than*

$$\frac{k}{1+(k-1)\dfrac{r}{k}}$$

for every $\varepsilon > 0$.

Proof of Theorem 5.6.8.
First, we concentrate on the case $r \in (0,1)$. Let $\mathcal{H}$ be the k-hypergraph from Theorem 5.4.2 with $k \geq 2$. According to Theorem 5.4.2, for every $\delta > 0$, the following cases (i) and (ii) are UG-hard to decide.

(i) Every vertex cover of $\mathcal{H}$ has size at least $|V(\mathcal{H})|\,(1-\delta)$.

(ii) The cardinality of a minimum vertex cover is at most

$$|V(\mathcal{H})|\left(\frac{1}{k}+\delta\right).$$

We will use $n = |V(\mathcal{H})|$. For a fixed $j \in \mathbb{N}$, we construct a new hypergraph $\mathcal{H}'$ consisting of $(1-r/k)n^j$ disjoint copies of $\mathcal{H}$ together with $n^j r/k$ disjoint k-uniform cliques of size $n+k-1$. Let us denote by $V_{\mathcal{H}}$ the set of vertices of the copies of $\mathcal{H}$ and $V_{\mathcal{K}}$ the set of vertices of the cliques. For notational simplicity, we introduce $N = |V(\mathcal{H}')| = |V_{\mathcal{H}}| + |V_{\mathcal{K}}| = n^{j+1} + o(n^{j+1})$. Note that the degrees of the vertices restricted to neighbors in $V_{\mathcal{H}}$ or $V_{\mathcal{K}}$ are at most

$$\binom{n+k-2}{k-1} = \mathrm{O}\left(n^{(k-1)}\right) = \mathrm{O}\left(N^{\frac{k-1}{j+1}}\right).$$

Accordingly, we can make $\mathcal{H}'$ to have $|E(\mathcal{H}')| = \omega\left(N^{\frac{k-1}{j+1}}\right)$ edges and maximum degree $\Delta_{\mathcal{H}'} = \omega(N^{\frac{k-1}{j+1}})$ by adding as many edges as needed with vertices in both $V_{\mathcal{H}}$ and $V_{\mathcal{K}}$ defining $E(\mathcal{H}')$. Using this construction, it is possible to construct edge sets with cardinality $o(n^{(j+1)k})$.

By definition, a vertex cover of $\mathcal{H}'$ must contain at least $n+k-1-(k-1) = n$ vertices of each clique. Thus, we need to include at least $n^{j+1}r/k$ vertices. Let us analyze the cases in the promise problem above. (i) If a vertex cover of $\mathcal{H}$ requires $n(1-\delta)$ vertices, then, we need at least

$$\left(1 - \frac{r}{k}\right) n^j \cdot n(1-\delta) \ = \ \left(1 - \frac{r}{k}\right) n^{j+1}(1-\delta)$$

additional vertices to cover all the copies. (ii) In the other case, we see that

$$\left(1 - \frac{r}{k}\right) n^{j+1} \left(\frac{1}{k} + \delta\right)$$

vertices suffice. Up to an $O(n^j)$ term, those vertices are sufficient to cover $\mathcal{H}'$. Consequently, for every $\delta > 0$, the following is UG-hard to decide.

(iii) Every vertex cover of $\mathcal{H}'$ has size at least

$$\left(1 - \frac{r}{k}\right) n^{1+j}(1-\delta) + \frac{n^{1+j}r}{k} + \mathrm{O}\left(n^j\right).$$

(iv) The cardinality of a minimum vertex cover of $\mathcal{H}'$ is at most

$$\left(1 - \frac{r}{k}\right) n^{1+j} \left(\frac{1}{k} + \delta\right) + \frac{n^{1+j}r}{k} + \mathrm{O}(n^j).$$

This implies the UG-hardness of approximating the VERTEX COVER problem to within any constant approximation ratio less than $R(\delta)$, where $R(\delta)$ is defined as follows.

$$R(\delta) = \frac{\left(1 - \frac{r}{k}\right) n^{1+j}(1-\delta) + \frac{n^{1+j}r}{k} + \mathrm{O}\left(n^j\right)}{\left(1 - \frac{r}{k}\right) n^{1+j} \left(\frac{1}{k} + \delta\right) + \frac{n^{1+j}r}{k} + \mathrm{O}(n^j)} \tag{5.20}$$

We are going to deduce a lower bound on the term $R(\delta)$ defined in (5.20).

$$\begin{aligned}
R(\delta) &\geq \frac{\left(1-\frac{r}{k}\right)n^{1+j}(1-\delta)+\frac{n^{1+j}r}{k}+\mathrm{O}\left(n^{j}\right)}{\left(1-\frac{r}{k}\right)n^{1+j}\left(\frac{1}{k}+\delta\right)+\frac{n^{1+j}r}{k}+\mathrm{O}\left(n^{j}\right)} \\
&= \frac{k\left(1-\frac{r}{k}\right)(1-\delta)+k\cdot\frac{r}{k}+\mathrm{O}\left(\frac{1}{n}\right)}{k\left(1-\frac{r}{k}\right)\left[\frac{1}{k}+\delta\right]+k\frac{r}{k}+\mathrm{O}\left(\frac{1}{n}\right)} \\
&= \frac{k-\delta k\left(1-\frac{r}{k}\right)+\mathrm{O}\left(\frac{1}{n}\right)}{1-\frac{r}{k}+r+\delta k\left(1-\frac{r}{k}\right)+\mathrm{O}\left(\frac{1}{n}\right)} \\
&\geq \frac{k-\delta k}{1-\frac{r}{k}+r+\delta k+o(1)} \\
&= \frac{k}{1+\frac{(k-1)r}{k}}-\left[\varepsilon(\delta)+o(1)\right]
\end{aligned}$$

The claimed inapproximability factor follows for the case $r \in (0,1)$. Moreover, we see that it suffices to add if necessary $o(n^{j+1})$ vertices to $V_{\mathcal{K}}$ together with the corresponding edges in order to obtain a regular k-hypergraph. Accordingly, it affects the inapproximability factor $R(\delta)$ only by a term of $o(1)$. ■

NP-Hardness of Approximation

Using the construction defined in the proof of the previous theorem, we are going to prove the following inapproximability result.

Theorem 5.6.9
For every $k \geq 3$, it is **NP***-hard to approximate the* VERTEX COVER *problem in r-nearly regular k-hypergraphs $\mathcal{H}$ having n vertices, maximum degree $\Delta_{\mathcal{H}} = \Omega(n^{\varepsilon})$ and $|E(\mathcal{H})| = o(n^{k})$ to within any constant approximation ratio less*

than

$$\frac{k-1}{1+(k-2)\dfrac{r}{k}}$$

for every $\varepsilon > 0$.

Proof of Theorem 5.6.9.

The claimed inapproximability factor will be proved by combining the construction from Theorem 5.6.8 together with the hard instance from Theorem 5.4.3. Recall from Theorem 5.4.3 that given a k-hypergraph $\mathcal{H}$ with $k \geq 3$, for every $\delta > 0$, the following cases (i) and (ii) are **NP**-hard to decide.

(i) Every vertex cover of $\mathcal{H}$ has size at least $|V(\mathcal{H})|\,(1-\delta)$.

(ii) The size of a minimum vertex cover of $\mathcal{H}$ is at most

$$|V(\mathcal{H})|\left(\frac{1}{k-1}+\delta\right).$$

Let us fix a constant $r \in (0,1)$ and an integer $j \geq 1$. The construction will be carried out as in Theorem 5.6.8. Let $\mathcal{H}'$ be the constructed k-hypergraph. Consequently, the following is **NP**-hard to decide.

(i) Every vertex cover of $\mathcal{H}'$ has size at least

$$\left(1-\frac{r}{k-1}\right)n^{1+j}(1-\delta)+\frac{n^{1+j}r}{k-1}+\mathrm{O}\left(n^j\right).$$

(ii) The size of a minimum vertex cover of $\mathcal{H}'$ is at most

$$\left(1-\frac{r}{k-1}\right)n^{1+j}\left(\frac{1}{k-1}+\delta\right)+\frac{n^{1+j}r}{k-1}+\mathrm{O}(n^j).$$

This implies the **NP**-hardness of approximating the VERTEX COVER problem to within any constant approximation ratio less than $R(\delta)$, where $R(\delta)$ is defined as follows.

$$R(\delta)=\frac{\left(1-\dfrac{r}{k-1}\right)n^{1+j}(1-\delta)+\dfrac{n^{1+j}r}{k-1}+\mathrm{O}\left(n^j\right)}{\left(1-\dfrac{r}{k-1}\right)n^{1+j}\left(\dfrac{1}{k-1}+\delta\right)+\dfrac{n^{1+j}r}{k-1}+\mathrm{O}(n^j)} \tag{5.21}$$

Let us deduce a lower bound on the term $R(\delta)$ in (5.21). We have that

$$\begin{aligned}
R(\delta) &= \frac{\left(1-\frac{r}{k-1}\right)n^{1+j}(1-\delta)+\frac{n^{1+j}r}{k-1}+\mathrm{O}\left(n^j\right)}{\left(1-\frac{r}{k-1}\right)n^{1+j}\left(\frac{1}{k-1}+\delta\right)+\frac{n^{1+j}r}{k-1}+\mathrm{O}\left(n^j\right)}\\
&= \frac{(k-1)\left(1-\frac{r}{k-1}\right)(1-\delta)+(k-1)\cdot\frac{r}{k-1}+\mathrm{O}\left(\frac{1}{n}\right)}{(k-1)\left(1-\frac{r}{k-1}\right)\left(\frac{1}{k-1}+\delta\right)+(k-1)\frac{r}{k-1}+\mathrm{O}\left(\frac{1}{n}\right)}\\
&= \frac{k-1-\delta(k-1)\left(1-\frac{r}{k-1}\right)+\mathrm{O}\left(\frac{1}{n}\right)}{1-\frac{r}{k-1}+\delta(k-1)\left(1-\frac{r}{k-1}\right)+r+\mathrm{O}\left(\frac{1}{n}\right)}\\
&\geq \frac{k-1-\delta(k-1)}{1-\frac{r}{k-1}+\delta(k-1)\left(1-\frac{r}{k-1}\right)+r+o(1)}\\
&= \frac{k-1}{1+\frac{(k-2)r}{k-1}}-[\varepsilon(\delta)+o(1)],
\end{aligned}$$

as claimed. In order to prove the case $(r=1)$, we may argue as in the proof of Theorem 5.6.8. ■

5.7 Bibliographic Notes

The material presented in this chapter is based on the paper [CKSV12], which contains the proofs of the Theorem 5.6.7, 5.6.8, 5.6.9, 5.5.7 and 5.5.9.

CHAPTER 6

Vertex Cover of k-Partite k-Hypergraphs

In this chapter, we investigate the approximability of the VERTEX COVER problem in dense and nearly regular k-partite k-hypergraphs.

We first consider the VERTEX COVER problem in dense k-balanced hypergraphs and prove that the problem is efficiently approximable with approximation ratio better than $k/2$. After that, we develop an improved technique for the extraction of a minimum vertex cover of a dense k-partite k-hypergraph by which we obtain an efficient approximation algorithm for the VERTEX COVER problem in dense k-partite k-hypergraphs with better approximation ratio. On the approximation hardness side, we propose a conjecture about the UG-hardness of the VERTEX COVER problem in k-partite k-hypergraphs. Assuming this conjecture, we prove an optimal inapproximability result for the dense version of the problem.

By combining the framework developed for the VERTEX COVER problem in nearly regular k-hypergraphs with a new method called randomized bucketing extraction, we design a randomized approximation algorithm for the VERTEX COVER problem in nearly regular k-partite k-hypergraphs with approximation ratio strictly less than $k/2$ and running time depending on the density of the underlying k-hypergraph. In particular, it entails the existence of quasi-polynomial and polynomial time approximation algorithms with approximation ratio less than $k/2$ for mildly sparse and subdense instances, respectively. On the other hand, we prove the optimality of the approximation ratio achieved by our algorithm based on the conjecture mentioned above.

6.1 Introduction

When the vertex set of a k-hypergraph is partitioned into k disjoint sets such that every edge contains exactly one vertex of every set of the k-partition and the k-partition is given as a part of the input, this restricted version is called the VERTEX COVER problem in k-partite k-hypergraphs.

The underlying problem has been investigated for applications connected to database problems including distributed data mining [FMO$^+$03], schema mapping discovery [GS10a] and optimization of finite automata [ISY05]. In bipartite graphs ($k = 2$), the size of a minimum vertex cover can be computed efficiently by finding a maximum matching in the same graph due to König's Theorem. On the other hand, for general k, Gottlob and Senellart [GS10a] proved that the VERTEX COVER problem in k-partite k-hypergraphs is **NP**-hard by constructing a reduction from the 3SAT problem. Since the reduction given in [GS10a] is also approximation preserving, it implies the **APX**-hardness of the problem. Lovasz [L75] studied the the VERTEX COVER problem restricted to k-partite k-hypergraphs and proved that the integrality gap of the natural linear programming relaxation is bounded by $k/2$. In more detail, he constructed a rounding scheme for the solution of the natural linear programming relaxation implying an efficient approximation algorithm for the problem with approximation ratio $k/2$. Aharoni, Holzman and Krivelevich [AHK96] constructed a family of tight examples witnessing that the integrality gap of the natural linear programming relaxation is $k/2 - o(1)$.

Based on the inapproximability result given by Dinur, Guruswami, Khot and Regev [DGKR05] for the VERTEX COVER problem in k-hypergraphs, Guruswami and Saket [GS10b] constructed a reduction from the former mentioned problem implying that for all $k \geq 5$, it is **NP**-hard to approximate the VERTEX COVER problem in k-partite k-hypergraphs to within any constant approximation ratio less than $k/4$. By applying the result of Kumar, Manokaran, Tulsiani and Vishnoi [KMTV11] with a modification to the LP integrality gap construction due to Ahorani, Holzman and Krivelevich [AHK96], Guruswami and Saket [GS10b] gave an optimal inapproximability result for the problem. More precisely, they proved that for all $k \geq 3$, it is UG-hard to approximate the VERTEX COVER problem in k-partite k-hypergraphs to within any constant approximation ratio less than $k/2$. Combining Long Code based gadgets with the Multi-Layered PCP constructed by Dinur, Guruswami, Khot and Regev [DGKR05] as starting point of their reduction, Sachdeva and Saket [SS11] gave a nearly optimal **NP**-hardness of approximation result for the problem. They proved that for all $k \geq 4$, it is **NP**-hard to approximate the VERTEX COVER problem in k-partite k-hypergraphs to within any constant approximation ratio less than $\big(k/2 - 1 + (2k)^{-1}\big)$.

Dense k-partite k-hypergraphs

To the best of the author's knowledge, this is the first result concerning the approximability of the VERTEX COVER problem in dense k-partite k-hypergraphs.

First, we consider the VERTEX COVER problem in dense k-balanced hypergraphs and prove that a similar approach designed for the VERTEX COVER problem in dense k-hypergraphs yields an efficient approximation algorithm for the VERTEX COVER problem in dense k-balanced hypergraphs with an approximation ratio better than $k/2$. After that, we develop an improved technique for the extraction of a minimum vertex cover of a dense k-partite k-partite by which we obtain an efficient approximation algorithm for the VERTEX COVER problem in dense k-partite k-hypergraphs with an approximation ratio better than $k/2$. It achieves a better approximation ratio and is applicable to a whole class of dense k-partite k-hypergraphs. As for approximation lower bounds, we propose a conjecture about the UG-hardness of the VERTEX COVER problem in k-partite k-hypergraphs. Assuming this conjecture, we prove an optimal inapproximability result for the dense version of the problem.

Nearly Regular k-partite k-hypergraphs

We investigate the approximability of the VERTEX COVER problem in nearly regular k-partite k-hypergraphs. By combining the framework developed in Section 5.6 for the VERTEX COVER problem in nearly regular k-hypergraphs with a new method called Randomized Bucketing Extraction, we design an randomized approximation algorithm for the VERTEX COVER problem in nearly regular k-partite k-hypergraphs with approximation ratio strictly less than $k/2$ and running time depending on the density of the underlying k-hypergraph. In particular, it entails the existence of quasi-polynomial and polynomial time approximation algorithms with approximation ratio less than $k/2$ for mildly sparse instances and for subdense instances, respectively. On the other hand, we prove the optimality of the approximation ratio achieved by our algorithm based on the conjecture mentioned above.

6.2 Outline of this Chapter

This chapter is organized as follows. In Section 6.4, we survey some of the known results concerning the approximability of the VERTEX COVER problem in k-partite k-hypergraphs. In Section 6.5, we study the VERTEX COVER problem restricted to dense k-balanced hypergraphs and dense k-partite k-hypergraphs. Finally, in Section 6.6, we investigate the approximability of the VERTEX COVER problem in nearly regular k-partite k-hypergraphs.

6.3 Preliminaries

We are going to introduce some notation used in this chapter. Let us first define the underlying problem according to Definition 4.1.1.

Definition 6.3.1 (VERTEX COVER problem in k-partite k-hypergraphs)

Instances: *A k-partite k-hypergraph $\mathcal{H}$ with given vertex partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$*

Solutions: *Subset $C \subseteq V(\mathcal{H})$ such that $C \cap e \neq \varnothing$ for all $e \in E(\mathcal{H})$*

Task: *Minimize the cardinality of C*

If the k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$ of a k-partite k-hypergraph $\mathcal{H}$ is given as a part of the input, we assume that $|V_{i+1}(\mathcal{H})| \leq |V_i(\mathcal{H})|$ for all $i \in [k-1]$. Furthermore, we say that $\mathcal{H}$ is k-balanced if the k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$ of $\mathcal{H}$ satisfies the property $|V_j(\mathcal{H})| = |V_i(\mathcal{H})|$ for all $i, j \in [k]$.

Given a k-partite k-hypergraph $\mathcal{H}$ with k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$ and $k \geq 2$, for each integer $j \in [k]$ and each vertex $v \in V_j(\mathcal{H})$, we define the v-induced $(k-1)$-partite $(k-1)$-hypergraph $\mathcal{H}(v)$ by $V\big(\mathcal{H}(v)\big) = \bigcup_{i \in [k] \setminus \{j\}} V_i(\mathcal{H})$ and $E\big(\mathcal{H}(v)\big) = \big\{e \setminus \{v\} \mid v \in e \in E(\mathcal{H})\big\}$.

In the following, we introduce classes of k-partite k-hypergraphs with respect to the parameter $\Psi_{\mathcal{H}}(n)$ which is defined for a k-partite k-hypergraph

$\mathcal{H}$ with n vertices and given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$ as follows.

$$\Psi_{\mathcal{H}}(n) = \left(\prod_{i\in[k]} |V_i|\right)\Big(|E(\mathcal{H})|\Big)^{-1}$$

We refer to a k-partite k-hypergraph $\mathcal{H}$ with n vertices as *dense*, *subdense*, *mildly sparse* and *non-dense* if we have $\Psi_{\mathcal{H}}(n) = \mathrm{O}(1)$, $\Psi_{\mathcal{H}}(n) = \mathrm{O}\big(\log n\big)$, $\Psi_{\mathcal{H}}(n) = \mathrm{poly}(\log n)$ and $\Psi_{\mathcal{H}}(n) = \omega(1)$, respectively.

For dense k-partite k-hypergraphs, we introduce even finer-grained classes. A k-partite k-hypergraph $\mathcal{H}$ with given vertex partition $\{V_i \mid i \in [k]\}$ is called (ε, ℓ)-*dense* for an integer $\ell \in [k-1] \cup \{0\}$ and a constant $\varepsilon \in (0,1)$ if there exists a set

$$I \in \binom{[k]}{\ell} \text{ s. t. } \forall S \in \left\{Y \in \binom{V(\mathcal{H})}{\ell} \,\middle|\, |V_i \cap Y| = 1 \forall i \in I\right\} : d_{\mathcal{H}}(S) \geq \varepsilon \cdot \prod_{i\in[k]\setminus I} |V_i|$$

holds.

6.4 The General Problem

Before we investigate the approximability of the VERTEX COVER problem in dense and nearly regular k-partite k-hypergraphs, we survey some of the known results concerning the approximability of the VERTEX COVER problem in k-partite k-hypergraphs.

6.4.1 An Approximation Upper Bound

First of all, we present a generalization of the König Theorem due to Lovász [L75] implying a rounding procedure that constructs efficiently a vertex cover of a given k-partite k-hypergraph with approximation ratio at most $k/2$. The rounding procedure uses the fractional solution of the natural linear programming relaxation $\mathsf{LP}_{6.1}$ given in Figure 6.1. Afterwards, we describe the integrality gap construction of Aharoni, Holzman and Krivelevich [AHK96] for the $\mathsf{LP}_{6.1}$. More precisely, they constructed a family of k-partite k-hypergraphs yielding an integrality gap of $k/2 - o(1)$.

$$
\begin{array}{ll}
\text{Minimize} & \displaystyle\sum_{v\in V(\mathcal{H})} x_v \\
\text{Subject to} & \\
 & \displaystyle\sum_{v\in e} x_v \geq 1 \quad \forall e \in E(\mathcal{H}) \\
 & 0 \leq x_v \leq 1 \quad \forall v \in V(\mathcal{H})
\end{array}
$$

Figure 6.1: Linear program $\mathsf{LP}_{6.1}$

In his doctoral thesis, Lovász [L75] studied the VERTEX COVER problem in k-partite k-hypergraphs and proved the following result.

Theorem 6.4.1 ([L75])
Let $\mathcal{H}$ be a k-partite k-hypergraph with $k \geq 3$, $\tau^*(\mathcal{H})$ the value of a fractional solution to the corresponding linear program $\mathsf{LP}_{6.1}$ and $\tau(\mathcal{H})$ the cardinality of a minimum vertex cover of $\mathcal{H}$. Then, the following inequality holds.

$$\frac{\tau(\mathcal{H})}{\tau^*(\mathcal{H})} \leq \frac{k}{2}$$

We are going to present the proof of Theorem 6.4.1. For this reason, we introduce some notation that will be used in the proof.

For every odd integer $m \geq 1$, we define the $3 \times m$ matrix $A(m) = (a_{ij})_{i\in[3],j\in[m]}$ as follows.

$$
A(m) = \begin{pmatrix}
1 & \frac{m+3}{2} & 2 & \frac{m+5}{2} & 3 & \dots & \frac{m-1}{2} & m & \frac{m+1}{2} \\
\frac{m+1}{2} & 1 & \frac{m+3}{2} & 2 & \frac{m+5}{2} & \dots & m-1 & \frac{m-1}{2} & m \\
m & m-1 & m-2 & m-3 & m-4 & \dots & 3 & 2 & 1
\end{pmatrix}
$$

Furthermore, for every integer $m \geq 1$, we introduce the $2 \times m$ matrix $B(m)$

defined as follows.

$$B(m) = \begin{pmatrix} 1 & 2 & 3 & \dots & m-2 & m-1 & m \\ m & m-1 & m-2 & \dots & 3 & 2 & 1 \end{pmatrix}$$

By combining the former introduced matrices $A(m)$ and $B(m)$, we are going to define the $k \times m$ matrix $C(k,m)$ for every odd integer $m \geq 1$ and every $k \in \mathbb{N}$.

Definition 6.4.1 (Matrix $C(k,m)$)
Let $m \geq 1$ be an odd integer. For every even integer $k \geq 2$, the $k \times m$ matrix $C(k,m) = (c_{ij})_{i\in[k],j\in[m]}$ is given as follows.

$$c_{ij} = \begin{cases} b_{1j} & \text{for even } i \text{ and } j \in [m] \\ b_{2j} & \text{for odd } i \text{ and } j \in [m] \end{cases} \tag{6.1}$$

For every odd integer $k \geq 3$, we define $C(k,m) = (c_{ij})_{i\in[k],j\in[m]}$ as follows.

$$c_{ij} = \begin{cases} a_{ij} & \text{for } i \in [3] \text{ and } j \in [m] \\ b_{1j} & \text{for even } i \geq 4 \text{ and } j \in [m] \\ b_{2j} & \text{for odd } i \geq 5 \text{ and } j \in [m] \end{cases} \tag{6.2}$$

Before we prove Theorem 6.4.1, we give an example of such a matrix.

Example 6.4.1 (Matrix $C(5,9)$)

$$C(5,9) = \begin{pmatrix} 0 & 5 & 1 & 6 & 2 & 7 & 3 & 8 & 4 \\ 4 & 0 & 5 & 1 & 6 & 2 & 7 & 3 & 8 \\ 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{pmatrix}$$

Let us now turn to the proof of Theorem 6.4.1.

Proof of Theorem 6.4.1.
Let $\mathcal{H}$ be a k-partite k-hypergraph with given vertex partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$ and $k \geq 3$, $\tau^*(\mathcal{H})$ the value of an optimal solution of $\mathsf{LP}_{6.1}$ for $\mathcal{H}$ and

$\tau(\mathcal{H})$ the cardinality of a minimum vertex cover of $\mathcal{H}$. Since $\mathsf{LP}_{6.1}$ has only integral coefficients, there exists an optimal solution $x^* : V(\mathcal{H}) \to [0,1]$ of $\mathsf{LP}_{6.1}$ such that $x^*(v)$ is rational for every $v \in V(\mathcal{H})$. Let us define the integer f by

$$f \;=\; \min\{n \in \mathbb{N} \mid (n \cdot x^*(v)) \in \mathbb{Z} \text{ for all } v \in V(\mathcal{H})\}\,. \tag{6.3}$$

Furthermore, we introduce $m = 2f - 1$. Notice that for all $k \geq 2$ and all odd integer $m \geq 1$, the matrix $C(k,m)$ has the following properties.

(*i*) Every row of $C(k,m)$ is a permutation of the numbers contained in the set $[m]$.

(*ii*) The sum of every column of $C(k,m)$ is bounded from above by

$$\frac{k \cdot (m+1)}{2}\,. \tag{6.4}$$

For every $j \in [m]$, we introduce the subset C_j of $V(\mathcal{H})$ as follows.

$$C_j \;=\; \bigcup_{i \in [k]} \{v \in V_i \mid kf \cdot x^*(v) \geq c_{ij}\} \tag{6.5}$$

We claim that for every $j \in [m]$, C_j is a vertex cover of $\mathcal{H}$. For the sake of contradiction, suppose that there exists an edge $e' = \{v_1, ..., v_k\} \in E(\mathcal{H})$ and an integer $l \in [m]$ such that $e' \cap C_l = \varnothing$ holds. By definition of C_l, we have that $kf \cdot x^*(v_i) < c_{il}$ for every $i \in [k]$. It implies that

$$\sum_{i \in [k]} x^*(v_i) \;<\; \sum_{i \in [k]} \frac{c_{il}}{kf} \;\leq\; \frac{\frac{k(m+1)}{2}}{kf} \;\leq\; \frac{m+1}{m+1} \;=\; 1 \tag{6.6}$$

contradicting the definition of x^*.

Since each row of the matrix $C(k,m)$ is a permutation and by (6.5), we have $|\{j \in [m] \mid v \in C_j\}| \;\leq\; kf \cdot x^*(v)$ for every $v \in V(\mathcal{H})$. Consequently, we deduce the following bound on the sum of the sizes of the sets C_j.

$$\sum_{j \in [m]} |C_j| \;\leq\; \sum_{v \in V(\mathcal{H})} \frac{k}{2}(m+1) \cdot x^*(v) \;=\; \frac{k}{2}(m+1) \cdot \tau^*(\mathcal{H}) \tag{6.7}$$

Since every C_j with $j \in [m]$ is a vertex cover of $\mathcal{H}$, we obtain the following.

$$\begin{aligned} \min\{|C_j| \mid j \in [m]\} \quad &\le \quad \sum_{j\in[m]} \frac{|C_j|}{m} \quad \le \quad \frac{\frac{k}{2}(m+1)\tau^*(\mathcal{H})}{m} \\ &= \quad \frac{k}{2}\tau^*(\mathcal{H})\left(1+\frac{1}{m}\right) \end{aligned}$$

By letting m tend to infinity, the proof of Theorem 6.4.1 follows. ■

As we will see, the proof of Theorem 6.4.1 implies the existence of an efficient approximation algorithm that on input of a k-partite k-hypergraph $\mathcal{H}$ and its k-partition constructs a vertex cover with approximation ratio $k/2$. Given the fractional solution $x^* : V(\mathcal{H}) \to [0,1]$ of $\mathsf{LP}_{6.1}$, $n = |V(\mathcal{H})|$ and a positive integer t, we introduce the upscaled function $x^t : V(\mathcal{H}) \to [0,1]$ defined by

$$x^t(v) = \big\lceil\, (x^*(v)\cdot n \cdot t \cdot 3k)\, \big\rceil (3k\cdot t\cdot n)^{-1} \text{ for every } v \in V(\mathcal{H}). \tag{6.8}$$

Furthermore, we set $f = 3k\cdot n\cdot t$ in (6.3) and obtain $m = 6k\cdot n\cdot t + 1$. For each $j \in [m]$, we define the set $C_j^t = \bigcup_{i\in[k]} \{v \in V_i(\mathcal{H}) \mid k\cdot f\cdot x^t(v) \ge c_{ij}\}$. We present the approximation algorithm for the VERTEX COVER problem in k-partite k-hypergraphs given in Figure 6.2.

```
Algorithm $\mathcal{A}_{6.2}$

  Input  : $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\})$, where $\mathcal{H}$ is a $k$-partite
           $k$-hypergraph and $\{V_i(\mathcal{H}) \mid i \in [k]\}$ its $k$-partition.
  Output: A vertex cover $S$ of $\mathcal{H}$.

  begin
     ① Compute the fractional solution $x^*$ of $\mathsf{LP}_{6.1}$ for $\mathcal{H}$;
     ② Compute $\widetilde{C} = \{C_j^t \mid j \in [6n\cdot k\cdot t+1]\}$ with $t = 2n^2$;
     ③ Let $C_{\min}$ be the smallest set in $\widetilde{C}$ ;
     return $C_{\min}$;
  end
```

Figure 6.2: Algorithm $\mathcal{A}_{6.2}$

By using the construction given in the proof of Theorem 6.4.1, we are going to prove the following statement.

Corollary 6.4.1
Let $\mathcal{H}$ be a k-partite k-hypergraph with $k \geq 3$ and $\{V_i(\mathcal{H}) \mid i \in [k]\}$ its k-partition. Then, on input $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\})$, algorithm $\mathcal{A}_{6.2}$ constructs in polynomial time a vertex cover of $\mathcal{H}$ with approximation ratio at most $k/2$.

Proof of Corollary 6.4.1.
Let $\mathcal{H}$ be a k-partite k-hypergraph with $k \geq 3$ containing n vertices and $\{V_i(\mathcal{H}) \mid i \in [k]\}$ its k-partition. Furthermore, let x^* be the fractional solution of $\mathsf{LP}_{6.1}$ for $\mathcal{H}$. Let us fix a $t \in \mathbb{N}$. Note that for all $t \in \mathbb{N}$ and $v \in V(\mathcal{H})$, we have that $x^*(v) \leq x^t(v)$. By (6.6), we deduce that for every $j \in m$, the set C_j^t is a vertex cover of $\mathcal{H}$. By (6.7), we obtain the following bound on the sum of the sizes of the sets C_j^t.

$$\sum_{j \in [m]} |C_j^t| \;\leq\; \sum_{v \in V(\mathcal{H})} \frac{k}{2}(m+1) \cdot x^t(v) \;\leq\; \frac{k}{2}(m+1) \cdot \tau^*(\mathcal{H}) + n \cdot \frac{k(m+1)}{2(3n \cdot k \cdot t)}$$

Let C be a minimum vertex cover of $\mathcal{H}$ and $C_{\min}$ the set that was constructed by algorithm $\mathcal{A}_{6.2}$ on input $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\})$. Since $\tau^*(\mathcal{H})$ is a lower bound on the cardinality of $|C|$, the size of $C_{\min}$ can be bounded from above as follows.

$$|C_{\min}| \;\leq\; \sum_{j \in [m]} \frac{|C_j^t|}{m} \;\leq\; \frac{k}{2}\frac{m+1}{m} \cdot \tau^*(\mathcal{H}) + \frac{m+1}{6 \cdot t \cdot m} \;\leq\; |C|\frac{m+1}{m}\left(\frac{k}{2} + \frac{1}{6t}\right).$$

Therefore, we obtain the following bound on the approximation ratio of algorithm $\mathcal{A}_{6.2}$.

$$\frac{|C_{\min}|}{|C|} \;\leq\; \frac{k}{2} + \frac{1}{m}\left(\frac{k}{2} + \frac{1}{6t}\right) + \frac{1}{6t} \;\leq\; \frac{k}{2} + \frac{1}{t}$$

Since we have $|C_{\min}|, |C| \in \mathbb{N}$, $|C| \leq n$ and $t = 2n^2$, we conclude that $|C_{\min}| \leq (k/2) \cdot |C|$. Finally, we note that x^* can be computed efficiently and the proof of Corollary 6.4.1 follows. ■

Next, we will see that the bound given in Theorem 6.4.1 is tight.

Integrality Gap of $\mathsf{LP}_{6.1}$

We are going to describe the integrality gap construction of Aharoni, Holzman and Krivelevich [AHK96] for the linear program $\mathsf{LP}_{6.1}$ and present the proof of the following theorem.

Theorem 6.4.2 ([AHK96])
For every $k \geq 3$, the integrality gap of $\mathsf{LP}_{6.1}$ *is* $\left(\frac{k}{2} - o(1)\right)$.

Proof of Theorem 6.4.2.
Given a positive integer n, we are going to construct a k-partite k-hypergraph $\mathcal{H}$, for which the value of the LP solution yields an integrality gap of

$$\frac{nk}{2(n+1)} = \frac{k}{2}\left(1 - \frac{k}{2(n+1)}\right) = \frac{k}{2} - o(1). \tag{6.9}$$

Let us start with the description of $\mathcal{H}$. The vertex set $V(\mathcal{H})$ of $\mathcal{H}$ is partitioned into $V_1(\mathcal{H}), \ldots, V_k(\mathcal{H})$, where

$$V_i(\mathcal{H}) = \{y^1_{ij} \mid j \in [n]\} \cup \{y^2_{il} \mid l \in [nk+1]\}. \tag{6.10}$$

Before we specify the edge set of $\mathcal{H}$, we define the LP solution. The LP values of the vertices of $\mathcal{H}$ are given by $x^* : V(\mathcal{H}) \to [0,1]$ with

$$\begin{aligned} x^*(y^1_{ij}) &= \frac{2j}{nk}, \quad \text{for all } j \in [n] \\ x^*(y^2_{il}) &= 0, \qquad \text{for all } l \in [nk+1]. \end{aligned}$$

The set of edges of $\mathcal{H}$ is defined as the set of all possible edges with exactly one vertex from each $V_i(\mathcal{H})$ and the property that the sum of the LP values of the corresponding vertices is at least 1. Formally, we have

$$E(\mathcal{H}) = \left\{ e \subseteq V(\mathcal{H}) \;\middle|\; \forall i \in [k] : \; |e \cap V_i(\mathcal{H})| = 1 \text{ and } \sum_{v \in e} x^*(v) \geq 1 \right\}. \tag{6.11}$$

Accordingly, we see that $\mathcal{H}$ is k-partite with $\{V_i(\mathcal{H}) \mid i \in [k]\}$ being the k-partition of $V(\mathcal{H})$. The value of the corresponding LP solution is

$$\sum_{v \in V} x^*(v) = k \sum_{j \in [n]} \frac{2j}{nk} = n + 1. \tag{6.12}$$

Let $\widetilde{V}$ be a minimum vertex cover of $\mathcal{H}$. In order to bound the size of the minimum vertex cover from below, we note that the set

$$\{v \in V(\mathcal{H}) \mid x^*(v) > 0\}$$

is a vertex cover of $\mathcal{H}$ with size nk. Thus, we have $|\widetilde{V}| \leq nk$. Notice that for any $i \in [k]$, the vertices in $\{y_{il}^2 \mid l \in [nk+1]\}$ have the same neighborhood. Consequently, we may assume that $y_{il} \notin \widetilde{V}$, as otherwise, $\widetilde{V}$ would include at least $nk+1$ such vertices.

For all $i \in [k]$, we introduce the index $j(i) \in [n] \cup \{0\}$ given by

$$j(i) = \begin{cases} 0 & \text{if for all } j \in [n],\ x_{ij} \in \widetilde{V}, \\ \max\ \{j \in [n] \mid x_{ij} \notin \widetilde{V}\} & \text{otherwise.} \end{cases} \tag{6.13}$$

Since $\widetilde{V}$ is a vertex cover of $\mathcal{H}$ and by definition of the indices $j(i)$, we have

$$\sum_{i\in[k]} x^*(y_{i\,j(i)}^1) \ < \ 1 \text{ implying } \sum_{i\in[k]} j(i) \ < \ \frac{nk}{2}.$$

On the other hand, it yields a lower bound on the cardinality of $\widetilde{V}$ given by

$$\sum_{i\in[k]} (n - j(i)).$$

Combining the deduced facts, we attain

$$|V'| \ \geq \ \sum_{i\in[k]} [n - j(i)] \ \geq \ nk - \sum_{i\in[k]} j(i) \ \geq \ nk - \frac{nk}{2} \ = \ \frac{nk}{2}. \tag{6.14}$$

By (6.12) and (6.14), we obtain the claimed integrality gap of $\mathsf{LP}_{6.1}$ in (6.9) and the proof of Theorem 6.4.2 follows. ■

6.4.2 Approximation Lower Bounds

Next, we are going to state some known hardness of approximation results for the VERTEX COVER problem in k-partite k-hypergraphs. We start with an inapproximability result due to Guruswami and Saket [GS10b].

UG-Hardness

By applying the techniques of Kumar, Manokaran, Tulsiani and Vishnoi [KMTV11] with a modification to the LP integrality gap construction due to Ahorani, Holzman and Krivelevich [AHK96], Guruswami and Saket [GS10b] gave an optimal inapproximability result for the VERTEX COVER problem in k-partite k-hypergraphs. More precisely, they proved that for all $k \geq 3$, it is UG-hard to approximate the problem to within any constant approximation ratio less than $k/2$. Let us present the precise statement.

Theorem 6.4.3 ([GS10b])
Let $\mathcal{H}$ be a k-partite k-hypergraph with given vertex partition and $k \geq 3$. For every $\delta > 0$, the following is UG-hard to decide.

- *Every vertex cover of $\mathcal{H}$ has size at least*

$$|V(\mathcal{H})|\left(\frac{1}{2(k-1)} - \delta\right).$$

- *The size of an optimal vertex cover of $\mathcal{H}$ is at most*

$$|V(\mathcal{H})|\left(\frac{1}{k(k-1)} + \delta\right).$$

Next, we are going to state the inapproximability result for the VERTEX COVER problem in k-partite k-hypergraphs due to Sachdeva and Saket [SS11] based on a weaker assumption.

NP-Hardness of Approximation

Sachdeva and Saket [SS11] obtained a nearly optimal **NP**-hardness of approximation result for the VERTEX COVER problem in k-partite k-hypergraphs. It implies that it is **NP**-hard to approximate the VERTEX COVER problem in k-partite k-hypergraphs to within any constant approximation ratio less than $\left(k/2 - 1 + (2k)^{-1}\right)$. In more detail, they proved the following theorem.

Theorem 6.4.4 ([SS11])
Let $\mathcal{H}$ be a k-partite k-hypergraph with given vertex partition and $k \geq 4$. For every $\delta > 0$, the following is **NP***-hard to decide.*

(*i*) *Every vertex cover of $\mathcal{H}$ has size at least*

$$|V(\mathcal{H})|\left(\frac{k-1}{2k\,[2k+1]} - \delta\right).$$

(*ii*) *The cardinality of a minimum vertex cover of $\mathcal{H}$ is at most*

$$|V(\mathcal{H})|\left(\frac{1}{(k-1)\big(2k+1\big)} + \delta\right).$$

6.5 Dense k-Partite k-Hypergraphs

According to Theorem 6.4.3, it is UG-hard to approximate the VERTEX COVER problem in k-partite k-hypergraphs to within any constant approximation ratio less than $k/2$. In this section, we investigate the approximability of the dense version of the problem and design an approximation algorithm for the problem with approximation ratio strictly less than $k/2$. On the other hand, we give an optimal inapproximability result.

6.5.1 Our Contribution

By extending the approach for the VERTEX COVER problem in dense k-hypergraphs in Section 5.5.3, we design an efficient approximation algorithm for the VERTEX COVER problem in dense k-balanced hypergraphs with approximation ratio better than $k/2$. Let us formulate our first result.

Theorem 6.5.1
There is a polynomial time approximation algorithm for the VERTEX COVER *problem in (ε, ℓ)-dense k-balanced hypergraphs with approximation ratio*

$$\frac{k}{k-(k-2)(1-\varepsilon)^{\frac{1}{k-\ell}}}.$$

Afterwards, we develop a technique for the extraction of a minimum vertex cover of given k-partite k-hypergraphs. In particular, we design an extraction algorithm that on input of a dense k-partite k-hypergraph $\mathcal{H}$ and its vertex partition constructs a large part of a minimum vertex cover of $\mathcal{H}$. More precisely, we obtain the following result.

Lemma 6.5.1
Let $\mathcal{H}$ be an ε-dense k-partite k-hypergraph with given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$ and C a minimum vertex cover of $\mathcal{H}$. There is an algorithm that on input $\mathcal{H}$ and its k-partition computes in polynomial time a collection $\widetilde{M}$ of subsets $W_i \subseteq V(\mathcal{H})$ such that the size of $\widetilde{M}$ is polynomial in $|V(\mathcal{H})|$ and $\widetilde{M}$ contains a set $S_C \subseteq C$ with $|S_C| \geq \varepsilon \cdot |V_k(\mathcal{H})|$.

As a consequence, we obtain a lower bound on the size of a minimum vertex cover of a given dense k-partite k-hypergraph.

Corollary 6.5.1
Let $\mathcal{H}$ be an ϵ-dense k-partite k-hypergraph and s the cardinality of its smallest partition. Then, the size of a minimum vertex cover of $\mathcal{H}$ is bounded from below by $(\varepsilon \cdot s)$.

In addition, we construct a family of k-hypergraphs, for which the bound in Lemma 6.5.1 is tight.

The mentioned extraction algorithm will play a key role in our improved approximation algorithm for the VERTEX COVER problem in dense k-partite k-hypergraphs. On the one hand, it achieves a better approximation ratio compared to the approximation algorithm given in Theorem 6.5.1. On the other hand, it is applicable to k-partite k-hypergraphs with given vertex partition. Let us formulate the precise statement.

Theorem 6.5.2
There is a polynomial time approximation algorithm for the VERTEX COVER *problem in ε-dense k-partite k-hypergraphs with approximation ratio*

$$\frac{k}{2 + (k-2)\varepsilon}\,.$$

As for approximation lower bounds, we give an UG-hardness and a **NP**-hardness of approximation result for the VERTEX COVER problem in dense k-partite k-hypergraphs. In particular, we obtain the following hardness result.

Theorem 6.5.3
For every $k \geq 3$, it is UG*-hard to approximate the* VERTEX COVER *problem in (ε, ℓ)-dense k-partite hypergraphs to within any constant approximation ratio less than*

$$\frac{k}{2 + \left(\frac{2(k-1)}{k + (k-2)\varepsilon}\right)(k-2)\varepsilon}$$

for every $\ell \in [k-1]$.

In addition, we formulate the following conjecture about the approximation hardness of the VERTEX COVER problem in k-partite k-hypergraphs. It implies the same inapproximability factor as in Theorem 6.4.3. But on the other hand, by combining the approximation preserving reduction used in Theorem 6.5.3 with Conjecture 6.5.1, we obtain an optimal inapproximability result for the VERTEX COVER problem in dense k-partite k-hypergraphs. Let us formulate our conjecture.

Conjecture 6.5.1
Let $\mathcal{H}$ be a k-partite k-hypergraph with given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$ and $k \geq 3$. Then, for every $\delta > 0$, the following is UG*-hard to decide.*

(i) Every vertex cover of $\mathcal{H}$ has size at least

$$|V(\mathcal{H})|\left(\frac{1}{k} - \delta\right).$$

(ii) The cardinality of a minimum vertex cover of $\mathcal{H}$ is at most

$$|V(\mathcal{H})|\left(\frac{2}{k^2} + \delta\right).$$

Assuming Conjecture 6.5.1, we give an approximation lower bound indicating that further densification in the sense of (ε, ℓ)-density does not affect the approximability of the underlying problem in contrast to the VERTEX COVER problem in (ε, ℓ)-dense k-hypergraphs. Let us formulate our inapproximability result.

Theorem 6.5.4
For every $k \geq 3$, it is UG*-hard to approximate the* VERTEX COVER *problem to within any constant approximation ratio less than*

$$\frac{k}{2 + (k-2)\varepsilon}$$

in (ε, ℓ)-dense k-partite hypergraphs for all $\ell \in [k-1]$ assuming Conjecture 6.5.1.

Based on Conjecture 6.5.1, we also give in Section 6.6.3 an optimal inapproximability result for the VERTEX COVER problem restricted to nearly regular k-partite k-hypergraphs

6.5.2 The Dense k-Balanced Case

We are going to design an approximation algorithm for the VERTEX COVER problem in k-balanced hypergraphs with an approximation ratio better than $k/2$. In particular, we are going to prove Theorem 6.5.1, which is restated below.

Theorem 6.5.1
There is a polynomial time approximation algorithm for the VERTEX COVER *problem in (ε, ℓ)-dense k-balanced hypergraphs with approximation ratio*

$$\frac{k}{k - (k-2)(1-\varepsilon)^{\frac{1}{k-\ell}}}.$$

In order to prove Theorem 6.5.1, we extend our approach for approximating the VERTEX COVER problem in dense k-hypergraphs in Section 5.5.3 to the dense k-balanced case. Firstly, we design an approximation algorithm that on input of a k-partite k-hypergraph $\mathcal{H}$ and a subset W of an optimal solution of $\mathcal{H}$ constructs a vertex cover of $\mathcal{H}$ with approximation ratio that is parametrized by the cardinality of the set W. Secondly, we give an algorithm that on input of a dense k-balanced hypergraph $\mathcal{H}$ extracts a large subset of a minimum vertex cover of $\mathcal{H}$. By combining the mentioned algorithms, we obtain the desired result. Afterwards, in Section 6.5.3, we design an improved extraction algorithm, which is applicable to dense k-partite k-hypergraphs and extracts a larger part of a minimum vertex cover of the input k-hypergraph.

Approximating the Remaining Instance

We present the approximation algorithm $\mathcal{A}_{6.3}$ defined in Figure 6.3 that on input of a k-partite k-hypergraph $\mathcal{H}$ and a subset W of a minimum vertex cover of $\mathcal{H}$ constructs a vertex cover of $\mathcal{H}$ with approximation ratio parametrized by the cardinality of the set W.

```
Algorithm $\mathcal{A}_{6.3}$
    Input  : $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, W)$, where $\mathcal{H}$ is a $k$-partite
             $k$-hypergraph with given vertex partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$
             and $W$ a subset of $V(\mathcal{H})$.
    Output: A vertex cover $S$ of $\mathcal{H}$.

    begin
        ① $R \leftarrow V_k(\mathcal{H})$;
        ② $\mathcal{H}' \leftarrow \mathcal{H}[V(\mathcal{H}) \backslash W]$;
        ③ $S' \leftarrow \mathcal{A}_{6.2}(\mathcal{H}', \{V_i(\mathcal{H}') \mid i \in [k]\})$;
        ④ Let $S$ be the smallest set among $R$ and $S' \cup W$;
        return $S$;
    end
```

Figure 6.3: Algorithm $\mathcal{A}_{6.3}$

We are going to prove the following lemma.

Lemma 6.5.2
Let $\mathcal{H}$ be a k-partite k-hypergraph with given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$, C

a minimum vertex cover of $\mathcal{H}$ and W a subset of $V(\mathcal{H})$ with $|W \cap C| \geq \delta \cdot |W|$ for some fixed constant $\delta \in (2k^{-1}, 1]$. On input $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, W)$, algorithm $\mathcal{A}_{6.3}$ constructs in polynomial time a vertex cover of $\mathcal{H}$ with approximation ratio

$$\frac{k}{2 + (\delta k - 2)\dfrac{|W|}{|V_k(\mathcal{H})|}}.$$

Proof of Lemma 6.5.2.
Let $\mathcal{H}$ be a k-partite k-hypergraph with given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$, C a minimum vertex cover of $\mathcal{H}$ and W a subset of $V(\mathcal{H})$ with

$$|W \cap C| \geq \delta \cdot |W| \text{ for some constant } \delta \in \left(\frac{2}{k}, 1\right]. \tag{6.15}$$

On input $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, W)$, algorithm $\mathcal{A}_{6.3}$ constructs the remaining k-partite k-hypergraph $\mathcal{H}'$ defined by $\mathcal{H}' = \mathcal{H}[V(\mathcal{H}) \backslash W]$. Let C' be a minimum vertex cover of $\mathcal{H}'$. Due to Theorem 6.4.1, we are able to construct in polynomial time a vertex cover S' of $\mathcal{H}'$ such that

$$|S'| \leq \frac{k}{2} \cdot |C'| \tag{6.16}$$

holds. Then, algorithm $\mathcal{A}_{6.3}$ returns $S \subseteq V(\mathcal{H})$ being the smallest set among $W \cup S'$ and $V_k(\mathcal{H})$. Consequently, we may assume that we have

$$|S| = \min\{|W \cup S'|, |V_k(\mathcal{H})|\} \leq |V_k(\mathcal{H})|. \tag{6.17}$$

Let us derive an upper bound on the cardinality of the produced solution S. Since the size of the optimal solution C can be bounded from below by $|C| \geq |C \cap W| + |C'|$, we obtain

$$\begin{aligned}
\frac{|S|}{|C|} &\leq \frac{|S|}{|C \cap W| + |C'|} \\
&\leq \frac{|S|}{\delta|W| + |C'|} && \text{(by (6.15))} \\
&= \frac{k}{\dfrac{k \cdot \delta \cdot |W| + k|C'|}{|S|}} = R_\delta
\end{aligned}$$

By the choice of δ (6.15), we have $\delta k - 2 > 0$. Consequently, we deduce that

$$\begin{aligned} R_\delta &= \frac{k}{\frac{2\,|W| + 2\,|S'|}{|S|} + \frac{(\delta k - 2)|W| + k \cdot |C'| - 2\,|S'|}{|S|}} \\ &\leq \frac{k}{2 + \frac{(\delta k - 2)|W| + k \cdot \left(|C'| - \frac{2}{k}\,|S'|\right)}{|S|}} && \text{(by (6.17))} \\ &\leq \frac{k}{2 + (\delta k - 2)\frac{|W|}{|S|}} && \text{(by (6.16))} \\ &\leq \frac{k}{2 + (\delta k - 2)\frac{|W|}{|V_k(\mathcal{H})|}} && \text{(by (6.17))} \end{aligned}$$

and the proof of Lemma 6.5.2 follows. ■

In the next section, we are going to define our first algorithm for the extraction of a part of an optimal vertex given a dense k-balanced hypergraph.

Extracting a Part of a Minimum Vertex Cover

In order to provide the set W for algorithm $\mathcal{A}_{6.3}$, we need to extract a large part of an optimal vertex cover of a given dense k-balanced hypergraph. Accordingly, we are going to define an algorithm for the extraction. As for the first step, we prove the following lemma.

Lemma 6.5.3

Let $\mathcal{H}$ be an ε-dense k-balanced hypergraph with given k-partition of the vertex set $\{V_i(\mathcal{H}) \mid i \in [k]\}$ and n vertices. Given a set B of $\left\lceil (1 - (1 - \varepsilon)^{\frac{1}{k}})n/k \right\rceil$-heaviest vertices in $V(\mathcal{H})$, then, all vertices $v \in B$ have degree in $\mathcal{H}$ at least

$$d_{\mathcal{H}}(v) \;\geq\; \left(1 - (1 - \varepsilon)^{\frac{k-1}{k}}\right)\left(\frac{n}{k}\right)^{k-1}.$$

Proof of Lemma 6.5.3.

Let $\mathcal{H}$ be a k-balanced hypergraph with given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$. $\mathcal{H}$

contains n vertices and m edges, where due to our assumption on the density of the hypergraph, we have that

$$m = |E(\mathcal{H})| \geq \varepsilon \cdot \left(\frac{n}{k}\right)^k . \tag{6.18}$$

Furthermore, let us denote by B the set consisting of some

$$\left[\left(1-(1-\varepsilon)^{\frac{1}{k}}\right)\frac{n}{k}\right]\text{-heaviest vertices in } V(\mathcal{H}).$$

For the sake of contradiction, suppose the statement is not true. Then, the number m of edges in $\mathcal{H}$ is strictly smaller than the number of edges in a k-balanced hypergraph, in which all vertices of B have the maximum possible degree and all the remaining vertices in $V(\mathcal{H})\backslash B$ have degree exactly

$$\left(1-(1-\varepsilon)^{\frac{k-1}{k}}\right)\left(\frac{n}{k}\right)^{k-1}.$$

Combining the bounds on the degree of the vertices, we attain that

$$\begin{aligned}
m &< \left(|B|\left(\frac{n}{k}\right)^{k-1} + \left(\frac{n}{k} - |B|\right)\left[1-(1-\varepsilon)^{\frac{k-1}{k}}\right]\left(\frac{n}{k}\right)^{k-1}\right) \\
&= \left(\left(1-(1-\varepsilon)^{\frac{1}{k}}\right)\frac{n}{k}\left(\frac{n}{k}\right)^{k-1}\right. \\
&\quad \left. + \left[\frac{n}{k} - \left(1-(1-\varepsilon)^{\frac{1}{k}}\right)\frac{n}{k}\right]\left(1-(1-\varepsilon)^{\frac{k-1}{k}}\right)\left(\frac{n}{k}\right)^{k-1}\right) \\
&= \left(1-(1-\varepsilon)^{\frac{1}{k}}\right)\left(\frac{n}{k}\right)^{k} + (1-\varepsilon)^{\frac{1}{k}}\left(1-(1-\varepsilon)^{\frac{k-1}{k}}\right)\left(\frac{n}{k}\right)^{k} \\
&= \varepsilon\left(\frac{n}{k}\right)^{k},
\end{aligned}$$

which is a contradiction with respect to (6.18). ■

We are going to prove Theorem 6.5.1 and first consider the case $\ell = 0$. In order to apply Lemma 6.5.2, we need to find a large subset W of a minimum vertex cover.

The algorithm $\mathcal{A}_{6.4}$ defined in Figure 6.4 returns a collection $\widetilde{W}$ of subsets $W_i \subseteq V(\mathcal{H})$ such that at least one of them is contained in a minimum vertex cover of $\mathcal{H}$.

```
Algorithm $\mathcal{A}_{6.4}$
Input  : $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, \varepsilon)$, where $\mathcal{H}$ is an $\varepsilon$-dense $k$-balanced
         hypergraph with given $k$-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$.
Output: A collection $\widetilde{W}$ of subsets of $V(\mathcal{H})$.

begin
    ① $\widetilde{W} \leftarrow \varnothing$;
    if $(k = 1)$ then
        ② $\widetilde{W} \leftarrow \left\{ \bigcup_{e \in E(\mathcal{H})} e \right\}$;
        return $\widetilde{W}$;
    else
        ③ Find a set $B$ consisting of some
        $[(1 - (1 - \epsilon)^{1/k})n/k]$-heaviest vertices of $\mathcal{H}$;
        ④ $\widetilde{W} \leftarrow \widetilde{W} \cup \{B\}$;
        foreach $v \in B$ do
            ⑤ $\epsilon' \leftarrow \left(1 - (1 - \epsilon)^{\frac{k}{(k+1)}}\right)$;
            ⑥ $\widetilde{W}' \leftarrow \mathcal{A}_{6.4}(\mathcal{H}(v), \{V_i(\mathcal{H}) \mid i \in [k], v \notin V_i(\mathcal{H})\}, \epsilon')$;
            ⑦ $\widetilde{W} \leftarrow \widetilde{W} \cup \widetilde{W}'$;
        end
    end
    return $\widetilde{W}$;
end
```

Figure 6.4: Algorithm $\mathcal{A}_{6.4}$

We are going to prove the following lemma.

Lemma 6.5.4

Let $\mathcal{H}$ be an ε-dense k-balanced hypergraph containing n vertices and given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$. In addition, let C be a vertex cover of $\mathcal{H}$. On input $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, \varepsilon)$, algorithm $\mathcal{A}_{6.4}$ constructs in polynomial time a collection $\widetilde{W} = \{W_i \subseteq V(\mathcal{H}) \mid i \in [s]\}$ of size $s = O(n^k)$ having the following *properties.*

(i) *There exists* $j \in [s]$ *with* $W_j \subseteq C$.

(ii) *For all* $i \in [s]$, *we have* $|W_i| \;\geq\; \left(1-(1-\varepsilon)^{\frac{1}{k}}\right)\frac{n}{k}$.

Proof of Lemma 6.5.4.

Let $\mathcal{H}$ be an ε-dense k-balanced hypergraph with n vertices and given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$. Furthermore, let C be a vertex cover of $\mathcal{H}$. On input $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, \varepsilon)$, algorithm $\mathcal{A}_{6.4}$ returns a collection $\widetilde{W}$ of size at most $|V_1|^k = \mathrm{O}(n^k)$ in $\mathrm{O}(n^k)$ time, which is polynomial since we assumed $k = \mathrm{O}(1)$.

The first condition is verified by induction. If all vertices in B belong to C, we have nothing to prove. Therefore, we may assume that there is a vertex $v \in B$ and a $b \in [k]$ such that $v \in V_b(\mathcal{H})$ and $v \notin C$. Then, the algorithm subtracts the whole partition V_b from $V(\mathcal{H})$ and proceeds on the $(k-1)$-balanced hypergraph $\mathcal{H}(v)$ with $(k-1)$-partition $\{V_i(\mathcal{H}(v)) \mid i \in [k]\backslash\{b\}\}$ and $E(\mathcal{H}(v)) \;=\; \{\, e\backslash\{v\} \mid v \in e \in E(\mathcal{H}) \,\}$. A vertex cover of $\mathcal{H}$ must contain a vertex cover of the $(k-1)$-balanced hypergraph $\mathcal{H}(v)$ as otherwise, some edges will not be covered. By induction, the recursive call returns a collection of subsets of $\mathcal{H}(v)$ including a subset W_j' contained in C. The base case $k = 1$ is trivial.

We prove the second property by induction as well. Suppose that for a fixed value of k, we have $|W_i| \;\geq\; (1-(1-\varepsilon)^{1/k})\,(n/k)$ for all $i \in [s]$ and all k-balanced hypergraphs. We now prove the property for $k+1$. By Lemma 6.5.3, the recursive calls are performed on ε'-dense k-balanced hypergraphs with $n - n/(k+1)$ vertices. Thus, by the induction hypothesis, the recursive call returns a collection of sets W_i of size

$$\begin{aligned} |W_i| \;&\geq\; \left(1-(1-\varepsilon')^{\frac{1}{k}}\right)\left(n - \frac{n}{k+1}\right)\frac{1}{k} \;=\; \left[1-\left(1-(1-\varepsilon)^{\frac{k}{k+1}}\right)^{\frac{1}{k}}\right]\frac{kn}{(k+1)k} \\ &\geq\; \left(1-(1-\varepsilon)^{\frac{1}{k+1}}\right)\left(\frac{n}{k+1}\right), \end{aligned}$$

as claimed. The base case $k = 1$ is verified, as in that case the algorithm returns at least εn vertices. ■

When the given k-balanced hypergraph $\mathcal{H}$ is (ε, ℓ)-dense with $\ell > 0$, it is possible to extract an even larger part of a minimum vertex cover of $\mathcal{H}$. We present the extraction algorithm $\mathcal{A}_{6.5}$ for the (ε, ℓ)-dense case with $\ell > 0$ defined in Figure 6.5.

```
Algorithm $\mathcal{A}_{6.5}$

Input : $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, \varepsilon, \ell)$, where $\mathcal{H}$ is an $(\varepsilon, \ell)$-dense
        $k$-hypergraph with $\ell > 0$ and given $k$-partition
        $\{V_i(\mathcal{H}) \mid i \in [k]\}$.
Output: A collection $\widetilde{W}$ of subsets $W_i \subseteq V(\mathcal{H})$.

begin
    ① $\widetilde{W} \leftarrow \{V_i(\mathcal{H}) \mid i \in [k]\}$;
    foreach $S \in \left\{P \in \binom{V(\mathcal{H})}{\ell} \,\middle|\, \exists I \in \binom{[k]}{\ell} \forall i \in I : |V_i \cap P| = 1\right\}$ do
        ② Let $S = \{v_i \in V_i(\mathcal{H}) \mid i \in I\}$;
        ③ Define $\mathcal{H}'$ to be the $(k-\ell)$-balanced hypergraph with
        vertex partition $\{V_i(\mathcal{H}) \mid i \in [k] \backslash I\}$, edge set
        $E(\mathcal{H}') = \{e \backslash S \mid S \subseteq e \in E(\mathcal{H})\}$ and vertex set $\bigcup_{i \in [k] \backslash I} V_i(\mathcal{H})$;
        ④ $\widetilde{W}' \leftarrow \mathcal{A}_{6.4}(\mathcal{H}', \{V_i(\mathcal{H}) \mid i \in [k] \backslash I\}, \varepsilon)$;
        ⑤ $\widetilde{W} \leftarrow \widetilde{W} \cup \widetilde{W}'$;
    end
    return $\widetilde{W}$;
end
```

Figure 6.5: Algorithm $\mathcal{A}_{6.5}$

We are going to prove the following lemma.

Lemma 6.5.5

Let $\mathcal{H}$ be an (ε, ℓ)-dense k-balanced hypergraph with $\ell > 0$, n vertices and given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$. Furthermore, let C be a vertex cover of $\mathcal{H}$ with $k \cdot |C| \leq n$. On input $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, \varepsilon, \ell)$, algorithm $\mathcal{A}_{6.5}$ constructs *in polynomial time a collection of sets* $\widetilde{W} = \{W_i \subseteq V(\mathcal{H}) \mid i \in [s]\}$ *of size* $s = O(n^k)$ *having the following properties.*

(i) *There exists* $j \in [s]$ *such that* W_j *is a subset of* C.

(ii) *For all* $i \in [s]$, *we have that*

$$|W_i| \;\geq\; \left(1 - (1-\varepsilon)^{\frac{1}{k-\ell}}\right)\left(\frac{n}{k}\right).$$

Proof of Lemma 6.5.5.

Let $\mathcal{H}$ be an (ε, ℓ)-dense k-balanced hypergraph with $\ell > 0$, n vertices and given vertex partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$. By definition, there exists a $I \subseteq [k]$ with $|I| = \ell$ such that for all subsets

$$S \in \left\{ S' \in \binom{V(\mathcal{H})}{\ell} \;\middle|\; |V_i \cap S'| = 1 \text{ for all } i \in I \right\} :\; d_{\mathcal{H}}(S) \geq \varepsilon \cdot \left(\frac{n}{k}\right)^{k-\ell} \qquad (6.19)$$

holds. On input $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, \varepsilon, \ell)$, algorithm $\mathcal{A}_{6.5}$ finds in polynomial time the set I of indices with the property (6.19) by exhaustive search. If there is a $k_0 \in [k]$ with $V_{k_0} \subseteq C$, we are done. Otherwise, there is a set $S_C = \{v_i^C \in V_i \mid i \in I\}$ of ℓ vertices with $S_C \cap C = \varnothing$. Let us consider the $(k-\ell)$-balanced hypergraph $\mathcal{H}'$ with vertex partition $\{V_i \mid i \in [k]\backslash I\}$ and edge set $E(\mathcal{H}') = \{e \backslash S_C \mid S_C \subseteq e \in E(\mathcal{H})\}$. Due to the property (6.19), we know that $\mathcal{H}'$ is ε-dense. In order to cover the edges in $E(\mathcal{H}')$, there is a vertex cover C' of $\mathcal{H}'$ that is also contained in C. According to Lemma 6.5.4, algorithm $\mathcal{A}_{6.4}$ returns a set $\widetilde{W} = \{W_i \subseteq V(\mathcal{H}') \mid i \in [s]\}$ on input $\mathcal{H}'$ with the following properties.

(i) There exists $j \in [s]$ such that W_j is a subset of C'.

(ii) For all $i \in [s]$, we have that $|W_i| \geq \left(1 - (1-\varepsilon)^{\frac{1}{k-\ell}}\right)\left(\frac{n}{k}\right)$.

In particular, it implies that there exists a $j \in [s]$ with $W_j \subseteq C$. Furthermore, we know that algorithm $\mathcal{A}_{6.4}$ constructs $\widetilde{W}$ in polynomial time and $s = \mathrm{O}\left(n^{k-\ell}\right)$. By enumerating all $\mathrm{O}(n^\ell)$ possibilities for S_C, we obtain the result in $\mathrm{O}(n^k)$ time. ■

We now give the proof of Theorem 6.5.1.

Proof of Theorem 6.5.1

Let $\mathcal{H}$ be an (ε,ℓ)-dense k-balanced hypergraph with given k-partition $\{V_i \mid i \in [k]\}$ containing n vertices and C a minimum vertex cover of $\mathcal{H}$. In order to prove our theorem, we are going to analyze algorithm $\mathcal{A}_{6.9}$ given in Figure 6.9. It combines both extraction algorithms providing a large part W of a minimum vertex cover of a given dense k-balanced hypergraph $\mathcal{H}$ with algorithm $\mathcal{A}_{6.5}$. As we will see, we obtain in this way the approximation algorithm for the VERTEX COVER problem in dense k-balanced hypergraphs with the desired approximation ratio.

Algorithm $\mathcal{A}_{6.9}$

```
Input  : $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, \varepsilon, \ell)$, where $\mathcal{H}$ is an $(\varepsilon,\ell)$-dense
         $k$-partite $k$-hypergraph with given vertex partition
         $\{V_i(\mathcal{H}) \mid i \in [k]\}$.
Output: A vertex cover $S$ of $\mathcal{H}$.

begin
    ① $\widetilde{S} \leftarrow \emptyset$;
    if $(\ell = 0)$ then
        ② $\widetilde{W} \leftarrow \mathcal{A}_{6.4}(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, \varepsilon)$;
    else
        ③ $\widetilde{W} \leftarrow \mathcal{A}_{6.5}(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, \varepsilon, \ell)$;
    end
    foreach $W_i \in \widetilde{W}$ do
        ④ $\widetilde{S'} \leftarrow \mathcal{A}_{6.3}(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, W_i)$;
        ⑤ $\widetilde{S} \leftarrow \widetilde{S} \cup \widetilde{S'}$;
    end
    ⑥ Let $S$ be the smallest set in the collection $\widetilde{S}$;
    return $S$;
end
```

Figure 6.6: Algorithm $\mathcal{A}_{6.6}$

Let us first consider the case $\ell > 0$. On input $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, \varepsilon, \ell)$, algorithm $\mathcal{A}_{6.5}$ constructs in polynomial time a collection $\widetilde{W}$ of subsets $W_i \subseteq V(\mathcal{H})$. According to Lemma 6.5.5, $\widetilde{W}$ contains a set W_j being included in C. In addition to that, the size of W_j is at least

$$|W_j| \ \geq \ |C \cap W_j| \ \geq \ \left(1 - (1-\varepsilon)^{\frac{1}{k-\ell}}\right)\frac{n}{k}. \tag{6.20}$$

For every set $W_i \in \widetilde{W}$, algorithm $\mathcal{A}_{6.3}$ returns on input $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, W_i, t)$ in polynomial time a vertex cover S_i of $\mathcal{H}$. By Lemma 6.5.2, we know that the size of the particular vertex cover S_j of $\mathcal{H}$ is at most

$$\begin{aligned}
\frac{|S_j|}{|C|} &\leq \frac{k}{2 + (k-2)\dfrac{|W_j|}{|V_k(\mathcal{H})|}} \\
&\leq \frac{k}{2 + (k-2)\dfrac{|W_j|}{(n/k)}} \\
&\leq \frac{k}{2 + (k-2)\left(1 - (1-\varepsilon)^{\frac{1}{k-\ell}}\right)} && \text{(by (6.20))} \\
&= \frac{k}{k - (k-2)(1-\varepsilon)^{\frac{1}{k-\ell}}}
\end{aligned}$$

Since the algorithm $\mathcal{A}_{6.9}$ returns the smallest vertex cover among S_i with $i \in [s]$, we obtain the claimed result. Finally, we note that a similar argumentation holds in the case $\ell = 0$.

By Lemma 6.5.5, the number s of sets W_i contained in $\widetilde{W}$ is $\mathrm{O}(n^k)$. Consequently, the running time of algorithm $\mathcal{A}_{6.9}$ remains polynomial and the proof of Theorem 6.5.1 follows. ■

6.5.3 An Improved Approximation Algorithm

In this section, we are going to design an efficient approximation algorithm for the VERTEX COVER problem in dense k-partite k-hypergraphs. It is the first approximation algorithm for the VERTEX COVER problem in dense

k-partite k-hypergraphs with approximation ratio strictly less than $k/2$. In addition, it achieves a better approximation ratio than algorithm $\mathcal{A}_{6.9}$ given in the previous section. In particular, we are going to prove Theorem 6.5.2 restated below.

Theorem 6.5.2
There is a polynomial time approximation algorithm for the VERTEX COVER *problem in ε-dense k-partite k-hypergraphs with approximation ratio*

$$\frac{k}{2 + (k-2)\varepsilon} .$$

A crucial ingredient of the proof of Theorem 6.5.2 is Lemma 6.5.1, in which we show that we can extract efficiently a large part of an optimal vertex cover of a given ε-dense k-partite k-hypergraph $\mathcal{H}$ with k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$. Moreover, we obtain in this way a constructive proof that the size of a vertex cover of $\mathcal{H}$ is bounded from below by $\big(\varepsilon \cdot |V_k(\mathcal{H})|\big)$.

The Improved Extraction Algorithm

We are going to define an improved extraction algorithm. On the one hand, it extracts a larger part of a minimum vertex cover compared to algorithm $\mathcal{A}_{6.9}$. On the other hand, it can be applied to k-partite k-hypergraphs, whereas algorithm $\mathcal{A}_{6.9}$ extracts only a large part of a minimum vertex cover if the given hypergraph is dense k-balanced. Moreover, we are going to prove Lemma 6.5.1.

Lemma 6.5.1
Let $\mathcal{H}$ be an ε-dense k-partite k-hypergraph with given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$ and C a minimum vertex cover of $\mathcal{H}$. There is an algorithm that on input $\mathcal{H}$ and its k-partition computes in polynomial time a collection $\widetilde{M}$ of subsets $W_i \subseteq V(\mathcal{H})$ such that the size of $\widetilde{M}$ is polynomial in $|V(\mathcal{H})|$ and $\widetilde{M}$ contains a set $S_C \subseteq C$ with $|S_C| \geq \varepsilon \cdot |V_k(\mathcal{H})|$.

We present our extraction algorithm for k-partite k-hypergraphs given in Figure 6.7.

Algorithm $\mathcal{A}_{6.7}$

```
Input  : $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\})$, where $\mathcal{H}$ is an $\varepsilon$-dense $k$-partite
         $k$-hypergraph with given $k$-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$.
Output: A collection $\widetilde{M}$ of subsets $W_i \subseteq V(\mathcal{H})$.

begin
    if $(k = 1)$ then
        ① $\widetilde{M} \leftarrow \left\{ \bigcup_{e \in E(\mathcal{H})} e \right\}$;
        return $\widetilde{M}$;
    else
        ② $p \leftarrow \left\lceil \frac{|V_k(\mathcal{H})|}{\Psi_{\mathcal{H}}(n)} \right\rceil$;
        ③ Find a set $\{v_1, .., v_p\}$ consisting of some $p$-heaviest vertices
        of $V_k(\mathcal{H})$ with $d_{\mathcal{H}}(v_i) \geq d_{\mathcal{H}}(v_{i+1})$ for all $i \in [p-1]$;
        ④ $\widetilde{M} \leftarrow \{\{v_1, .., v_p\}\}$;
        for $i = 1, \ldots, p$ do
            ⑤ $R \leftarrow \{v_j \mid j \in [i-1]\}$;
            ⑥ $\widetilde{M}' \leftarrow \mathcal{A}_{6.7}(\mathcal{H}(v_i), \{V_j(\mathcal{H}) \mid j \in [k-1]\})$;
            ⑦ $\widetilde{M} \leftarrow \widetilde{M} \cup \{R \cup S \mid S \in \widetilde{M}'\}$;
        end
    end
    return $\widetilde{M}$;
end
```

Figure 6.7: Algorithm $\mathcal{A}_{6.7}$

Before we prove Lemma 6.5.1, we describe the main idea of the proof. We denote by n_k the cardinality of the set $V_k(\mathcal{H})$. Let C be an optimal vertex cover of $\mathcal{H}$. For a $p \in [n_k]$, let $R = \{v_1, .., v_p\}$ be a set consisting of some p-heaviest vertices of $V_k(\mathcal{H})$ with $d_{\mathcal{H}}(v_i) \geq d_{\mathcal{H}}(v_{i+1})$. Then, we argue that either the whole set R is contained in C, or algorithm $\mathcal{A}_{6.7}$ finds the highest degree vertex in R, say v_u, with $v_u \notin C$. Clearly, we have $\{v_1, \ldots, v_{u-1}\} \subseteq C$.

Accordingly, algorithm $\mathcal{A}_{6.7}$ tries to obtain a large part of a vertex cover of the v_u-induced hypergraph $\mathcal{H}(v_u)$. As we will see, $\mathcal{H}(v_u)$ is still dense enough and algorithm $\mathcal{A}_{6.7}$ returns a part of a vertex cover C' of $\mathcal{H}(v_u)$ with $C' \subseteq C$. The union $\{v_1, \dots, v_{u-1}\} \cup C'$ yields a large part of a minimum vertex cover of $\mathcal{H}$.

Let us now turn to the proof of Lemma 6.5.1.

Proof of Lemma 6.5.1.

We will split the proof of Lemma 6.5.1 in several parts. In particular, we prove that given an ε-dense k-partite k-hypergraph $\mathcal{H}$ with n vertices and its k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$, the algorithm $\mathcal{A}_{6.7}$ produces a collection $\widetilde{M}$ of subsets $S \subseteq V(\mathcal{H})$ having the following properties:

(i) For all vertex cover C of $\mathcal{H}$, there is a $S_C \in \widetilde{M}$ such that $S_C \subseteq C$.

(ii) For every $S \in \widetilde{M}$, the cardinality of S is at least $|S| \;\geq\; \varepsilon \cdot |V_k(\mathcal{H})|$.

(iii) The algorithm $\mathcal{A}_{6.7}$ constructs $\widetilde{M}$ in polynomial time and the cardinality of $\widetilde{M}$ is $\mathrm{O}(n^k)$.

Let us start with property (iii).

(iii) We see that the size of $\widetilde{M}$ is bounded from above by $|V_1|^k = \mathrm{O}(n^k)$. This implies that the running time of $\mathcal{A}_{6.7}$ on input $\mathcal{H}$ is polynomial in n.

The properties (ii) and (iii) will be proved by induction.

In the case $(k = 1)$, the set $\bigcup\limits_{e \in E(\mathcal{H})} e$ is by definition subset of every vertex cover of $\mathcal{H}$. Since $\mathcal{H}$ is ε-dense, the cardinality of $|E(\mathcal{H})|$ is bounded from below by $\varepsilon \cdot |V_1(\mathcal{H})|$.

We may assume that $k > 1$ and the statement is true for all $(k-1)$-partite $(k-1)$-hypergraphs. Let $\mathcal{H}$ be an ε-dense k-partite k-hypergraph, C a vertex cover of $\mathcal{H}$ and $\{v_1, .., v_p\}$ be a set consisting of some p-heaviest vertices of $V_k(\mathcal{H})$, where

$$p \;=\; \left\lceil \frac{|V_k(\mathcal{H})|}{\Psi_{\mathcal{H}}(n)} \right\rceil. \tag{6.21}$$

Furthermore, we assume that $d_{\mathcal{H}}(v_i) \geq d_{\mathcal{H}}(v_{i+1})$ for all $i \in [p-1]$. If $\{v_1,..,v_p\}$ is contained in C, we have constructed a subset of C with cardinality

$$p \;=\; \left\lceil \frac{|V_k(\mathcal{H})|}{\Psi_{\mathcal{H}}(n)} \right\rceil \;\geq\; \frac{|E(\mathcal{H})| \cdot |V_k(\mathcal{H})|}{\prod\limits_{i\in[k]} |V_i(\mathcal{H})|} \;\geq\; \varepsilon \cdot |V_k(\mathcal{H})|.$$

Otherwise, there is an integer $u \in [p]$ such that $R_u = \{v_1,\ldots,v_{u-1}\} \subseteq C$ and $v_u \notin C$. But this means that C contains a vertex cover of the v_u-induced $(k-1)$-partite $(k-1)$-hypergraph $\mathcal{H}(v_u)$ in order to cover all edges in $\{e \in E(\mathcal{H}) \mid v_u \in e\}$. By our induction hypothesis, algorithm $\mathcal{A}_{6.7}$ returns on input $\mathcal{H}(v_u)$ a collection $\widetilde{M_u}$ containing a set S_u which is a subset of a vertex cover of $\mathcal{H}(v_u)$ and of C.

It remains to be proved that the cardinality of S_u is large enough. More precisely, we are going to prove that $|S_u|$ can be bounded from below by

$$|S_u| \;\geq\; \varepsilon \cdot |V_k(\mathcal{H})| - |R_u|. \tag{6.22}$$

Therefore, we need to analyze the density of the v_u-induced hypergraph $\mathcal{H}(v_u)$. The edge set of $\mathcal{H}(v_u)$ is given by $\{\, e\backslash\{v_u\} \mid v_u \in e \in E(\mathcal{H}) \,\}$. Thus, we have to obtain a lower bound on the degree of v_u in $\mathcal{H}$. Since we have

$$|\{\, e \in E(\mathcal{H}) \mid e \cap R_u \neq \varnothing \,\}| \;\leq\; |R_u| \prod_{i\in[k-1]} |V_i(\mathcal{H})|, \tag{6.23}$$

the vertices included in $V_k(\mathcal{H})\backslash R_u$ have an average degree in $\mathcal{H}$ at least

$$\begin{aligned}
\frac{\sum\limits_{v\in V_k(\mathcal{H})\backslash R_u} d_{\mathcal{H}}(v)}{|V_k(\mathcal{H})\backslash R_u|} \;&\geq\; \frac{\varepsilon \cdot \prod\limits_{i\in[k]} |V_i(\mathcal{H})| - |\{e \in E(\mathcal{H}) \mid e \cap R_u \neq \varnothing\}|}{|V_k(\mathcal{H})\backslash R_u|} \\
&\geq\; \frac{\varepsilon \cdot \prod\limits_{i\in[k]} |V_i(\mathcal{H})| - |R_u| \prod\limits_{i\in[k-1]} |V_i(\mathcal{H})|}{|V_k(\mathcal{H})\backslash R_u|} \qquad \text{(by (6.23))} \\
&\geq\; \frac{(\varepsilon \cdot |V_k(\mathcal{H})| - |R_u|) \prod\limits_{i\in[k-1]} |V_i(\mathcal{H})|}{|V_k(\mathcal{H})\backslash R_u|}
\end{aligned}$$

Let $\widetilde{v}$ be the heaviest vertex included in $V_k(\mathcal{H})\backslash R_u$. Then, $\widetilde{v}$ must have a degree in $\mathcal{H}$ at least

$$d_{\mathcal{H}}(\widetilde{v}) \quad \geq \quad \frac{(\varepsilon \cdot |V_k(\mathcal{H})| - |R_u|) \prod\limits_{l \in [k-1]} |V_l|}{|V_k(\mathcal{H})\backslash R_u|}.$$

Accordingly, we deduce that the size of $E(\mathcal{H}(v_u))$ can be bounded from below by

$$|E(\mathcal{H}(v_u))| \;\geq\; \frac{(\varepsilon \cdot |V_k(\mathcal{H})| - |R_u|) \prod\limits_{i \in [k-1]} |V_i(\mathcal{H})|}{|V_k(\mathcal{H})\backslash R_u|} \tag{6.24}$$

The vertex partition of $\mathcal{H}(v_u)$ is simply given by $\{V_i(\mathcal{H}) \mid i \in [k-1]\}$. By our induction hypothesis and by (6.24), the size of every set contained in $\widetilde{M}_u$ is at least

$$\frac{|E(\mathcal{H}(v_u))|}{\prod\limits_{i \in [k-1]} |V_i(\mathcal{H})|} \cdot |V_{k-1}(\mathcal{H})| \;\geq\; \frac{(\varepsilon \cdot |V_k(\mathcal{H})| - |R_u|) \cdot \prod\limits_{i \in [k-1]} |V_i(\mathcal{H})|}{|V_k(\mathcal{H})\backslash R_u| \cdot \prod\limits_{i \in [k-1]} |V_i(\mathcal{H})|} \cdot |V_{k-1}(\mathcal{H})|$$

By $|V_{k-1}(\mathcal{H})| \geq |V_k(\mathcal{H})|$, we deduce the following lower bound on the size of sets contained in $\widetilde{M}_u$.

$$\begin{aligned} \frac{(\varepsilon \cdot |V_k(\mathcal{H})| - |R_u|) \prod\limits_{i \in [k-1]} |V_i(\mathcal{H})|}{|V_k(\mathcal{H})\backslash R_u| \prod\limits_{i \in [k-1]} |V_i(\mathcal{H})|} \cdot |V_{k-1}(\mathcal{H})| \;&\geq\; \frac{(\varepsilon \cdot |V_k(\mathcal{H})| - |R_u|)}{|V_k(\mathcal{H})\backslash R_u|} \cdot |V_k(\mathcal{H})| \\ &\geq\; \frac{(\varepsilon \cdot |V_k(\mathcal{H})| - |R_u|)}{|V_k(\mathcal{H})|} \cdot |V_k(\mathcal{H})| \\ &=\; \varepsilon \cdot |V_k(\mathcal{H})| - |R_u| \end{aligned}$$

We obtain the claimed property (6.22). By combining the deduced facts, we get that

$$|R_u \cup S_u| \;\geq\; |R_u| + (\varepsilon \cdot |V_k(\mathcal{H})| - |R_u|) \;=\; \varepsilon \cdot |V_k(\mathcal{H})|.$$

Clearly, this argumentation on the size of $R_u \cup S_u$ can be applied to every $u \in [p]$ and the proof of Lemma 6.5.1 follows. ■

Before we present our polynomial time approximation algorithm for the VERTEX COVER problem in dense k-partite k-hypergraphs, we prove that the bound stated in Lemma 6.5.1 is tight.

Family of Tight Examples

We are going to define a family of k-balanced hypergraphs and prove that for each member of this family, the bound stated in Lemma 6.5.1 is tight. Let us define the family of k-balanced hypergraphs.

Definition 6.5.1 (Family of k-balanced hypergraphs $\mathscr{H}_T$)
For every $k \in \mathbb{N}$, $l \in \mathbb{N}$ and $u \in [l]$, we define the k-balanced hypergraph $\mathcal{H}(k,l,\varepsilon)$ with $\varepsilon = u/l$ and k-partition $\{V_i(\mathcal{H}(k,l,\varepsilon)) \mid i \in [k]\}$. $V_k(\mathcal{H}(k,l,\varepsilon))$ consists of the set of vertices $\{v_1,\dots,v_l\}$. For each $u \in [l]$, we introduce the subset $V_k^u = \{v_1,\dots,v_u\}$ of $V_k(\mathcal{H}(k,l,\varepsilon))$. Then, the edge set of $\mathcal{H}(k,l,u/l)$ is given by

$$E(\mathcal{H}(k,l,u/l)) = \left\{ \{v^1,\dots,v^k\} \,\middle|\, v^1 \in V_1(\mathcal{H}(k,l,u/l)),\dots,v^k \in V_k^u \right\}.$$

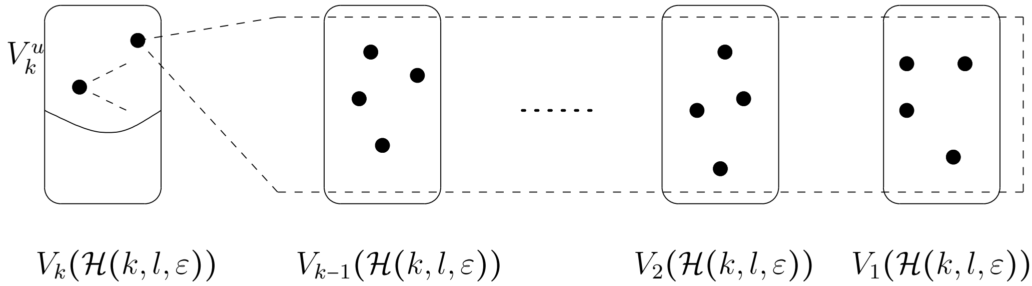
The family $\mathscr{H}_T$ of k-balanced hypergraphs is defined as follows.

$$\mathscr{H}_T = \left\{ \mathcal{H}\left(k,l,\frac{u}{l}\right) \,\middle|\, k \in \mathbb{N}, l \in \mathbb{N}, u \in [l] \right\}$$

We are going to prove the following lemma.

Lemma 6.5.6
For every $k \in \mathbb{N}$, $l \in \mathbb{N}$ and $u \in [l]$, the algorithm $\mathcal{A}_{6.4}$ constructs on input $\mathcal{H}(k,l,u/l) \in \mathscr{H}_T$ a collection $\widetilde{M}$ containing a minimum vertex cover of $\mathcal{H}(k,l,u/l)$ of size u.

Before we turn to the proof of Lemma 6.5.6, we display a member of the family $\mathscr{H}_T$ in Figure 6.8.



V_k^u
$V_k(\mathcal{H}(k,l,\varepsilon))$
$V_{k-1}(\mathcal{H}(k,l,\varepsilon))$
$V_2(\mathcal{H}(k,l,\varepsilon))$
$V_1(\mathcal{H}(k,l,\varepsilon))$

Figure 6.8: Illustration of the k-hypergraph $\mathcal{H}(k,l,\varepsilon)$

Let us proceed to the proof of Lemma 6.5.6.

Proof of Lemma 6.5.6.
For a fixed $p \geq 1$, $u \in [p]$ and $k \geq 2$, we consider $\mathcal{H}(k,p,\varepsilon) \in \mathscr{H}_T$ with $\varepsilon = u/p$. First, we note that $\mathcal{H}(k,p,\varepsilon)$ is ε-dense, since we have

$$\frac{|E(\mathcal{H}(k,p,\varepsilon))|}{\prod\limits_{j\in[k]} |V_j(\mathcal{H}(k,p,\varepsilon))|} = \frac{|V_k^u|}{|V_k(\mathcal{H}(k,p,\varepsilon))|} = \frac{u}{p} = \varepsilon. \tag{6.25}$$

Then, algorithm $\mathcal{A}_{6.4}$ returns a collection $\widetilde{M}$ including V_k^u. By (6.25), we obtain

$$|V_k^u| = \left(\frac{|V_k^u|}{|V_k(\mathcal{H}(k,p,\varepsilon))|}|V_k(\mathcal{H}(k,p,\varepsilon))|\right) = \varepsilon \cdot |V_k(\mathcal{H}(k,p,\varepsilon))|.$$

On the other hand, the remaining hypergraph $\mathcal{H}'$ with vertex partition $\{V_1(\mathcal{H}(k,p,\varepsilon)),\dots,V_k(\mathcal{H}(k,p,\varepsilon))\backslash V_k^u\}$ and edge set $E(\mathcal{H}') = \{e \in E(\mathcal{H}) \mid e \cap V_k^u = \varnothing\}$ is already covered, since $E(\mathcal{H}')$ is by definition of $\mathcal{H}(k,p,\varepsilon)$ the empty set. Therefore, V_k^u is a vertex cover of $\mathcal{H}(k,p,\varepsilon)$.

According to Corollary 6.5.1, every vertex cover of $\mathcal{H}(k,p,\varepsilon)$ is bounded from below by $\varepsilon\cdot|V_k(\mathcal{H}(k,p,\varepsilon))|$ implying that V_k^u must be an optimal vertex cover. ■

Let us now turn to the proof of Theorem 6.5.2.

Proof of Theorem 6.5.2

Let $\mathcal{H}$ be an ε-dense k-partite k-hypergraph with given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$ and C a minimum vertex cover of $\mathcal{H}$. The algorithm $\mathcal{A}_{6.9}$ defined in Figure 6.9 combines the extraction algorithm $\mathcal{A}_{6.7}$ to generate a large enough subset W of a minimum vertex cover of $\mathcal{H}$ together with the algorithm $\mathcal{A}_{6.3}$ that on input W and $\mathcal{H}$ constructs a vertex cover of $\mathcal{H}$ with approximation ratio parametrized by the cardinality of W.

We present our polynomial time approximation algorithm for the VERTEX COVER problem in dense k-partite k-hypergraphs given in Figure 6.9.

```
Algorithm $\mathcal{A}_{6.9}$
Input  : $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\})$, where $\mathcal{H}$ is an $\varepsilon$-dense $k$-partite
         $k$-hypergraph and $\{V_i(\mathcal{H}) \mid i \in [k]\}$ the given vertex
         partition of $\mathcal{H}$.
Output: A vertex cover $S$ of $\mathcal{H}$.
begin
    ① $\widetilde{S} \leftarrow \varnothing$;
    ② $\widetilde{M} \leftarrow \mathcal{A}_{6.7}(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\})$;
    foreach $M \in \widetilde{M}$ do
        ③ $S \leftarrow \mathcal{A}_{6.3}(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, M)$;
        ④ $\widetilde{S} \leftarrow \widetilde{S} \cup \{S\}$
    end
    ⑤ Let $S$ be the smallest set in the collection $\widetilde{S}$;
    return $S$;
end
```

Figure 6.9: Algorithm $\mathcal{A}_{6.9}$

The extraction algorithm $\mathcal{A}_{6.7}$ constructs on input $\mathcal{H}$ and its partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$ a collection $\widetilde{M}$ of subsets $M \subseteq V(\mathcal{H})$. According to Lemma 6.5.1, there is a $M_C \in \widetilde{M}$ such that $M_C \subseteq C$ and $|V_k(\mathcal{H})| \cdot \varepsilon \le |M_C|$. In addition, we know that the size of $\widetilde{M}$ is polynomial in $|V(\mathcal{H})|$. Then, we apply algorithm $\mathcal{A}_{6.3}$ to every set in $\widetilde{M}$ together with $\mathcal{H}$ yielding a vertex cover of $\mathcal{H}$. Due to Lemma 6.5.2, the size of the smallest vertex cover among them, say S', can be bounded from above by

$$|S'| \le \left(\frac{k}{2 + (\delta k - 2)\dfrac{|M_C|}{|V_k(\mathcal{H})|}} \right) \cdot |C|.$$

Since we have $|M_C| \le |M_C \cap C|$ and $|V_k(\mathcal{H})| \cdot \varepsilon \le |M_C|$, we obtain

$$\frac{|S'|}{|C|} \le \frac{k}{2 + (k-2)\dfrac{|V_k(\mathcal{H})| \cdot \varepsilon}{|V_k(\mathcal{H})|}} \le \frac{k}{2 + (k-2)\varepsilon}$$

and proof of Theorem 6.5.2 follows. ■

6.5.4 Approximation Lower Bounds

In this section, we study the approximation hardness of the VERTEX COVER problem restricted to dense k-partite k-hypergraphs. We give an UG-hardness and a **NP**-hardness of approximation result for the underlying problem. Furthermore, we propose a conjecture on the UG-hardness of approximating the VERTEX COVER problem in k-partite k-hypergraphs. Assuming this conjecture, we obtain an optimal inapproximability result.

UG-Hardness

We construct an approximation preserving reduction from the VERTEX COVER problem in k-partite k-hypergraphs to the dense version of the problem. By densifying hard instances given in Theorem 6.4.3, we are going to prove the following UG-hardness result.

Theorem 6.5.3
For every $k \geq 3$, it is UG*-hard to approximate the* VERTEX COVER *problem in (ε, ℓ)-dense k-partite hypergraphs to within any constant approximation ratio less than*

$$\frac{k}{2 + \left(\frac{2(k-1)}{k + (k-2)\varepsilon}\right)(k-2)\varepsilon}$$

for every $\ell \in [k-1]$.

Proof of Theorem 6.5.3.
Let us first concentrate on the ε-dense case. Afterwards, we extend the range of ℓ. As starting point of our reduction, we consider the k-partite k-hypergraph $\mathcal{H}$ given in Theorem 6.4.3 and construct an ε-dense k-partite k-hypergraph $\mathcal{H}'$ by adding a k-partite clique $\mathcal{K}$ of appropriate size to $\mathcal{H}$. Recall that according to Theorem 6.4.3, for every $\delta > 0$, the following two cases (i) and (ii) of the promise problem are UG-hard to decide.

(i) Every vertex cover of $\mathcal{H}$ has size at least

$$|V(\mathcal{H})|\left(\frac{1}{2(k-1)}-\delta\right).$$

(ii) The cardinality of a minimum vertex cover is at most

$$|V(\mathcal{H})|\left(\frac{1}{k(k-1)}+\delta\right).$$

We start with the description of $\mathcal{H}'$. Let $\{V_i(\mathcal{H}) \mid i \in [k]\}$ be the given k-partition of $\mathcal{H}$. The i-th partition of the clique $\mathcal{K}$ is denoted by $V_i(\mathcal{K})$. Then, we join the set $V_i(\mathcal{K})$ of

$$\left(\frac{\varepsilon}{1-\varepsilon}\right)|V_i(\mathcal{H})| \tag{6.26}$$

vertices to $V_i(\mathcal{H})$ forming the set $V_i(\mathcal{H}')$ for every $i \in [k]$. Consequently, we obtain

$$|V_i(\mathcal{H}')| = |V_i(\mathcal{H})| + \left(\frac{\varepsilon}{1-\varepsilon}\right)|V_i(\mathcal{H})| \text{ for all } i \in [k]. \tag{6.27}$$

Then, we add all possible edges e to $E(\mathcal{H}')$ with the restrictions $|e| = k$, $V_k(\mathcal{K}) \cap e \neq \varnothing$ and $e \cap V_i(\mathcal{H}') \neq \varnothing$ for every $i \in [k-1]$.

Let us now analyze how the size of a vertex cover of $\mathcal{H}$ transforms. The cases (i) and (ii) from Theorem 6.4.3 transform into the following. For every $\delta > 0$, it is UG-hard to decide which of the cases (iii) and (iv) hold.

(iii) Every vertex cover of $\mathcal{H}'$ has size at least

$$|V(\mathcal{H})|\left(\frac{1}{2(k-1)}-\delta\right)+\frac{\varepsilon}{1-\varepsilon}\frac{|V_k(\mathcal{H})|}{k}.$$

(iv) The cardinality of a minimum vertex cover is at most

$$|V(\mathcal{H})|\left(\frac{1}{k(k-1)}+\delta\right)+\frac{\varepsilon}{1-\varepsilon}\frac{|V_k(\mathcal{H})|}{k}.$$

Let us introduce $n = |V(\mathcal{H})|$. It implies that for every $\delta > 0$, it is UG-hard to approximate the VERTEX COVER problem in ε-dense k-partite k-

hypergraphs to within any constant approximation ratio less than $R(\delta)$, where $R(\delta)$ is defined as follows.

$$R(\delta) = \frac{n\left(\frac{1}{2(k-1)} - \delta\right) + \frac{\varepsilon}{1-\varepsilon}\frac{|V_k(\mathcal{H})|}{k}}{n\left(\frac{1}{k(k-1)} + \delta\right) + \frac{\varepsilon}{1-\varepsilon}\frac{|V_k(\mathcal{H})|}{k}}$$

Due to our assumption on the sizes of the k-partitions $\{V_i(\mathcal{H}) \mid i \in [k]\}$, we have $k \cdot |V_k(\mathcal{H})| \leq n$. We are going to deduce a lower bound on $R(\delta)$.

$$\begin{aligned}
R(\delta) &\geq \frac{\frac{(1-\varepsilon)n}{2(k-1)} - \delta(1-\varepsilon)n + \frac{\varepsilon n}{k}}{\frac{(1-\varepsilon)n}{k(k-1)} + \delta(1-\varepsilon)n + \frac{\varepsilon n}{k}} = \frac{\frac{(1-\varepsilon)k}{2(k-1)k} + \frac{2\varepsilon(k-1)}{2k(k-1)}}{\frac{1-\varepsilon}{(k-1)k} + \frac{\varepsilon(k-1)}{k(k-1)}} - \delta' \\
&= \frac{\frac{k - \varepsilon k + 2\varepsilon k - 2\varepsilon}{2(k-1)k}}{\frac{1 - \varepsilon + \varepsilon k - \varepsilon}{(k-1)k}} - \delta' = \frac{k + (k-2)\varepsilon}{2(1 + (k-2)\varepsilon)} - \delta' \\
&= \frac{k}{\frac{2k(1+(k-2)\varepsilon)}{k+(k-2)\varepsilon}} - \delta'
\end{aligned}$$

The last term can be simplified in the following way in order to obtain the desired inapproximability factor.

$$\begin{aligned}
\frac{k}{\frac{2k(1+(k-2)\varepsilon)}{k+(k-2)\varepsilon}} - \delta' &= \frac{k}{\frac{2k + 2(k-2)\varepsilon + (2k-2)(k-2)\varepsilon}{k+(k-2)\varepsilon}} - \delta' \\
&= \frac{k}{2 + \frac{(2k-2)(k-2)\varepsilon}{k+(k-2)\varepsilon}} - \delta' \\
&= \frac{k}{2 + \frac{2(k-1)(k-2)\varepsilon}{k+(k-2)\varepsilon}} - \delta'
\end{aligned}$$

Finally, we have to verify that the constructed k-partite k-hypergraph $\mathcal{H}'$ is indeed ε-dense.

$$\frac{|E(\mathcal{H}')|}{\prod\limits_{i\in[k]}|V_i(\mathcal{H}')|} \geq \frac{\left(\frac{\varepsilon|V_k(\mathcal{H})|}{1-\varepsilon}\right)\prod\limits_{i\in[k-1]}|V_i(\mathcal{H}')|}{\left(|V_k(\mathcal{H})|+\frac{\varepsilon|V_k(\mathcal{H})|}{1-\varepsilon}\right)\prod\limits_{i\in[k-1]}|V_i(\mathcal{H}')|} = \frac{\frac{\varepsilon}{1-\varepsilon}}{\left(\frac{1-\varepsilon}{1-\varepsilon}+\frac{\varepsilon}{1-\varepsilon}\right)} = \varepsilon$$

Note that the constructed k-partite k-hypergraph is also (ε,ℓ)-dense for all $\ell > 0$. Hence, we obtain the same inapproximability factor for those cases finishing the proof of Theorem 6.5.3. ■

An Optimal Inapproximability Result

Next, we combine the former construction with a conjecture on the UG-hardness of the VERTEX COVER problem in k-partite k-hypergraphs. In particular, we postulate the following.

Conjecture 6.5.1
Let $\mathcal{H}$ be a k-partite k-hypergraph with given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$ and $k \geq 3$. Then, for every $\delta > 0$, the following is UG*-hard to decide.*

(i) *Every vertex cover of $\mathcal{H}$ has size at least*

$$|V(\mathcal{H})|\left(\frac{1}{k}-\delta\right).$$

(ii) *The cardinality of a minimum vertex cover of $\mathcal{H}$ is at most*

$$|V(\mathcal{H})|\left(\frac{2}{k^2}+\delta\right).$$

Combining Conjecture 6.5.1 with the construction in Theorem 6.5.3, it yields the following inapproximability result, which matches asymptotically the approximation upper bound achieved by our approximation algorithm described in Section 6.5.3. In addition, it indicates that further densification in the sense of (ε,ℓ)-density does not affect the approximability of the underlying problem in contrast to the VERTEX COVER problem in (ε,ℓ)-dense k-hypergraphs.

We now prove Theorem 6.5.4 restated below.

Theorem 6.5.4
For every $k \geq 3$, it is UG*-hard to approximate the* VERTEX COVER *problem to within any constant approximation ratio less than*

$$\frac{k}{2+(k-2)\varepsilon}$$

in (ε, ℓ)-dense k-partite hypergraphs for all $\ell \in [k-1]$ assuming Conjecture 6.5.1.

Proof of Theorem 6.5.4.
Let us consider the k-partite k-hypergraph $\mathcal{H}$ from Conjecture 6.5.1. Then, we fix a constant $\varepsilon \in (0,1)$. Assuming Conjecture 6.5.1, the following promise problem is UG-hard to decide.

(i) Every vertex cover of $\mathcal{H}$ has size at least

$$|V(\mathcal{H})|\left(\frac{1}{k}-\delta\right).$$

(ii) The cardinality of a minimum vertex cover of $\mathcal{H}$ is at most

$$|V(\mathcal{H})|\left(\frac{2}{k^2}+\delta\right).$$

Utilizing $\mathcal{H}$ as a hard instance combined with the construction in Theorem 6.5.3, we obtain the k-partite k-hypergraph $\mathcal{H}'$, for which the following is UG-hard to decide for every $\delta > 0$.

(iii) Every vertex cover of $\mathcal{H}'$ has size at least

$$|V(\mathcal{H})|\left(\frac{1}{k}-\delta\right)+\frac{\varepsilon}{1-\varepsilon}\cdot|V_k(\mathcal{H})|.$$

(iv) The cardinality of a minimum vertex cover of $\mathcal{H}'$ is at most

$$|V(\mathcal{H})|\left(\frac{2}{k^2}+\delta\right)+\frac{\varepsilon}{1-\varepsilon}\cdot|V_k(\mathcal{H})|.$$

Let us denote $n = |V(\mathcal{H})|$. It implies the UG-hardness of approximating VERTEX COVER problem in ε-dense k-partite k-hypergraphs for every $\delta > 0$ to within

$$\begin{aligned}
\frac{n\left(\frac{1}{k} - \delta\right) + \frac{\varepsilon}{1-\varepsilon} \cdot |V_k(\mathcal{H})|}{n\left(\frac{2}{k^2} + \delta\right) + \frac{\varepsilon}{1-\varepsilon} \cdot |V_k(\mathcal{H})|} &\geq \frac{n\left(\frac{1}{k} - \delta\right)(1-\varepsilon) + \frac{\varepsilon n}{k}}{n\left(\frac{2}{k^2} + \delta\right)(1-\varepsilon) + \frac{\varepsilon n}{k}} \qquad \left(\text{by } |V_k| \leq \frac{n}{k}\right) \\
&= \frac{k - \delta(1-\varepsilon)k^2}{2(1-\varepsilon) + \delta(1-\varepsilon)k^2 + k\varepsilon} \\
&= \frac{k}{2 + (k-2)\varepsilon} - \delta'.
\end{aligned}$$

Finally, we note that the resulting k-partite k-hypergraph $\mathcal{H}'$ is (ε, ℓ)-dense for all $\ell \in [k-1]$. ■

NP-Hardness of Approximation

Combining our reduction from Theorem 6.4.3 with the hard instance given in Theorem 6.4.4, we are going to prove the following **NP**-hardness of approximation result.

Theorem 6.5.5

For every $k \geq 4$, it is **NP***-hard to approximate the* VERTEX COVER *problem in (ε, ℓ)-dense k-partite k-hypergraphs to within any constant approximation ratio less than*

$$\frac{(k - 1 + 3\varepsilon(k+1))(k-1)}{2(k + \varepsilon(2k^2 - 2k - 1))}$$

for all $\ell \in [k-1]$.

Proof of Theorem 6.5.5.

Let us consider the k-partite k-hypergraph $\mathcal{H}$ with n vertices constructed in Theorem 6.4.4, for which the following two cases (i) and (ii) are **NP**-hard to decide for every $\delta > 0$.

(i) Every vertex cover of $\mathcal{H}$ has size at least

$$|V(\mathcal{H})|\left(\frac{k-1}{2k(2k+1)} - \delta\right).$$

(ii) The cardinality of a minimum vertex cover of $\mathcal{H}$ is at most

$$|V(\mathcal{H})|\left(\frac{1}{(k-1)(2k+1)}+\delta\right).$$

Let us fix a constant $\varepsilon \in (0,1)$. Utilizing $\mathcal{H}$ as a hard instance combined with the construction in Theorem 6.5.3, we obtain the ε-dense k-partite k-hypergraph $\mathcal{H}'$, for which the following is **NP**-hard to decide for every $\delta > 0$.

(iii) Every vertex cover of $\mathcal{H}'$ has size at least

$$|V(\mathcal{H})|\left(\frac{k-1}{2k(2k+1)}-\delta\right)+\frac{\varepsilon}{1-\varepsilon}\cdot|V_k(\mathcal{H})|.$$

(iv) The cardinality of a minimum vertex cover of $\mathcal{H}'$ is at most

$$|V(\mathcal{H})|\left(\frac{1}{(k-1)(2k+1)}+\delta\right)+\frac{\varepsilon}{1-\varepsilon}\cdot|V_k(\mathcal{H})|.$$

It implies the **NP**-hardness of approximating the VERTEX COVER problem in ε-dense k-partite k-hypergraphs for every $\delta > 0$ to within any constant approximation ratio better than $R(\delta)$, where $R(\delta)$ is defined as follows.

$$R(\delta) = \frac{|V(\mathcal{H})|\left(\frac{k-1}{2k(2k+1)}-\delta\right)+\frac{\varepsilon}{1-\varepsilon}\cdot|V_k(\mathcal{H})|}{|V(\mathcal{H})|\left(\frac{1}{(k-1)(2k+1)}+\delta\right)+\frac{\varepsilon}{1-\varepsilon}\cdot|V_k(\mathcal{H})|} \tag{6.28}$$

The inapproximability factor $R(\delta)$ can be simplified in the following way.

$$\begin{aligned} R(\delta) &= \frac{\frac{(k-1)(1-\epsilon)}{2k(2k+1)}+\frac{\epsilon}{k}}{\frac{1-\epsilon}{(k-1)(2k+1)}+\frac{\epsilon}{k}}-\delta' \\ &= \frac{\frac{k-1+3\varepsilon(k+1)}{2k(2k+1)(1-\varepsilon)}}{\frac{k+\varepsilon(2k^2-2k-1)}{(1-\varepsilon)k(k-1)(2k+1)}}-\delta' \\ &= \frac{(k-1+3\varepsilon(k+1))(k-1)}{2(k+\varepsilon(2k^2-2k-1))}-\delta' \end{aligned}$$

Finally, we note that the constructed k-partite k-hypergraph is (ε,ℓ)-dense for all $\ell \in [k-1]$. ■

6.6 Nearly Regular k-Partite k-Hypergraphs

In this section, we extend the range of the density of the considered k-hypergraph and investigate the approximability of the VERTEX COVER problem in nearly regular k-partite k-hypergraphs. Especially, for subdense and mildly sparse instances, we design a randomized approximation algorithm with approximation ratio strictly less than $k/2$ running in polynomial and quasi-polynomial time, respectively. On the other hand, we obtain a hardness of approximation result for the problem proving the optimality of the approximation ratio of our algorithm based on Conjecture 6.5.1.

6.6.1 Contribution

By combining the techniques developed in Section 5.6.3 for the VERTEX COVER problem in nearly regular k-hypergraphs with a new method called randomized bucketing extraction, we design a randomized approximation algorithm with approximation ratio strictly less than $k/2$ and running time depending on the density of the nearly regular k-hypergraph. More precisely, we prove the following theorem.

Theorem 6.6.1
For every $\varepsilon > 0$ and $k \geq 3$, there is a randomized approximation algorithm which computes a vertex cover for a given r-nearly regular k-partite k-hypergraph $\mathcal{H}$ with n vertices and given k-partition with approximation ratio

$$\frac{k}{2+(k-2)\cdot r}+\varepsilon$$

in $\mathrm{poly}(n)\exp\left[\mathrm{O}(\,\Psi_{\mathcal{H}}(n)\,)\right]$ *time.*

The crucial ingredient of our approximation algorithm is the randomized bucketing extraction method which could be also applicable to related covering problems with density constraints.

As for lower bounds, we obtain a tight approximation hardness result proving the optimality of the approximation ratio of our algorithm under Conjecture 6.5.1. Let us formulate the precise statement.

Theorem 6.6.2

For every $k \geq 3$, it is UG *-hard to approximate the* VERTEX COVER *problem in r-nearly regular k-balanced hypergraphs with $|E(\mathcal{H})| = \Omega(n^{\varepsilon})$ for every $\varepsilon > 0$ to within any constant approximation ratio less than*

$$\frac{k}{2 + (k-2) \cdot r}$$

assuming Conjecture 6.5.1.

6.6.2 The Randomized Bucketing Algorithm

This section is devoted to the proof of Theorem 6.6.1. For this reason, we first introduce the notation that will be needed and give a high level view of the proof of Theorem 6.6.1.

Conventions and Outline of the Proof

Similarly to the approach developed in Section 5.6.3 for the VERTEX COVER problem in nearly regular k-hypergraphs, we iteratively remove small subsets of a minimum vertex cover of a given nearly regular k-partite k-hypergraph. However, the randomized extraction technique used in Section 5.6.3 cannot be applied successfully and we have to establish a more sophisticated method to overcome this problem. We propose the randomized bucketing extraction method which will be described later on.

Starting with a r-nearly regular k-partite k-hypergraph $\mathcal{H}$, we first iteratively remove and collect vertex subsets until a sufficiently small set of vertices remain. Let us first suppose that at every iteration i of the algorithm, we are able to guess a sufficiently large subset of an optimal solution of the current k-partite k-hypergraph $\mathcal{H}_i$. This subset of vertices is removed together with the edges that they cover to form $\mathcal{H}_{i+1}$. We will see in the next subsection how we can sample the set $\widetilde{M}$ computed by the improved extraction algorithm in Section 6.5.3 to perform this guessing step efficiently. The union of the removed sets will form the set M allowing us to use Lemma 6.5.2. All in all, we aim at obtaining a subset M of a minimum vertex cover of $\mathcal{H}$

having cardinality approximately $r \cdot |V_k(\mathcal{H})|$.

Letting $\mathcal{H}_i$ be the k-partite k-hypergraph considered at the i-th step ($\mathcal{H}_1 = \mathcal{H}$), we denote by n_i its number of vertices, by $E(\mathcal{H}_i)$ its edge set. Furthermore, we define the parameter $\Psi_i(n_i)$ as follows.

$$\Psi_i(n_i) = \left(\prod_{j \in [k]} |V_j(\mathcal{H}_i)| \right) (|E(\mathcal{H}_i)|)^{-1}$$

We denote by $\{V_j(\mathcal{H}_i) \mid j \in [k]\}$ its given vertex partition and introduce the parameter s_i defined below.

$$s_i = n_i - (1-r)|V_k(\mathcal{H}_i)| - \sum_{j \in [k-1]} |V_j(\mathcal{H}_i)|$$

In the case $s_i = 0$, we obtain that

$$n_i = (1-r)|V_k| + \sum_{i \in [k-1]} |V_i| \Rightarrow n_i = n - r|V_k|.$$

Since $n - n_i$ is the size of the extracted set M, s_i can serve as a measure of progress of the procedure. At every step, we remove $\alpha|V_k(\mathcal{H})|/\Psi_{\mathcal{H}}(n)$ vertices until $s_i \leq \alpha|V_k(\mathcal{H})|$ for a small constant $\alpha > 0$ specified later on. Thus, at the end of the procedure, we will have $s_i \leq \alpha|V_k(\mathcal{H})|$ implying $|M| \geq (r-\alpha)|V_k(\mathcal{H})|$. We now show that we can always find a set of this size contained in a minimum vertex cover.

Randomized Bucketing Extraction

By utilizing our randomized bucketing extraction method, we turn algorithm $\mathcal{A}_{6.7}$ into a randomized version for extracting a part of a minimum vertex cover of a given k-partite k-hypergraph while introducing only a constant number of candidate sets. In order to define the method and describe our algorithm, we are going to introduce some notation.

For a fixed integer $k \geq 2$, we introduce for every constant $\gamma \in (2/3, 1)$ the scaled probability $\zeta(\gamma) \in (2/3, 1)$ defined as follows.

$$\zeta(\gamma) = 1 - \frac{1-\gamma}{k} \tag{6.29}$$

For a given set of vertices, we are going to introduce the corresponding buckets. Let $B_t = \{v_1, \ldots, v_t\}$ be a set of t vertices of a given k-hypergraph $\mathcal{H}$. For every $l \in [t]$ and $q \in [f]$ with $f = \lfloor t \cdot l^{-1} \rfloor$, we introduce the set $B_{t,l}(q) \subseteq B_t$, where

$$B_{t,l}(q) \;=\; \left\{ v_j \;\middle|\; j \in \{1 + (q-1) \cdot l, \ldots, l + (q-1) \cdot l\} \right\}.$$

Furthermore, we define $B^+_{t,l}(0) = \varnothing$. For every $q \in [f]$, we introduce the set $B^+_{t,l}(q)$ and $B_{t,l}(f+1)$ given by

$$B^+_{t,l}(q) = \bigcup_{i \in [q]} B_{t,l}(i) \text{ and } B_{t,l}(f+1) = B_t \backslash B^+_{t,l}(f).$$

For every $t \in \mathbb{N}$ and $\gamma \in (2/3, 1)$, we define the interval length $L(t,\gamma)$ and the number of intervals $n(t,\gamma)$ by

$$L(t,\gamma) = \max\{1, \lfloor t(1 - \zeta(\gamma)) \rfloor\} \text{ and } n(t,\gamma) = \left\lceil \frac{t}{L(t,\gamma)} \right\rceil, \text{ respectively.} \tag{6.30}$$

Finally, for every $k \geq 2$ and $\gamma \in (2/3, 1)$, we introduce the sample size $\widehat{s}_k(\gamma)$ defined as follows.

$$\widehat{s}_k(\gamma) = \frac{\log\left(1 - \gamma^{(k-1)^{-1}}\right)}{\log(\zeta(\gamma))} \tag{6.31}$$

Before we define the algorithm for the extraction, we describe briefly the main idea of the method. Let $\mathcal{H}$ be a dense k-partite k-hypergraph and C a minimum vertex cover of $\mathcal{H}$. Recall from algorithm $\mathcal{A}_{6.7}$ that for a given set of vertices $R_p \subseteq V_k(\mathcal{H})$ with $R_p \notin C$, we had to find the highest degree vertex $v \in R_p$ with $v \notin C$. This task was accomplished by exhaustive search. As we want to decrease running time, we will solve this problem approximately. We divide the vertices in R_p into disjoint buckets $B_i \subseteq R_p$ according to their degree in $\mathcal{H}$. For a constant $\beta \in (0,1)$, we have either $|R_p \cap C| \geq \beta |R_p|$ or there is a bucket B_j with $|B_i \cap C| < \beta |B_i|$. In the latter case, we have to find the bucket B_k with the highest degree vertices and $|B_k \cap C| < \beta |B_k|$. Then, we have $|(B_1 \cup \ldots B_{k-1}) \cap C| \geq \beta |B_1 \cup \ldots B_{k-1}|$ and we can choose a vertex $v \in B_k$ such that $v \notin C$ with probability at most $(1 - \beta)$.

We present the randomized extraction algorithm defined in Figure 6.10.

```
Algorithm $\mathcal{A}_{6.10}$

Input : $(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, t, \gamma, l, \widehat{s}_k(\gamma))$ with $k$-partition
        $\{V_i(\mathcal{H}) \mid i \in [k]\}$, where $\mathcal{H}$ is a $k$-partite $k$-hypergraph,
        $t \in \mathbb{N}$ with $t \le |V_k(\mathcal{H})|/\Psi_{\mathcal{H}}(n)$, $\gamma \in (2/3, 1)$, $l \in \mathbb{N}$ with
        $l \le L(t, \gamma)$ and $\widehat{s}_k(\gamma)$ the sample size.
Output: A collection $\widetilde{W}$ of subsets $W_i \subseteq V(\mathcal{H})$.

begin
    ① $\widetilde{W} \leftarrow \varnothing$;
    if $k = 1$ then
        ② Find a set $A \subseteq V_1(\mathcal{H})$ containing $t$ arbitrary vertices of $\mathcal{H}$;
        ③ $\widetilde{W} \leftarrow \{A\}$;
        return $\widetilde{W}$;
    else
        ④ Find a set $\{v_1, \ldots, v_t\}$ containing $t$-heaviest vertices of
        $V_k(\mathcal{H})$ with $d_{\mathcal{H}}(v_i) \ge d_{\mathcal{H}}(v_{i+1})$ for all $i \in [t-1]$;
        ⑤ $B_t \leftarrow \{v_1, \ldots, v_t\}$;
        ⑥ $\widetilde{W} \leftarrow \widetilde{W} \cup \{B_t\}$;
        ⑦ $f \leftarrow \left\lceil \frac{t}{l} \right\rceil$;
        foreach $q \in [f]$ do
            for $i = 1, \ldots, \widehat{s}_k(\gamma)$ do
                ⑧ Choose $v_j \in B_{t,l}(q)$ uniformly at random;
                ⑨ $\widetilde{W}' \leftarrow \mathcal{A}_{6.10}(\mathcal{H}(v_j), t - (j-1), \gamma, l, \widehat{s}_k(\gamma))$;
                ⑩ $\widetilde{W} \leftarrow \widetilde{W} \cup \left\{\{v_1, \ldots, v_{j-1}\} \cup A \mid A \in \widetilde{W}'\right\}$;
            end
        end
    end
    return $\widetilde{W}$;
end
```

Figure 6.10: Algorithm $\mathcal{A}_{6.10}$

We are going to prove the following lemma.

Lemma 6.6.1
Let $\mathcal{H}$ be a k-partite k-hypergraph with given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$, C a minimum vertex cover of $\mathcal{H}$, $t \in \mathbb{N}$ with $t \leq |V_k(\mathcal{H})|/\Psi_{\mathcal{H}}(n)$, $l \in \mathbb{N}$ with $l \leq L(t,\gamma)$ and a constant $\gamma \in (2/3, 1)$. On input $(\mathcal{H}, t, \gamma, l, \widehat{s}_k(\gamma))$, algorithm $\mathcal{A}_{6.10}$ constructs in polynomial time a collection $\widetilde{W}$ of subsets $W_i \subseteq V(\mathcal{H})$ with the following properties.

(i) Every $W_i \in \widetilde{W}$ has size $|W_i| = t$.

(ii) There is a $W_j \in \widetilde{W}$ with $|W_j \cap C| \geq \gamma \cdot t$ with probability at least γ.

(iii) The cardinality of $\widetilde{W}$ can be bounded from above as follows.

$$|\widetilde{W}| \leq \left(\left\lceil \frac{t}{l} \right\rceil \cdot \widehat{s}_k(\gamma) + 1\right)^{(k-1)}$$

Proof of Lemma 6.6.1.
Let $\mathcal{H}$ be a k-partite k-hypergraph with given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$ and C a minimum vertex cover of $\mathcal{H}$. Furthermore, we fix a constant $\gamma \in (2/3, 1)$, $t \in \mathbb{N}$ with

$$t \leq \frac{|V_k(\mathcal{H})|}{\Psi_{\mathcal{H}}(n)} \tag{6.32}$$

and $l \in \mathbb{N}$ with $l \leq L(t,\gamma)$. Let $B_t = \{v_1, \ldots, v_t\}$ be a set containing t-heaviest vertices of $V_k(\mathcal{H})$ with $d_{\mathcal{H}}(v_i) \geq d_{\mathcal{H}}(v_{i+1})$ for all $i \in [t-1]$. Notice that by definition of $\zeta(\gamma)$ in (6.29), we obtain $\zeta(\gamma) \geq \gamma$. Due to step ⑥ in the description of algorithm $\mathcal{A}_{6.10}$, we have that $B_t \in \widetilde{W}$. Consequently, in the case, in which we have $|B_t \cap C| \geq \zeta(\gamma) \cdot |B_t| \geq \gamma \cdot |B_t|$, we have nothing to prove.

Since we may assume that $|B_t \cap C| < \zeta(\gamma) \cdot |B_t|$ holds, there is an integer $u' \in [n(t,\gamma)]$ with $|B_{t,l}(u') \cap C| < \zeta(\gamma) \cdot |B_{t,l}(u')|$. Let $u \in [n(t,\gamma)]$ be the smallest index in $[n(t,\gamma)]$ with the property

$$|B_{t,l}(u) \cap C| < \zeta(\gamma) \cdot |B_{t,l}(u)|. \tag{6.33}$$

By the definition of u, we obtain the following.

$$\left|B^+_{t,l}(u-1) \cap C\right| = \bigcup_{i \in [u-1]} |B_{t,l}(i) \cap C| \geq \zeta(\gamma) \cdot \left| \bigcup_{i \in [u-1]} B_{t,l}(i) \right| = \zeta(\gamma) \cdot \left|B^+_{t,l}(u-1)\right|$$

Let $B_{t,l}(u)$ be given by $B_{t,l}(u) = \{x, \ldots, y\}$ with $d_{\mathcal{H}}(y) \le d_{\mathcal{H}}(v)$ for all $v \in B_{t,l}(u)$. In the worst case, there is no vertex in $B_{t,l}(u)$ that is also contained is C. In addition to that, in step ⑧ in the description of algorithm $\mathcal{A}_{6.10}$, we choose the vertex y. By $\left|B^{+}_{t,l}(u-1) \cap C\right| \ge \zeta(\gamma) \cdot \left|B^{+}_{t,l}(u-1)\right|$ and $B_{t,l}(u) \cap C = \varnothing$, we obtain

$$L(\gamma) + (1 - \zeta(\gamma)) \cdot \left|B^{+}_{t,l}(u-1)\right| \;\ge\; |B_{t,l}(u)\backslash C| + \left|B^{+}_{t,l}(u-1)\backslash C\right| \qquad (6.34)$$

after one iteration. We are going to prove that it is possible to extract $t' = t - |B^{+}_{t,l}(u)| + 1$ vertices of $C\backslash B^{+}_{t,l}(u)$ by proceeding on the $(k-1)$-partite $(k-1)$-hypergraph $\mathcal{H}(y)$. For this reason, let us analyze the density of the hypergraph $\mathcal{H}(y)$. The degree of y in $\mathcal{H}$ is at least

$$\frac{\left(\dfrac{|V_k(\mathcal{H})|}{\Psi_{\mathcal{H}}(n)} - |B^{+}_{t,l}(u)| + 1\right)}{|V_k(\mathcal{H})| - |B^{+}_{t,l}(u)| + 1}\left(\prod_{i \in [k-1]} |V_i(\mathcal{H})|\right).$$

This implies that the density of the hypergraph $\mathcal{H}(y)$ is bounded from below by

$$\frac{\left(\dfrac{|V_k(\mathcal{H})|}{\Psi_{\mathcal{H}}(n)} - |B^{+}_{t,l}(u)| + 1\right)}{\left(|V_k(\mathcal{H})| - |B^{+}_{t,l}(u)| + 1\right)}$$

By the definition of $\mathcal{H}(y)$, there is a vertex cover C_y of $\mathcal{H}(y)$ that is contained in C. Since we want to extract $t' = t - |B^{+}_{t,l}(u)| + 1$ vertices from $V(\mathcal{H}(y))$, we have to prove that $|C_y| \ge t'$. According to Lemma 6.5.1, the size of C_y is bounded from below by the product of the density of $\mathcal{H}(y)$ and the cardinality of $V_{k-1}(\mathcal{H}(y))$. Consequently, we obtain

$$\begin{aligned}
|C_y| \;&\ge\; \frac{\dfrac{|V_k(\mathcal{H})|}{\Psi_{\mathcal{H}}(n)} - |B^{+}_{t,l}(u)| + 1}{|V_k(\mathcal{H})| - |B^{+}_{t,l}(u)| + 1} \cdot |V_{k-1}(\mathcal{H}(y))| \\
&\ge\; \frac{\dfrac{|V_k(\mathcal{H})|}{\Psi_{\mathcal{H}}(n)} - |B^{+}_{t,l}(u)| + 1}{|V_k(\mathcal{H})| - |B^{+}_{t,l}(u)| + 1} \cdot |V_k(\mathcal{H})| \\
&\ge\; \frac{|V_k(\mathcal{H})|}{\Psi_{\mathcal{H}}(n)} - |B^{+}_{t,l}(u)| + 1 \\
&\ge\; t - |B^{+}_{t,l}(u)| + 1.
\end{aligned}$$

Thus, by recursively removing vertices from $\mathcal{H}$, algorithm $\mathcal{A}_{6.10}$ constructs a collection $\widetilde{W}$ of subsets $W_i \subseteq V(\mathcal{H})$ each having size t.

We are going to prove that there is a $W_j \in \widetilde{W}$ such that $|W_j \cap C| \geq t \cdot \gamma$ with probability at least γ. Let us first suppose that we obtain in each iteration a vertex $u \notin C$. By (6.34), after $(k-1)$ iterations, we have extracted at least one set $W_j \in \widetilde{W}$ including at most $(k-1)L(t,\gamma) + (1-\zeta(\gamma))t$ vertices that are not contained in C. Note that

$$\begin{aligned}(k-1)L(t,\gamma) + (1-\zeta(\gamma))t &= (k-1)\lfloor(1-\zeta(\gamma))t\rfloor + (1-\zeta(\gamma))t \\ &\leq k \cdot (1-\zeta(\gamma))t.\end{aligned}$$

By the definition of $\zeta(\gamma)$ in (6.29), we deduce that $k \cdot (1-\zeta(\gamma))t \leq (1-\gamma) \cdot t$ implying $|W_j \cap C| \geq \gamma \cdot t$.

Next, we are going to prove that the probability of extracting a subset of C having size at least $\gamma \cdot t$ is at least γ. By (6.33), the probability that a random vertex of $B_{t,l}(u)$ belongs to C is at most $\zeta(\gamma)$. Thus, with probability at least $1 - (\zeta(\gamma))^{\widehat{s}_k(\gamma)}$, we get a vertex $y \notin C$ in the selected sample, and C must contain a vertex cover of the $(k-1)$-partite $(k-1)$-hypergraph $\mathcal{H}(y)$ defined by y. By iterating this step at most $(k-1)$ times, we get in worst case that the probability is at least

$$\left(1 - [\zeta(\gamma)]^{\widehat{s}_k(\gamma)}\right)^{(k-1)} \geq \gamma,$$

due the definition of $\widehat{s}_k(\gamma)$.

Finally, we will bound the size of the collection $\widetilde{W}$. Since we generate at most $\lceil t/l \rceil$ buckets together with a set consisting of the union of all in each iteration created buckets and choose randomly $\widehat{s}_k(\gamma)$ vertices from each bucket, the size of $\widetilde{W}$ can be bounded from above by $(\lceil t/l \rceil \cdot \widehat{s}_k(\gamma) + 1)^{k-1}$. Due to the choice of the parameter values, we deduce that $\widetilde{W}$ can be constructed efficiently and the proof follows. ■

Deriving a Uniform Lower Bound

Due to the previous lemma, we are able to extract efficiently a large part of a minimum vertex cover of a given k-partite k-hypergraph $\mathcal{H}_i$ while introducing only a constant number of candidate sets. Unfortunately, the number of extracted vertices depends on the density of $\mathcal{H}_i$ and cardinality of $V_k(\mathcal{H}_i)$. As both could decrease in the extraction process, we want to deduce a lower bound on the number of extracted vertices of algorithm $\mathcal{A}_{6.10}$ in terms of the density and number of vertices of the original k-hypergraph. More precisely, we are going to prove the following lemma.

Lemma 6.6.2

Let $\mathcal{H}$ be a r-nearly regular k-partite k-hypergraph with n vertices, given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$ and maximum degree $\Delta_{\mathcal{H}}$. For a constant $\alpha \in (0, r)$, let V_i be a subset of $V(\mathcal{H})$ with $|V_i| \le (r-\alpha)|V_k(\mathcal{H})|$. For every k-partite k-hypergraph $\mathcal{H}_i$ defined by $\mathcal{H}_i = \mathcal{H}[V(\mathcal{H})\backslash V_i]$ with n_i vertices, the following inequality holds.

$$\frac{|V_k(\mathcal{H}_i)|}{\Psi_{\mathcal{H}_i}(n_i)} \ \ge\ \alpha \cdot \frac{|V_k(\mathcal{H})|}{\Psi_{\mathcal{H}}(n)}$$

Proof of Lemma 6.6.2.

Let $\mathcal{H}$ be a r-nearly regular k-partite k-hypergraph with n vertices and given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$. We fix a constant $\alpha \in (0, r)$ and a subset $V_i \subseteq V(\mathcal{H})$ with $|V_i| \le (r-\alpha)|V_k(\mathcal{H})|$. Then, we define $\mathcal{H}_i$ by $\mathcal{H}_i = \mathcal{H}[V(\mathcal{H})\backslash V_i]$ with $|V(\mathcal{H}_i)| = n_i$. Furthermore, we introduce the parameter s_i defined as follows.

$$s_i \ = \ n_i - (1-r)|V_k(\mathcal{H})| - \sum_{i \in [k-1]} |V_i(\mathcal{H})| \tag{6.35}$$

First of all, we will bound the number of edges of the k-hypergraph from below and prove that $|E(\mathcal{H}_i)| \ \ge \ s_i \cdot \Delta_{\mathcal{H}}$ holds. For this reason, we introduce the k-partite k-hypergraph $\mathcal{H}'$ defined by $V(\mathcal{H}') = V(\mathcal{H})$ and $E(\mathcal{H}') = E(\mathcal{H})\backslash E(\mathcal{H}_i)$. As by definition all the vertices of $\mathcal{H}_i$ form an independent set in $\mathcal{H}'$, the size of an maximum independent set of $\mathcal{H}$ is at least $n_i = s_i + (1-r)|V_k(\mathcal{H})| + \sum_{i \in [k-1]} |V_i(\mathcal{H})|$. According to Lemma 5.6.4, we know that the number of edges of $\mathcal{H}'$ is bounded from above by

$$\Delta_{\mathcal{H}}\left(n - \left(s_i + (1-r)|V_k(\mathcal{H})| + \sum_{i \in [k-1]} |V_i(\mathcal{H})|\right)\right) = \Delta_{\mathcal{H}}(r|V_k(\mathcal{H})| - s_i).$$

Consequently, the number of edges of $\mathcal{H}_i$ can be bounded from below by

$$\begin{aligned}|E(\mathcal{H}_i)| &= |E(\mathcal{H})| - |E(\mathcal{H}')| \geq r \cdot \Delta_{\mathcal{H}} \cdot |V_k(\mathcal{H})| - \Delta_{\mathcal{H}}(r \cdot |V_k(\mathcal{H})| - s_i) \\ &= \Delta_{\mathcal{H}} \cdot s_i.\end{aligned}$$

Combining the deduced facts, we conclude that

$$\begin{aligned}\frac{|V_k(\mathcal{H}_i)|}{\Psi_{\mathcal{H}_i}(n_i) \cdot s_i} &= \frac{|E(\mathcal{H}_i)||V_k(\mathcal{H}_i)|}{s_i \prod\limits_{j \in [k]} |V_j(\mathcal{H}_i)|} \geq \frac{\Delta_{\mathcal{H}} \cdot s_i \cdot |V_k(\mathcal{H}_i)|}{s_i \prod\limits_{j \in [k]} |V_j(\mathcal{H}_i)|} \\ &\geq \frac{\Delta_{\mathcal{H}}}{\prod\limits_{j \in [k-1]} |V_j(\mathcal{H}_i)|} \geq \frac{|E(\mathcal{H})|}{|V_k(\mathcal{H})| \prod\limits_{j \in [k-1]} |V_j(\mathcal{H}_i)|} \\ &\geq \frac{|E(\mathcal{H})|}{\prod\limits_{j \in [k]} |V_j(\mathcal{H})|} = \frac{1}{\Psi_{\mathcal{H}}(n)}.\end{aligned}$$

By the definition of $\mathcal{H}_i$, we get $n_i \geq |V(\mathcal{H})| - (r - \alpha)|V_k(\mathcal{H})|$ implying the following.

$$\begin{aligned}s_i &= n_i - (1-r)|V_k(\mathcal{H})| - \sum_{i \in [k-1]} |V_i(\mathcal{H})| \\ &\geq |V(\mathcal{H})| - (r-\alpha)|V_k(\mathcal{H})| - (1-r)|V_k(\mathcal{H})| - \sum_{i \in [k-1]} |V_i(\mathcal{H})| \\ &\geq \alpha \cdot |V_k(\mathcal{H})|\end{aligned}$$

In conclusion, we derive the following inequality finishing the proof.

$$\frac{|V_k(\mathcal{H}_i)|}{\Psi_{\mathcal{H}_i}(n_i)} \geq \frac{s_i}{\Psi_{\mathcal{H}}(n)} \geq \alpha \cdot \frac{|V_k(\mathcal{H})|}{\Psi_{\mathcal{H}}(n)}$$

■

Thus far, we are ready to give the proof of Theorem 6.6.1.

Proof of Theorem 6.6.1

Let $\mathcal{H}$ be a r-nearly regular k-partite k-hypergraph with given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$ and n vertices. Given $\mathcal{H}$, for every constant $\alpha \in (0, r)$, we define the number of iterations $T_{\mathcal{H}}(\alpha)$, where $T_{\mathcal{H}}(\alpha) = \lceil (r/\alpha - 1)\, \Psi_{\mathcal{H}}(n) \rceil$.

We now present the approximation algorithm for the VERTEX COVER problem in nearly regular k-partite k-hypergraphs defined in Figure 6.11.

```
Algorithm $\mathcal{A}_{6.11}$
    Input : $(\mathcal{H}, r, \{V_i(\mathcal{H}) \mid i \in [k]\}, \varepsilon)$, where $\mathcal{H}$ is a $r$-nearly regular
            $k$-partite $k$-hypergraph with $k$-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$
            and $\varepsilon > 0$ a constant.
    Output: A vertex cover $W$ of $\mathcal{H}$.
    begin
        ① $\delta \leftarrow \min\left\{\frac{1}{3}, \frac{\varepsilon}{8k^2}\right\}, \gamma \leftarrow 1 - \delta,\ \alpha \leftarrow \frac{r}{9 \cdot \delta^{-2} + 1}$;
        ② $\widetilde{W} \leftarrow \emptyset, \mathscr{H}^{act} \leftarrow \{\mathcal{H}\},\ \mathscr{H}^{new} \leftarrow \emptyset$;
        ③ $t \leftarrow \left\lceil \alpha \cdot \frac{|V_k(\mathcal{H})|}{\Psi_{\mathcal{H}}(n)} \right\rceil,\ l \leftarrow L(t, \gamma)$;
        for $j = 1, \ldots, T_{\mathcal{H}}(\alpha)$ do
            foreach $\mathcal{H}' \in \mathscr{H}^{act}$ do
                ④ $\widehat{C} \leftarrow \mathcal{A}_{6.10}(\mathcal{H}', \{V_i(\mathcal{H}') \mid i \in [k]\}, t, \gamma, l, \widehat{s_k}(\gamma))$;
                ⑤ $\mathscr{H}^{new} \leftarrow \mathscr{H}^{new} \cup \{\mathcal{H}'[V(\mathcal{H}') \backslash C] \mid C \in \widehat{C}\}$;
            end
            ⑥ $\mathscr{H}^{act} \leftarrow \mathscr{H}^{new}$ ;
            ⑦ $\mathscr{H}^{new} \leftarrow \emptyset$;
        end
        foreach $\mathcal{H}_i \in \mathscr{H}^{act}$ do
            ⑧ $W \leftarrow \mathcal{A}_{6.3}(\mathcal{H}, \{V_i(\mathcal{H}) \mid i \in [k]\}, V(\mathcal{H}) \backslash V(\mathcal{H}_i))$;
            ⑨ $\widetilde{W} \leftarrow \widetilde{W} \cup \{W\}$;
        end
        ⑩ Let $W$ be the smallest set among $\widetilde{W}$ ;
        return $W$;
    end
```

Figure 6.11: Algorithm $\mathcal{A}_{6.11}$

In each iteration, we apply algorithm $\mathcal{A}_{6.10}$ to the actual k-hypergraph

$\mathcal{H}_i$ in order to extract a part of a minimum vertex cover of $\mathcal{H}$. According to Lemma 6.6.2, for a small enough constant $\alpha > 0$ specified later on, the number of extracted vertices is at least $\alpha \cdot |V_k(\mathcal{H})|/\Psi_{\mathcal{H}}(n)$ provided that $s_i \geq \alpha|V_k(\mathcal{H})|$ holds. Then, the number of required iterations in order to extract $(r-\alpha)|V_k(\mathcal{H})|$ vertices is

$$T_{\mathcal{H}}(\alpha) = \left\lceil (r-\alpha)|V_k(\mathcal{H})| \left(\frac{\alpha \cdot |V_k(\mathcal{H})|}{\Psi_{\mathcal{H}}(n)} \right)^{-1} \right\rceil = \left\lceil \left(\frac{r}{\alpha} - 1 \right) \Psi_{\mathcal{H}}(n) \right\rceil. \tag{6.36}$$

Let us fix a constant $\varepsilon > 0$. Then, we determine the values of the parameters δ, γ, and α as follows.

$$\delta = \min\left\{ \frac{1}{3}, \frac{\varepsilon}{8k^2} \right\}, \gamma = 1 - \delta, \text{ and } \alpha = \frac{r}{9 \cdot \delta^{-2} + 1} \tag{6.37}$$

Let $\mathscr{H}_j^{act}$ be the collection of k-hypergraphs constructed by algorithm $\mathcal{A}_{6.11}$ after j iterations, where $j \in [T_{\mathcal{H}}(\alpha)]$. For each $\mathcal{H}_i \in \mathscr{H}_j^{act}$, we consider the set W_j^i defined by $W_j^i = V(\mathcal{H}) \backslash V(\mathcal{H}_i)$. We see that $\mathcal{A}_{6.11}$ performs a exploration of a search tree branching on every subset W_j^i in the collection of candidates $\{W_j^i \cup C \mid C \in \widehat{C} = \mathcal{A}_{6.10}(\mathcal{H}_i, \{V_s(\mathcal{H}_i) \mid s \in [k]\}, t, \gamma, l, \widehat{s}_k(\gamma))\}$. A root-to-leaf path in this tree yields a set $W_{T_{\mathcal{H}}(\alpha)}^i$ defined as the union of all the candidates selected along the path. We now prove that this search tree contains a path yielding a suitable set $W_{T_{\mathcal{H}}(\alpha)}^c$ with probability at least 3/4.

Let C be a minimum vertex cover of $\mathcal{H}$. We are going to prove that for every $\rho \in (0,1)$, algorithm $\mathcal{A}_{6.11}$ constructs a set $W_{T_{\mathcal{H}}(\alpha)}^c$ containing $(r-\alpha)|V_k(\mathcal{H})|$ vertices such that $|W_{T_{\mathcal{H}}(\alpha)}^c \cap C| \geq (1-\rho) \cdot \gamma^2 \cdot |W_{T_{\mathcal{H}}(\alpha)}^c|$ with probability at least

$$1 - \exp\left[-\Psi_{\mathcal{H}}(n)(r/\alpha - 1)\gamma \frac{\rho^2}{2} \right]. \tag{6.38}$$

For each $i \in [T_{\mathcal{H}}(\alpha)]$, we introduce the random variable $X_i \in \{0,1\}$, where $X_i = 1$ corresponds to the event of extracting at least $\gamma \cdot \alpha \cdot |V_k(\mathcal{H})|/\Psi_{\mathcal{H}}(n)$ vertices of C in the i-th iteration of algorithm $\mathcal{A}_{6.11}$. According to Lemma 6.6.1, we know that $\mathsf{Pr}(X_i = 1) \geq \gamma$. By letting

$$X = \sum_{i \in [T_{\mathcal{H}}(\alpha)]} X_i, \tag{6.39}$$

we define $T_{\mathcal{H}}(\alpha)$ independent poisson trials with expected value

$$\mathbb{E}[X] \geq T_{\mathcal{H}}(\alpha) \cdot \gamma = \left(\frac{r}{\alpha} - 1\right) \cdot \gamma \cdot \Psi_{\mathcal{H}}(n)$$

Since we extract at least $\gamma \cdot \alpha \cdot |V_k(\mathcal{H})|/\Psi_{\mathcal{H}}(n)$ vertices of C in every iteration with probability at least γ, the expected number of vertices of $W^c_{T_{\mathcal{H}}(\alpha)}$ that are contained in C is

$$\mathbb{E}[X] \cdot \gamma \cdot \alpha \cdot \frac{|V_k(\mathcal{H})|}{\Psi_{\mathcal{H}}(n)} \geq (r - \alpha) \cdot \gamma^2 \cdot |V_k(\mathcal{H})|$$

By using Chernoff bounds, we derive the claimed statement given in (6.38). In particular, by setting $\rho = \delta$, we obtain a set $W^c_{T_{\mathcal{H}}(\alpha)}$ containing at least

$$(1 - \delta)\gamma^2(r - \alpha)|V_k(\mathcal{H})| \tag{6.40}$$

vertices of C with probability at least

$$1 - \exp\left[-\Psi_{\mathcal{H}}(n)(r/\alpha - 1)\gamma\frac{\delta^2}{2}\right] \geq 1 - \exp[-3] \geq \frac{3}{4}. \tag{6.41}$$

By using (6.40) and the parameter values, the cardinality of $W^c_{T_{\mathcal{H}}(\alpha)} \cap C$ can be further simplified in the following way.

$$\begin{aligned} |W^c_{T_{\mathcal{H}}(\alpha)} \cap C| &= (1 - \delta)\gamma^2(r - \alpha)|V_k(\mathcal{H})| \\ &\geq (1 - \delta)^3 (r - \alpha) \cdot |V_k(\mathcal{H})| \\ &\geq (1 - 3 \cdot \delta) \cdot (r - \alpha) \cdot |V_k(\mathcal{H})| \end{aligned}$$

Let W be the smallest vertex cover among all by algorithm $\mathcal{A}_{6.3}$ constructed vertex covers. Since we apply algorithm $\mathcal{A}_{6.3}$ to each set $W^i_{T_{\mathcal{H}}(\alpha)}$, according to Lemma 6.5.2, we obtain with probability at least 3/4 a vertex cover with approximation ratio at most

$$\begin{aligned} \frac{|W|}{|C|} &\leq \frac{k}{2 + ([1 - 3\delta]k - 2)\dfrac{(r - \alpha)|V_k(\mathcal{H})|}{|V_k(\mathcal{H})|}} \\ &\leq \frac{k}{2 + (k - 2) \cdot r - 4 \cdot \delta k} \\ &\leq \frac{k}{2 + (k - 2) \cdot r} + 4\delta \cdot k^2 \\ &\leq \frac{k}{2 + (k - 2) \cdot r} + \varepsilon \end{aligned}$$

The algorithm $\mathcal{A}_{6.11}$ generates a search tree of height $T_{\mathcal{H}}(\alpha)$ and fan-out less than $(\widehat{s}_k\,(\gamma)\cdot(t/l+2))^{k-1}$. At every node of the tree, algorithm $\mathcal{A}_{6.10}$ is called taking

$$\mathrm{poly}(n) + \mathrm{O}\left(\left(\widehat{s}_k\,(\gamma)\cdot\left(\frac{t}{l}+2\right)\right)^{k-1}\right)$$

time. This implies that the overall running time of algorithm $\mathcal{A}_{6.11}$ is

$$\mathrm{poly}(n)\cdot\mathrm{O}\left(\left(\widehat{s}_k\,(\gamma)\cdot\left(\frac{t}{l}+2\right)\right)^{(k-1)\cdot T_{\mathcal{H}}(\alpha)}\right) = \mathrm{poly}(n)\cdot\exp\left[\mathrm{O}\left(\Psi_{\mathcal{H}}(n)\right)\right]$$

and the proof follows. ■

6.6.3 An Optimal Inapproximability Result

Assuming Conjecture 6.5.1, we give a tight approximation lower bound for the VERTEX COVER problem in nearly regular k-partite k-hypergraphs in a specified range of the density and maximum degree of the given k-hypergraph. In particular, we prove that the approximation ratio achieved by algorithm $\mathcal{A}_{6.11}$ is optimal under this conjecture. Let us restate our theorem.

Theorem 6.6.2

For every $k \geq 3$, it is UG*-hard to approximate the* VERTEX COVER *problem in r-nearly regular k-balanced hypergraphs with $|E(\mathcal{H})| = \Omega(n^{\varepsilon})$ for every $\varepsilon > 0$ to within any constant approximation ratio less than*

$$\frac{k}{2+(k-2)\cdot r}$$

assuming Conjecture 6.5.1.

Proof of Theorem 6.6.2.

As starting point of our reduction, we consider the k-partite k-hypergraph $\mathcal{H}$ with $k \geq 3$ and given k-partition $\{V_i(\mathcal{H}) \mid i \in [k]\}$ from Conjecture 6.5.1. By densifying $\mathcal{H}$, we are going to construct a k-balanced k-hypergraph $\mathcal{H}'$ having the claimed properties. Recall that, assuming Conjecture 6.5.1, the following is UG-hard to decide for every $\delta > 0$.

(i) Every vertex cover of $\mathcal{H}$ has size at least

$$|V(\mathcal{H})|\left(\frac{1}{k}-\delta\right).$$

(ii) The cardinality of a minimum vertex cover of $\mathcal{H}$ is at most

$$|V(\mathcal{H})|\left(\frac{2}{k^2}+\delta\right).$$

Let us fix a constant $r \in (0,1)$ and an integer $j \in \mathbb{N}$. We use $n = |V(\mathcal{H})|$. Then, we construct $\mathcal{H}'$ consisting of $(1-r)kn^j$ disjoint copies of $\mathcal{H}$ and of kn^jr disjoint complete k-balanced cliques of size n. In order to define the k-partition and edge set of $\mathcal{H}'$, we need to introduce some notation. We denote the l-th copy of $\mathcal{H}$ by $\mathcal{H}^l$ with k-partition $\{V_i(\mathcal{H}^l) \mid i \in [k]\}$. In addition, the l-th copy of the clique is denoted by $\mathcal{K}^l$ with k-partition $\{V_i(\mathcal{K}^l) \mid i \in [k]\}$. For every $s \in [k]$, we introduce the sets $V_s^{\mathcal{H}}$ defined as follows.

$$\begin{array}{rcllcll}
V_1^{\mathcal{H}} & = & \{v \in V_i(\mathcal{H}^l) \mid 1 & = & i+l-1 & \pmod{k}\} \\
& \vdots & & \vdots & & \\
V_s^{\mathcal{H}} & = & \{v \in V_i(\mathcal{H}^l) \mid s & = & i+l-1 & \pmod{k}\} \\
& \vdots & & \vdots & & \\
V_{k-1}^{\mathcal{H}} & = & \{v \in V_i(\mathcal{H}^l) \mid k-1 & = & i+l-1 & \pmod{k}\} \\
V_k^{\mathcal{H}} & = & \{v \in V_i(\mathcal{H}^l) \mid 0 & = & i+l-1 & \pmod{k}\}
\end{array}$$

Accordingly, for all $i \in [k]$ and $u \in [k]$, we define the sets $V_{iu}^{\mathcal{K}}$ of vertices, where

$$V_{iu}^{\mathcal{K}} = \left\{v \in V_i(\mathcal{K}^p) \;\middle|\; p = s + (u-1)n^jr \text{ with } s \in [n^jr]\right\}.$$

Then, the vertex partition of the k-partite k-hypergraph $\mathcal{H}'$ is given by

$$\left\{V_i^{\mathcal{H}} \cup \left(\bigcup_j V_i(\mathcal{K}^j)\right) \;\middle|\; i \in [k]\right\}.$$

In order to define the edges of the k-hypergraph $\mathcal{H}'$, we introduce the sets of edges E'_i for $i \in [k]$.

$$\begin{array}{rcll} E'_1 & = & \Big\{\{v_1,\dots,v_k\} & \Big| \;\; v_1 \in V^{\mathcal{K}}_{11}, v_2 \in V^{\mathcal{H}}_2,\dots,v_k \in V^{\mathcal{H}}_k\Big\} \\ \vdots & & & \;\;\vdots \\ E'_i & = & \Big\{\{v_1,\dots,v_k\} & \Big| \;\; v_1 \in V^{\mathcal{H}}_1,\dots,v_i \in V^{\mathcal{K}}_{ii},\dots,v_k \in V^{\mathcal{H}}_k\Big\} \\ \vdots & & & \;\;\vdots \\ E'_k & = & \Big\{\{v_1,\dots,v_k\} & \Big| \;\; v_1 \in V^{\mathcal{H}}_1,\dots,v_{k-1} \in V^{\mathcal{H}}_{k-1}, v_k \in V^{\mathcal{K}}_{kk}\Big\} \end{array}$$

By introducing $N = |V(\mathcal{H}')|$, we obtain $|V_i(\mathcal{H}')| = N/k$ for every $i \in [k]$. The edge set of $\mathcal{H}'$ contains all edges of the copies of $\mathcal{H}$ and the k-partite cliques. Furthermore, by adding as many hyperedges as needed from the set $\bigcup\limits_{i\in[k]} E'_i$, it is possible to construct a r-nearly regular k-balanced hypergraph $\mathcal{H}'$ with $|E(\mathcal{H}')| = \omega\big(N^{\frac{k-1}{j+1}}\big)$. Notice that a vertex cover of $\mathcal{H}'$ must include n/k vertices of each clique. Assuming Conjecture 6.5.1, the following is UG-hard to decide for every $\delta > 0$.

(i) Every vertex cover of $\mathcal{H}'$ has size at least

$$(1-r)\,kn^{j+1}\left(\frac{1}{k}-\delta\right)+\frac{kn^{j+1}r}{k}.$$

(ii) The cardinality of a minimum vertex cover of $\mathcal{H}$ is at most

$$(1-r)\,kn^{j+1}\left(\frac{2}{k^2}+\delta\right)+\frac{kn^{j+1}r}{k}.$$

It implies that for every $\delta > 0$, it is UG-hard to approximate the VERTEX COVER problem to within any constant approximation ratio less than $R(\delta)$, where $R(\delta)$ is defined as follows.

$$R(\delta) \;=\; \frac{(1-r)\,kn^{j+1}\left(\frac{1}{k}-\delta\right)+\frac{kn^{j+1}r}{k}}{(1-r)\,kn^{j+1}\left(\frac{2}{k^2}+\delta\right)+\frac{kn^{j+1}r}{k}}$$

In order to derive the desired inapproximability factor, we are going to deduce a lower bound on $R(\delta)$.

$$
\begin{aligned}
R(\delta) &= \frac{(1-r)\,kn^{j+1}\left(\dfrac{1}{k}-\delta\right)+\dfrac{kn^{j+1}r}{k}}{(1-r)\,kn^{j+1}\left(\dfrac{2}{k^2}+\delta\right)+\dfrac{kn^{j+1}r}{k}} \\
&= \frac{(1-r)\,n^{j+1}+n^{j+1}r-\delta\,(1-r)\,n^{j+1}k}{\dfrac{(1-r)\,n^{j+1}2}{k}+n^{j+1}r+\delta\,(1-r)\,n^{j+1}k} \\
&= \frac{k-\delta\,(1-r)\,k^2}{(1-r)\,2+kr+\delta\,(1-r)\,k^2} \\
&\geq \frac{k-\delta k^2}{2+(k-2)r+\delta k^2} \\
&= \frac{k}{2+(k-2)r}-\varepsilon\,(k,\delta)
\end{aligned}
$$

We obtain the claimed inapproximability factor and the proof follows. ■

6.7 Bibliographic Notes

Some parts of the material presented in this chapter are based on the paper [KSV11]. In particular, the proofs of Theorem 6.5.2, 6.5.3, 6.5.4 and 6.5.5 appeared in [KSV11].

Part III

Sparse Instance Methods and Paradigms

CHAPTER 7

The (1,2)-Steiner Tree Problem

In this chapter, we investigate the approximation hardness of a bounded weight optimization problem and use sparse instance methods to derive explicit approximation lower bounds for the underlying problem. In particular, we construct an approximation preserving reduction from a bounded occurrence CSP called the MAX-HYBRID-LIN2 problem (cf. Definition 4.9.1) to the $(1,2)$-STEINER TREE problem. We exploit the very special structure of the constraints of the CSP in order to obtain an improved inapproximability threshold for the $(1,2)$-STEINER TREE problem. In the subsequent chapters, we construct several reductions from a well-suited bounded occurrence CSP to other problems and apply this method successfully.

The best up to now known inapproximability threshold for the $(1,2)$-STEINER TREE problem is 383/382 due to Hauptmann [H07]. We prove that it is **NP**-hard to approximate the problem to within any constant approximation ratio less than 221/220.

7.1 Introduction

Given a connected graph $\mathcal{G}$ with non-negative costs on edges and a subset of terminal vertices $S \subseteq V(\mathcal{G})$, the STEINER TREE problem asks for the minimum cost subgraph of $\mathcal{G}$ spanning S.

The STEINER TREE problem belongs to the fundamental problems in combinatorial optimization and is among the most important problems in network design with great theoretical and practical relevance. It emerges in a number of contexts with applications ranging from optical and wireless communication systems, energy supply, transportation and distribution networks, internet routing and broadcast problems to VLSI design (see, e.g., [HRW92]). Furthermore, it appears either as a subproblem or as a special case of many other combinatorial optimization problems.

In 1972, Karp proved in his seminal work [K75] the **NP**-hardness of the STEINER TREE problem leaving less hope for polynomial time exact algorithms for this problem. Accordingly, we are interested in efficient approximation algorithms for the STEINER TREE problem with good performance guarantees. A sequence of improved approximation algorithms appeared in the literature. Here, we only mention the most important results starting with the approximation algorithm due to Zelikovsky [Z93] with approximation ratio 1.83. Berman and Ramaiyer [BR94] improved the approximation upper bound for this problem to 1.78. Karpinski and Zelikovsky [KZ97a] designed an approximation algorithm with approximation ratio 1.64 by a sophisticated preprocessing method. By iterating the former approach, Hougardy and Prömel [HP99] achieved an approximation ratio of 1.60. In 2000, Robins and Zelikovsky [RZ00] designed an approximation scheme with approximation ratio $(1.55 + \varepsilon)$ for every constant $\varepsilon > 0$. After 10 years without progression, the long series of improved approximation algorithm culminated in the work of Byrka, Grandoni, Rothvoß and Sanità [BGRS10] with the currently best known approximation upper bound of 1.39 using a novel iterative randomized rounding technique.

On the other hand, Bern and Plassmann [BP89] proved that the STEINER TREE problem is **APX**-hard. In 2003, Thimm [T03] claimed erroneously an inapproximability result for the STEINER TREE problem yielding an approximation lower bound of 163/162. The reduction uses the hardness result due to Håstad [H01] as a starting point. Nevertheless, his construction was corrected and improved by M. Chlebík and J. Chlebíková [CC08]. They proved that it is **NP**-hard to approximate the STEINER TREE problem to within any constant approximation ratio less than 96/95.

The Steiner Tree Problem with Weights One and Two

The $(1,2)$-STEINER TREE problem is an important restriction of the STEINER TREE problem, in which we restrict the input graph $\mathcal{G}$ to be the complete graph with edge weights one and two.

In 1989, Bern and Plassmann [BP89] introduced and considered the $(1,2)$-STEINER TREE problem. They were able to achieve an approximation upper bound of 4/3 for this problem. Robins and Zelikovsky [RZ00] developed an approximation scheme for the $(1,2)$-STEINER TREE problem with approximation ratio $(1.28+\varepsilon)$ for every constant $\varepsilon > 0$. After that, Berman, Karpinski and Zelikovsky [BKZ09] designed the currently best known polynomial time approximation algorithm for this problem with approximation ratio 5/4.

On the lower bound side, Berman and Plassmann [BP89] proved that the $(1,2)$-STEINER TREE problem is **APX**-hard by constructing a reduction from the VERTEX COVER problem restricted to graphs with bounded maximum degree. Hauptmann [H07] combined this reduction with the explicit lower bounds for the VERTEX COVER problem restricted to graphs with maximum degree 4 due to Berman and Karpinski [BK99]. It implies that it is **NP**-hard to approximate the $(1,2)$-STEINER TREE problem to within any constant approximation ratio less than 383/382.

In this work, we prove that it is **NP**-hard to approximate the $(1,2)$-STEINER TREE problem to within any constant approximation ratio less than 221/220.

7.2 Outline of this Chapter

This chapter is organized as follows. In Section 7.4, we formulate our main results. In Section 7.5, we give a high-level view of our reduction. In Section 7.6, we define the corresponding instance $(\mathcal{G}_{\mathscr{L}}, S_{\mathscr{L}})$ of the $(1,2)$-STEINER TREE problem given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem. In Section 7.7, we construct a Steiner tree for $S_{\mathscr{L}}$ in $G_{\mathscr{L}}$ according to a given assignment ϕ to the variables in $\mathscr{L}$. In Section 7.8, we define the assignment $\psi_{\mathcal{T}}$ from a Steiner tree $\mathcal{T}$ for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$. In Section 7.9, we give the proof of Theorem 7.4.1.

7.3 Preliminaries

Given a graph $\mathcal{G}$ and a subset $S \subseteq V(\mathcal{G})$, a Steiner tree $\mathcal{T}$ for S in $\mathcal{G}$ is a tree in $\mathcal{G}$ containing all vertices of S. We refer to the elements of S as *terminals* and to the elements of $V \backslash S$ as *Steiner vertices*.

According to Definition 4.1.1, the STEINER TREE problem is defined as follows.

Definition 7.3.1 (STEINER TREE problem)

Instances: *A graph $\mathcal{G}$, a set $S \subseteq V(\mathcal{G})$ of terminals and a cost function $c : E(\mathcal{G}) \to \mathbb{Q}_+$*

Solutions: *A tree $\mathcal{T}$ in $\mathcal{G}$ with $S \subseteq V(\mathcal{T})$*

Task: *Minimize $c(\mathcal{T}) = \sum_{e \in E(\mathcal{T})} c(e)$*

The special case of the STEINER TREE problem, in which we have given a complete graph and a cost function being restricted to weights one and two, is called the (1,2)-STEINER TREE problem. Let us formulate the problem according to Definition 4.1.1.

Definition 7.3.2 ((1,2)-STEINER TREE problem)

Instances: *A graph $\mathcal{G}$ with $E(\mathcal{G}) = \binom{V(\mathcal{G})}{2}$, a set $S \subseteq V(\mathcal{G})$ of terminals and a cost function $c : \binom{V(\mathcal{G})}{2} \to \{1,2\}$*

Solutions *A tree $\mathcal{T}$ in $\mathcal{G}$ with $S \subseteq V(\mathcal{T})$*

Task: *Minimize $c(\mathcal{T}) = \sum_{e \in E(\mathcal{T})} c(e)$*

In order to describe an instance of the (1,2)-STEINER TREE problem, it suffices to specify the edges of weight one. Therefore, we represent an instance $(\mathcal{G}, S, c)$ of the (1,2)-STEINER TREE problem by a graph $\mathcal{G}'$ and the set $S' = S$. Edges of the graph $\mathcal{G}'$ denote edges with weight one in $\binom{V(\mathcal{G})}{2}$, whereas all $e \in \binom{V(\mathcal{G})}{2} \backslash E(\mathcal{G})$ are edges with weight two. Moreover, the cost function $c : \binom{V(\mathcal{G})}{2} \to \{1,2\}$ of the instance $(\mathcal{G}, S, c)$ can be reconstructed as follows.

$$c(e) = \begin{cases} 1 & \text{if } e \in E(\mathcal{G}') \\ 2 & \text{otherwise} \end{cases}$$

In the following, we refer to edges with weight one as *black* edges, whereas edges with weight two will be called *red* edges.

7.4 Our Contribution

In the subsequent sections, we construct an approximation preserving reduction from the MAX-HYBRID-LIN2 problem (see Definition 4.9.1) to the

$(1,2)$-STEINER TREE problem. In particular, we prove the following statement.

Theorem 7.4.1
Given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 *problem with n variables, m_2 equations with two variables and m_3 equations with three variables as given in Theorem 4.9.1, we construct in polynomial time an instance $(\mathcal{G}_{\mathscr{L}}, S_{\mathscr{L}})$ of the* $(1,2)$-STEINER TREE *problem with the following properties.*

- *(i) If there exists an assignment to the variables of $\mathscr{L}$, which leaves at most u equations unsatisfied, then, there exist a Steiner tree $\mathcal{T}$ for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$ with $c(\mathcal{T}) = 2 \cdot n + 2 \cdot m_2 + 8 \cdot m_3 + u$.*
- *(ii) From every Steiner tree $\mathcal{T}$ for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$ with $c(\mathcal{T}) = 2{\cdot}n+2{\cdot}m_2+8{\cdot}m_3+u$, we can construct in polynomial time an assignment $\psi_{\mathcal{T}}$ to the variables of $\mathscr{L}$ that leaves at most u equations in $\mathscr{L}$ unsatisfied.*

The former theorem can be used to derive an explicit approximation lower bound for the $(1,2)$-STEINER TREE problem by reducing instances of the MAX-HYBRID-LIN2 problem of the form described in Theorem 4.9.1 to the $(1,2)$-STEINER TREE problem.

Corollary 7.4.1
It is **NP**-*hard to approximate the* $(1,2)$-STEINER TREE *problem to within any constant approximation ratio less than* $221/220$.

Proof of Corollary 7.4.1.
For a fixed constant $\varepsilon > 0$, we choose $\delta \in (0, 1/2)$ such that

$$\frac{221-\delta}{220+\delta} \geq \frac{221}{220} - \varepsilon$$

holds. Given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem with 42ν variables, 60ν equations with two variables and 2ν equations with exactly three variables, we construct the corresponding instance $(\mathcal{G}_{\mathscr{L}}, S_{\mathscr{L}})$ of the $(1,2)$-STEINER TREE problem with the properties described in Theorem 7.4.1. Then, we conclude according to Theorem 4.9.1 that there exist a

Steiner tree for $S_{\mathcal{H}}$ in $\mathcal{G}_{\mathcal{H}}$ with cost at most

$$2 \cdot 42\nu + 2 \cdot 60\nu + 8 \cdot 2\nu + \delta\nu \;=\; (220 + \delta)\nu$$

or the cost of a Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$ is bounded from below by

$$2 \cdot 42\nu + 2 \cdot 60\nu + 8 \cdot 2\nu + (1 - \delta)\nu \;=\; (221 - \delta)\nu.$$

By Theorem 4.9.1, we know that the two cases above are **NP**-hard to distinguish. Hence, for every $\varepsilon > 0$, it is **NP**-hard to find a solution to the (1, 2)-STEINER TREE problem to within any constant approximation ratio better less than

$$\frac{221 - \delta}{220 + \delta} \geq \frac{221}{220} - \epsilon$$

and the proof of Corollary 7.4.1 follows. ■

7.5 The High-Level View of the Reduction

Given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem, we will construct a corresponding graph $\mathcal{G}_{\mathscr{L}}$ and define the associated terminal set $S_{\mathscr{L}}$.

For every variable x_i occurring in an equation in $\mathscr{L}$, $\mathcal{G}_{\mathscr{L}}$ contains two Steiner vertices representing the two possible values of x_i. We refer to them as the *variable vertices* of x_i. The instance $(\mathcal{G}_{\mathscr{L}}, S_{\mathscr{L}})$ has the property that it is advantageous to include exactly one of the variable vertices corresponding to a variable. Moreover, every Steiner tree $\mathcal{T}$ with $c(\mathcal{T})$ can be transformed into a normed solution $\mathcal{T}'$ such that $c(\mathcal{T}') \leq c(\mathcal{T})$ and $\mathcal{T}'$ includes exactly one variable vertex for every variable.

For every equation ℓ in $\mathscr{L}$, we define an associated subgraph $\mathcal{G}(\ell)$ simulating the equation ℓ given an assignment to the variables in ℓ. The subgraph $\mathcal{G}(\ell)$ is connected to a variable vertex if the corresponding variable occurs in ℓ. Furthermore, we show that if we include a wrong combination of variable vertices corresponding to a non-satisfying assignment of the equation ℓ, we have to use at least one red edge, whereas a satisfying combination entails the possibility of using only black edges. Therefore, a non-satisfying combination will be punished by paying a unit more.

7.6 Constructing $(\mathcal{G}_{\mathscr{L}}, S_{\mathscr{L}})$ from $\mathscr{L}$

Given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem with variables $\{x_1, \ldots, x_n\}$, m_2 equations ℓ_i^2 with two variables and m_3 equations ℓ_j^3 with exactly three variables, we now describe the construction of the corresponding instance $(\mathcal{G}_{\mathscr{L}}, S_{\mathscr{L}})$ of the $(1,2)$-STEINER TREE problem.

Variable Graph

For every variable $x_i \in \{x_1, \ldots, x_n\}$, we introduce two vertices, v_i^1 and v_i^0. They correspond to the possible assignments to x_i. In the remainder, we refer to them as the *variable vertices.* In addition, we introduce the terminal t_i, which we call the *variable terminal* of x_i. Both variable vertices are connected to its variable terminal.

Next, for every equation ℓ in $\mathscr{L}$, we are going to define the corresponding subgraph $\mathcal{G}(\ell)$ in $\mathcal{G}_{\mathscr{L}}$. We start with equations with two variables.

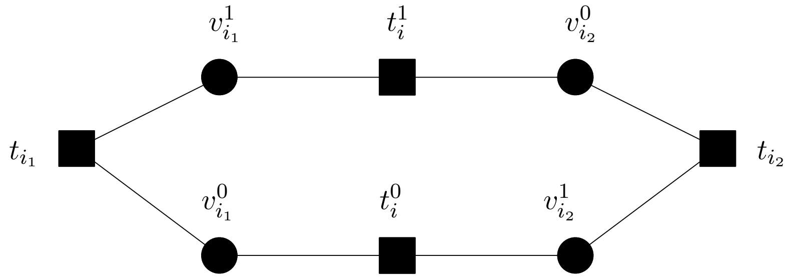


Figure 7.1: The subgraph corresponding to $\ell_i^2 \equiv x_{i_1} \oplus x_{i_2} = 0$ connected to variable terminals t_{i_1} and t_{i_2}.

Subgraph for Equations with Two Variables

For every equation of the form $\ell_i^2 \equiv x_{i_1} \oplus x_{i_2} = 0$ with $i \in [m_2]$ and $i_1 < i_2$, we define the subgraph $\mathcal{G}(\ell_i^2)$ consisting of the vertices $v_{i_1}^1$, $v_{i_1}^0$, $v_{i_2}^1$, $v_{i_2}^0$, t_i^1 and t_i^0. Both, t_i^1 and t_i^0 are terminals. We refer to t_i^1 and t_i^0 as the *equation terminals* of $\mathcal{G}(\ell_i^2)$. The terminal t_i^1 is connected to $v_{i_1}^1$ and $v_{i_2}^0$. On the other hand, $\mathcal{G}(\ell_i^2)$ contains the edges $\{t_i^0, v_{i_1}^0\}$ and $\{t_i^0, v_{i_1}^1\}$. The graph is displayed in Figure 7.1.

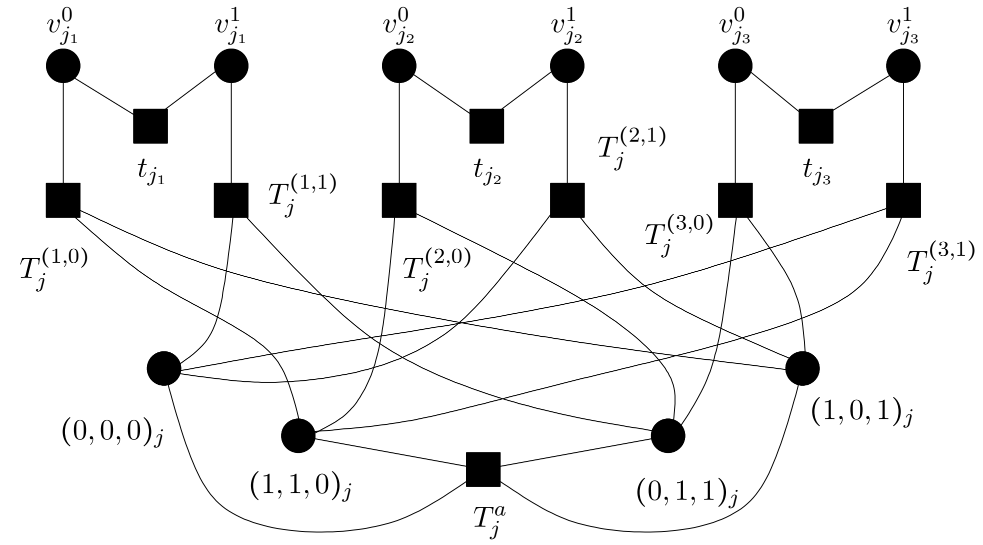


Figure 7.2: Subgraph for the equation $\ell_j^3 \equiv x_{j_1} \oplus x_{j_2} \oplus x_{j_3} = 0$ with the corresponding variable terminals.

As for the next step, we define the graph $\mathcal{G}(\ell_j^3)$ corresponding to the equation with three variables ℓ_j^3 in $\mathscr{L}$.

Subgraph for Equations with Three Variables

Let us start with equations of the form $x_{j_1} \oplus x_{j_2} \oplus x_{j_3} = 0$. Then, the subgraph $\mathcal{G}(\ell_j^3)$ contains the Steiner vertices $(0,0,0)_j$, $(1,0,1)_j$, $(1,1,0)_j$ and $(0,1,1)_j$. We refer to these vertices as the *special vertices* of $\mathcal{G}(\ell_j^3)$. In addition, we introduce the terminals $T_j^{(1,0)}$, $T_j^{(1,1)}$, $T_j^{(2,0)}$, $T_j^{(2,1)}$, $T_j^{(3,0)}$, $T_j^{(3,1)}$ and T_j^b. The terminal T_j^b is connected to all special vertices of $\mathcal{G}(\ell_j^3)$, whereas the special vertex $(i_1, i_2, i_3)_j$ is connected to $T_j^{(l,1-i_l)}$ for all $l \in \{1,2,3\}$. Finally, all special vertices are joined by an edge with the main terminal. We call $T_j^{(1,0)}$, $T_j^{(1,1)}$, $T_j^{(2,0)}$, $T_j^{(2,1)}$, $T_j^{(3,0)}$ and $T_j^{(3,1)}$ the *equation terminals* of $\mathcal{G}(\ell_j^3)$. On the other hand, we refer to T_j^b as the *base terminal* of $\mathcal{G}(\ell_j^3)$. The graph $\mathcal{G}(\ell_j^3)$ is displayed in Figure 7.2. For equations ℓ_j^3 of the form $x_{j_1} \oplus x_{j_2} \oplus x_{j_3} = 1$, we introduce a similar graph $\mathcal{G}(\ell_j^3)$. The special vertices are given by $(1,1,1)_j$, $(0,0,1)_j$, $(1,0,0)_j$ and $(0,1,0)_j$, whereas the equation terminals are defined as $T_j^{(1,0)}$, $T_j^{(1,1)}$, $T_j^{(2,0)}$, $T_j^{(2,1)}$, $T_j^{(3,0)}$ and $T_j^{(3,1)}$. As before, the special vertices of $\mathcal{G}(\ell_j^3)$ are all connected to the base terminal T_j^b. Finally, we join the special vertex $(i_1,i_2,i_3)_j$ with $T_j^{(l,1-i_l)}$ by an edge for all $l \in \{1,2,3\}$. The graph $\mathcal{G}(\ell_j^3)$ is displayed in Figure 7.3.

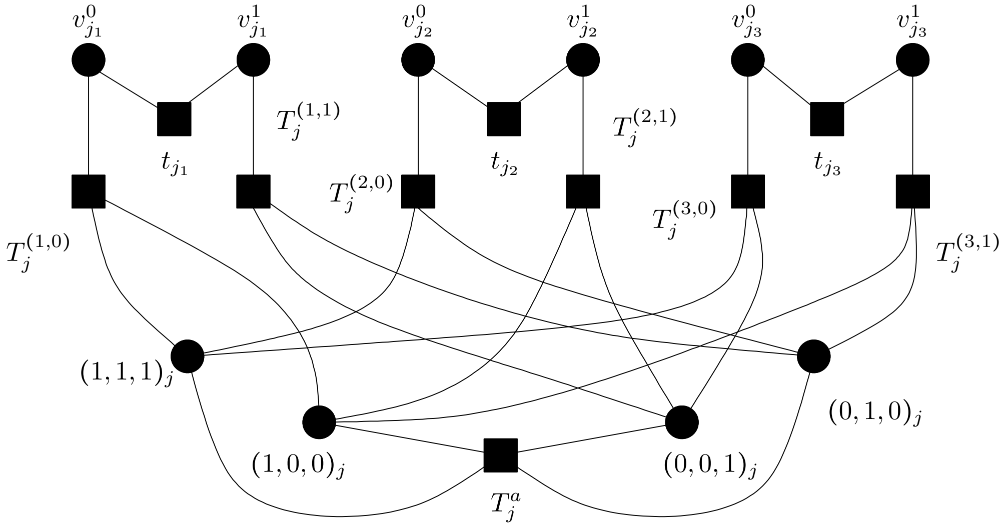


Figure 7.3: Subgraph for $\ell_j^3 \equiv x_{j_1} \oplus x_{j_2} \oplus x_{j_3} = 1$ with the corresponding variable terminals.

Finally, we introduce the terminal T_m which is called the *main terminal*. Every introduced Steiner vertex of $\mathcal{G}_{\mathscr{L}}$ will be connected to the terminal T_m.

7.7 Constructing the Steiner tree $\mathcal{T}_\phi$ from ϕ

Given an assignment ϕ to the variables of $\mathscr{L}$, we are going to construct a Steiner tree $\mathcal{T}_\phi$ for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$. Let us begin with the description of $\mathcal{T}_\phi$.

Variable Vertices

For all $i \in [n]$, we connect the Steiner vertex $v_i^{\phi(x_i)}$ to the main terminal T_m. Furthermore, every variable vertex $v_i^{\phi(x_i)}$ is joined by an edge with its variable terminal t_i for all $i \in [n]$.

As for the next step, we are going to connect equation terminals in graphs corresponding to equations with two variables.

Equations with Two variables

Let us consider the equation $\ell_k^2 \equiv x_i \oplus x_j = 0$ with $i < j$. We have to distinguish two cases.

Case ($\phi(x_i) = \phi(x_j)$):
If $\phi(x_i) = \phi(x_j)$ holds, it is possible to add the edges $\{v_i^{\phi(x_i)}, t_k^{\phi(x_i)}\}$ and $\{v_j^{\phi(x_j)}, t_k^{1-\phi(x_j)}\}$ to the actual tree. Notice that both edges are black.

Case ($\phi(x_i) \neq \phi(x_j)$):
In this case, we use the edge $\{v_i^{\phi(x_i)}, t_k^{\phi(x_i)}\}$ and join T_m with $t_k^{1-\phi(x_i)}$ by means of a red edge.

Next, we consider equations with exactly three variables.

Equations with Three Variables

Given an equation of the form $\ell_j^3 \equiv x_{i_1} \oplus x_{i_2} \oplus x_{i_3} = b_j$ with $b_j \in \{0,1\}$, we connect the vertex $v_{i_l}^{\phi(x_{i_l})}$ with the terminal $T_j^{(l,\phi(x_{i_l}))}$ for all $l \in \{1,2,3\}$ in $\mathcal{G}(\ell_j^3)$. Analogously, we will distinguish two cases.

Case (ϕ satisfies the equation ℓ_j^3):
In this case, we join the special vertex $\big(\phi(x_{i_1}), \phi(x_{i_2}), \phi(x_{i_3})\big)_j$ with the main terminal and T_j^b by a black edge. Then, we connect $\big(\phi(x_{i_1}), \phi(x_{i_2}), \phi(x_{i_3})\big)_j$ with terminals $T_j^{(1,1-\phi(x_{i_1}))}$, $T_j^{(2,1-\phi(x_{i_2}))}$ and $T_j^{(3,1-\phi(x_{i_3}))}$. Notice that we used only black edges due to the construction of $\mathcal{G}(\ell_j^3)$.

Case (ϕ leaves ℓ_j^3 unsatisfied):
Then, we join $\big(\phi(x_{i_1}), 1-\phi(x_{i_2}), \phi(x_{i_3})\big)_j$ with T_j^b and T_m by black edges. In addition, we connect $\big(\phi(x_{i_1}), 1-\phi(x_{i_2}), \phi(x_{i_3})\big)_j$ with terminals $T_j^{(1,1-\phi(x_{i_1}))}$ and $T_j^{(3,1-\phi(x_{i_3}))}$. The remaining terminal $T_j^{(2,1-\phi(x_{i_2}))}$ is joined with T_j^b by a

red edge.

In summary, we note that we have to use at most one red edge in order to include all terminals if the underlying equation is not satisfied. Accordingly, we have to spend two black edges for each variable in $\mathscr{L}$, two black edges for each satisfied equation with two variables and eight black edges for each satisfied equation with three variables. In addition, we have to pay one unit more for every equation that is not satisfied by ϕ.

We obtain the following statement.

Lemma 7.7.1
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem with n variables, m_2 equations with two variables, m_3 equations with three variables and ϕ an assignment to the variables of $\mathscr{L}$ leaving u equations in $\mathscr{L}$ unsatisfied. Then, $\mathcal{T}_\phi$ is a Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$ with cost $c(\mathcal{T}_\phi) \leq 2n + 2m_2 + 8m_3 + u$.*

7.8 Defining the Assignment

Given a Steiner tree $\mathcal{T}$ for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$, we are going to construct an assignment $\psi_{\mathcal{T}}$ according to $\mathcal{T}$. In order to assign a value to a variable x_i in $\mathscr{L}$, we have to establish a criterion that tells us which value we have to assign to x_i. The associated mapping $\psi_{\mathcal{T}}$ assigns the value $b \in \{0, 1\}$ to x_i if the corresponding variable vertex v_i^b is included in $\mathcal{T}$. Therefore, we have to transform the underlying solution $\mathcal{T}$ into a Steiner tree $\widehat{\mathcal{T}}$ for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$ containing exactly one variable vertex per variable without increasing the cost. In addition, this *normed* Steiner tree has some other nice properties in order to achieve a stronger basis for relating the number of satisfied equations in $\mathscr{L}$ by $\psi_{\widehat{\mathcal{T}}}$ to the cost of $\widehat{\mathcal{T}}$.

The Normed Steiner Tree

In order to specify a well-defined assignment, we have to attain a normed solution. We now give the definition of a normed Steiner tree.

Definition 7.8.1 (Normed Steiner tree)
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem, $(\mathcal{G}_{\mathscr{L}}, S_{\mathscr{L}})$ its corresponding instance of the* $(1,2)$-STEINER TREE *problem and $\mathcal{T}$ a Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$. We refer to $\mathcal{T}$ as a normed Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$ if the following conditions hold.*

- *For all variables x_i in $\mathscr{L}$, we have that $|\{v_i^1, v_i^0\} \cap V(\mathcal{T})| = 1$.*
- *For all variables x_i in $\mathscr{L}$, we have $\left|\{\{v_i^1, T_m\}, \{v_i^0, T_m\}\} \cap E(\mathcal{T})\right| = 1$.*
- *For all equations with three variables ℓ_j^3 in $\mathscr{L}$, there is at least one special vertex of $\mathcal{G}(\ell_j^3)$ contained in $V(\mathcal{T})$.*

Given a normed Steiner tree $\mathcal{T}$ for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$, we define an assignment to the variables in $\mathscr{L}$ according to which variable vertices are included in $V(\mathcal{T})$. We introduce the assignment $\psi_{\mathcal{T}}$, which assigns the value $b \in \{0,1\}$ to the variable x_i if v_i^b is included in $V(\mathcal{T})$. Note that due to the definition of a normed Steiner tree, $\psi_{\mathcal{T}}$ is indeed well-defined. We now give the definition of $\psi_{\mathcal{T}}$.

Definition 7.8.2 (Assignment $\psi_{\mathcal{T}}$)
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem with variables $\{x_1, \dots, x_n\}$, $(\mathcal{G}_{\mathscr{L}}, S_{\mathscr{L}})$ its corresponding instance of the* $(1,2)$-STEINER TREE *problem and $\mathcal{T}$ a normed Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$. Given $\mathcal{T}$, we define the associated assignment $\psi_{\mathcal{T}} : \{x_1, \dots, x_n\} \to \{0,1\}$ as follows.*

$$\psi_{\mathcal{T}}(x_i) = \begin{cases} 1 & \textit{if } v_i^1 \in V(\mathcal{T}) \\ 0 & \textit{otherwise} \end{cases}$$

As for the next step, we are going to analyze the relation between the cost of a normed Steiner tree $\mathcal{T}$ and the number of satisfied equations by $\psi_{\mathcal{T}}$.

Local Costs in Subgraphs $G(\ell)$

Given a normed Steiner tree $\mathcal{T}$ for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$, we are interested how the cost of $\mathcal{T}$ change depending on which variable vertices are included in $\mathcal{T}$. For this reason, we define the *assignment star* $\mathcal{S}_{\mathcal{T}}$ of a normed Steiner tree

$\mathcal{T}$. $\mathcal{S}_{\mathcal{T}}$ consists of the main terminal of $\mathcal{G}_{\mathscr{L}}$ which is joined by an edge with every variable vertex included in $V(\mathcal{T})$. In addition, those variable vertices are connected to their corresponding variable terminals. We now give the definition of the assignment star.

Definition 7.8.3 (Assignment Star $\mathcal{S}_{\mathcal{T}}$)
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem with variables $\{x_1, \dots, x_n\}$, $(\mathcal{G}_{\mathscr{L}}, S_{\mathscr{L}})$ its corresponding instance of the* $(1,2)$-STEINER TREE *problem and $\mathcal{T}$ a normed Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$. The assignment star $\mathcal{S}_{\mathcal{T}}$ of $\mathcal{T}$ is defined by $V(\mathcal{S}_{\mathcal{T}}) = \{\, T_m, v_i^b, t_i \mid v_i^b \in V(\mathcal{T}),\ i \in [n]\}$ and $E(\mathcal{S}_{\mathcal{T}}) = \big\{\{T_m, v_i^b\}, \{v_i^b, t_i\} \bigm| v_i^b \in V(\mathcal{T}),\ i \in [n]\big\}$.*

Next, we are going to analyze how the cost of a normed Steiner $\mathcal{T}$ tree change and in particular, the local cost of a subgraph $\mathcal{G}(\ell)$ with ℓ in $\mathscr{L}$ according to whether $\psi_{\mathcal{T}}$ satisfies ℓ or not. More precisely, we want to attain a lower bound on the cost that are associated with the subgraph $\mathcal{G}(\ell)$ given a normed Steiner tree $\mathcal{T}$ and furthermore, how the cost increase when $\psi_{\mathcal{T}}$ leaves the corresponding equation ℓ unsatisfied. For this reason, we introduce the notion of local costs.

Definition 7.8.4 (Local Costs for $\mathcal{S}_{\mathcal{T}}$ in $\mathcal{G}(\ell)$)
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem, $(\mathcal{G}_{\mathscr{L}}, S_{\mathscr{L}})$ its corresponding instance of the* $(1,2)$-STEINER TREE *problem and $\mathcal{T}$ a normed Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$. Given an equation ℓ in $\mathscr{L}$ and the assignment star $\mathcal{S}_{\mathcal{T}}$ of $\mathcal{T}$, we define the local costs for $\mathcal{S}_{\mathcal{T}}$ in $\mathcal{G}(\ell)$ as the minimum cost of edges that must be added to $\mathcal{S}_{\mathcal{T}}$ in order to connect all equation terminals of $\mathcal{G}(\ell)$.*

Given a normed Steiner tree $\mathcal{T}$ for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$ and its assignment star $\mathcal{S}_{\mathcal{T}}$, we are going to analyze the local cost in subgraphs $\mathcal{G}(\ell)$ for every ℓ in $\mathscr{L}$. More precisely, we are going to prove the following lemma.

Lemma 7.8.1
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem, $(\mathcal{G}_{\mathscr{L}}, S_{\mathscr{L}})$ its corresponding instance of the* $(1,2)$-STEINER TREE *problem and $\mathcal{T}$ a normed*

Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$. Given an equation with two variables ℓ^2, the local costs for $\mathcal{S}_{\mathcal{T}}$ in $\mathcal{G}(\ell^2)$ are 2 *units if $\psi_{\mathcal{T}}$ satisfies ℓ^2, and* 3 *otherwise. Given an equation with three variables ℓ^3, the local costs for $\mathcal{S}_{\mathcal{T}}$ in $\mathcal{G}(\ell^3)$ are* 8 *units if $\psi_{\mathcal{T}}$ satisfies ℓ^3 and* 9 *otherwise.*

Proof of Lemma 7.8.1.
By differentiating between equations with two and three variables, we now analyze the local costs in the corresponding subgraph. Let us start with equations of the form $\ell_i^2 \equiv x_{i_1} \oplus x_{i_2} = 0$.

Case (equations with two variables):
Let us consider the equation $\ell_i^2 \equiv x_{i_1} \oplus x_{i_2} = 0$. Let $\mathcal{T}$ be a normed Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$ and $\mathcal{S}_{\mathcal{T}}$ its assignment star. Since both terminals, t_i^0 and t_i^1, must be included in $\mathcal{T}$, we attain a lower bound on the local costs in $\mathcal{G}(\ell_i^2)$ of 2 units for including two black edges. Clearly, the only two possibilities to connect both terminals with black edges are firstly, $v_{i_1}^0$ and $v_{i_2}^0$ are included in $V(\mathcal{T})$ and secondly, $v_{i_1}^1$ and $v_{i_2}^1$ are contained in $V(\mathcal{T})$. Otherwise, we have to use at least one red edge, which means that we have to spend an unit more in order to connect all terminals.

Case (equations with three variables):
Given an equation of the form $\ell_j^3 \equiv x_{j_1} \oplus x_{j_2} \oplus x_{j_3} = 0$, we have to connect all seven terminals of $\mathcal{G}(\ell_j^3)$. Hence, we obtain a lower bound on the local cost for $\mathcal{S}(\mathcal{T})$ in $\mathcal{G}(\ell_j^3)$ of 8 units.

If the variable vertices $v_{j_1}^{i_1}$, $v_{j_2}^{i_2}$ and $v_{j_3}^{i_3}$ possess one of the following combinations $(i_1, i_2, i_3) \in \big\{(0,0,0), (1,1,0), (0,1,1), (1,0,1)\big\}$, we may use black edges to connect the terminals $T_j^{(1,i_1)}$,$T_j^{(2,i_2)}$ and $T_j^{(3,i_3)}$. Furthermore, we join the main terminal with the special vertex $(i_1, i_2, i_3)_j$ by a black edge. By including $(i_1, i_2, i_3)_j$ in our solution, it entails the possibility of connecting all remaining three terminals by black edges. The former combinations of vertices induce local costs for $\mathcal{S}(\mathcal{T})$ in $\mathcal{G}(\ell_j^3)$ of 8 units.

If, by contrast, the variable vertices $v_{j_1}^{i_1}$, $v_{j_2}^{i_2}$ and $v_{j_3}^{i_3}$ admit one of the combinations $(i_1, i_2, i_3) \in \{(1,0,0), (1,1,1), (0,1,0), (0,0,1)\}$, there are only special vertices that are connected to at most two remaining terminals by black edges. Hence, we obtain local cost for $\mathcal{S}(\mathcal{T})$ in $\mathcal{G}(\ell_j^3)$ of 9 units.

Finally, we conclude that a similar situation holds when the underlying equation is of the form $x_{j_1} \oplus x_{j_2} \oplus x_{j_3} = 1$ and the proof follows. ∎

Local Transformation

We now define a series of local transformations which enables us to convert a given Steiner tree $\mathcal{T}$ for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$ successively in a normed Steiner tree $\mathcal{T}'$ with $c(\mathcal{T}') \leq c(\mathcal{T})$. By each of the local transformations, the underlying solution $\mathcal{T}$ gains another property without increasing the cost of it. Our first transformation is concerning Steiner trees $\mathcal{T}$ for $S_{\mathscr{L}}$, in which both variable vertices corresponding to a variable in $\mathscr{L}$ are not included in $V(\mathcal{T})$.

Transformation ❶
Let x_i be a variable in $\mathscr{L}$ with $|\{v_i^1, v_i^0\} \cap V(\mathcal{T})| = 0$. Let $\mathcal{C}$ be the connected component of $\mathcal{T}$ that arises by removing the variable terminal t_i from $V(\mathcal{T})$ with $T_m \in V(\mathcal{C})$. Notice that the edge $r \in E(\mathcal{T})$ connecting t_i with $\mathcal{C}$ must be red. We remove r from $E(\mathcal{T})$ and add two black edges to $E(\mathcal{T})$ instead. More precisely, we use the black edges $\{t_i, v_i^0\}$ and $\{T_m, v_i^0\}$. Every path, that used the edge r, can now lead over $t_i - v_i^0 - T_m$. Hence, the solution is still connected. Since we added a vertex and an edge, the obtained solution remains a tree. Notice that the cost of the solution has not been increased during the process.∎

So far, the actual solution $\mathcal{T}$ is a Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$ including at least one variable vertex for every variable in $\mathscr{L}$. But still, there is possibly a variable vertex in $V(\mathcal{T})$, which is not connected to the main terminal. The next transformation resolves this situation.

Transformation ❷
Let v_i^a with $a \in \{0, 1\}$ be a variable vertex included in $V(\mathcal{T})$. Furthermore, we assume that $\{v_i^a, T_m\}$ is not contained in $E(\mathcal{T})$. We add the black edge

$\{v_i^a, T_m\}$ to $E(\mathcal{T})$. As a consequence, we obtain a simple cycle $\mathcal{C}_s$ in $\mathcal{T}$, which contains the edge $\{T_m, v_i^a\}$. Due to the construction of $\mathcal{G}_{\mathscr{L}}$, we can delete an arbitrary edge from $\mathcal{C}_s$, except black edges joining a variable vertex with T_m. Since we added and then, deleted an edge of a simple cycle, $\mathcal{T}$ remains connected. Finally, we note that $\mathcal{T}$ maintains a Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$ and the cost of the solution has not been increased during the process.■

Our actual solution $\mathcal{T}$ contains for every variable x_i in $\mathscr{L}$ at least one variable vertex and all contained variable vertices are connected to T_m by a black edge. However, our goal is to obtain a normed Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$, which contains exactly one variable vertex for every variable. The following transformation removes one of the two possibly existing variable vertices. In what follows, we will differentiate between variables according to their occurrences in equations. Recall from Theorem 4.9.1 that a variable of an instance of the MAX-HYBRID-LIN2 problem occurs exactly in three equations, thereof at least twice in equations with two variables. The last occurrence is either in an equation with exactly three variables or in another equation with two variables. The following transformation deals with the latter case.

Transformation ❸

Let $\ell_{i_1}^2 \equiv x_{j_1} \oplus x_i = 0$, $\ell_{i_2}^2 \equiv x_i \oplus x_{j_2} = 0$ and $\ell_{i_3}^2 \equiv x_i \oplus x_{j_3} = 0$ be the three equations in which the variable x_i occur. Furthermore, we assume that v_i^0 and v_i^1 are both included in $\mathcal{T}$ as well as the edges $\{v_i^0, T_m\}$ and $\{v_i^1, T_m\}$. In the following, we define some operations, which we will apply exemplary to the scenario displayed in Figure 7.4. We now define the rule that determines the particular variable vertex to be removed from $V(\mathcal{T})$. For this reason, we introduce a notion that will be very useful in this concept.

In the remainder, we refer to an equation terminal t as *switchable* in $\mathcal{T}$ if there exists two Steiner vertices $v, w \in V(\mathcal{T})$ with $\{t, v\} \in E(\mathcal{G}_{\mathscr{L}})$ and $\{t, w\} \in E(\mathcal{G}_{\mathscr{L}})$. In the scenario displayed in Figure 7.4, the terminals $t_{i_1}^1$, $t_{i_3}^0$ and $t_{i_2}^1$ are switchable. In addition, we refer to equation terminals, which are not switchable in $\mathcal{T}$, simply as *unswitchable*.

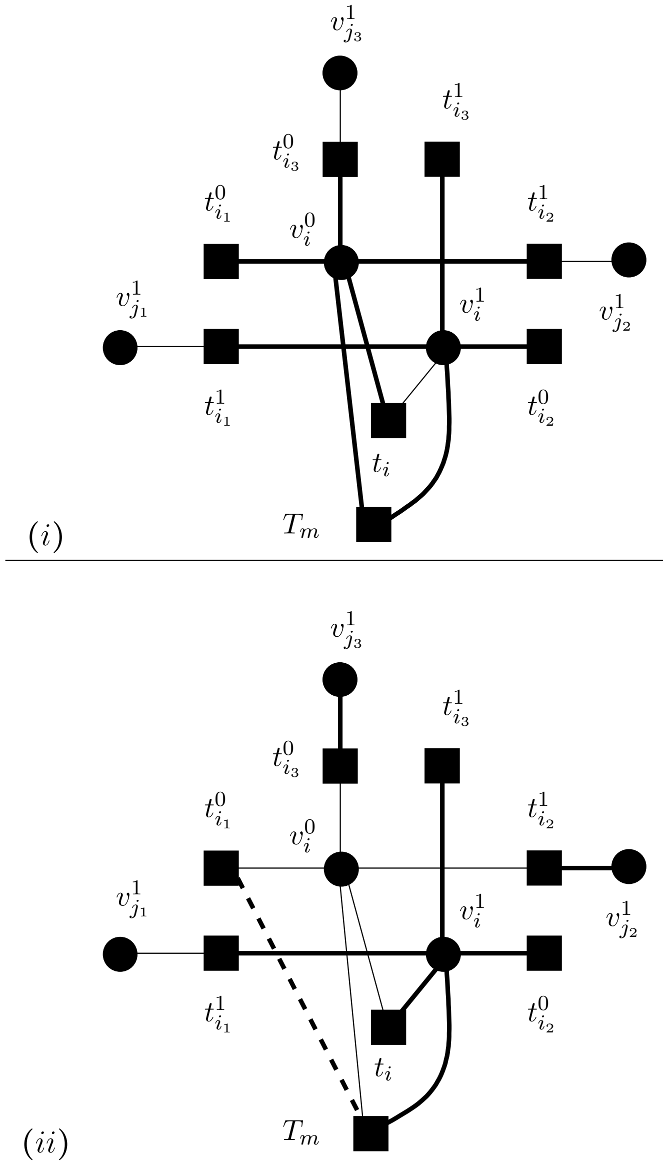


Figure 7.4: Situations before (i) and after (ii) Transformation ❸. The straight black lines are black edges of $\mathcal{G}_{\mathscr{L}}$, whereas thick lines represent black edges in $E(\mathcal{T})$. Finally, red edges included in $E(\mathcal{T})$ are indicated by dashed lines.

We remove the variable vertex from $\mathcal{T}$ which is connected to at most one unswitchable terminal. If both variable vertices possess this property, we remove v_i^0 from $V(\mathcal{T})$.

Notice that the existence of a variable vertex with at most one unswitchable terminal is guaranteed due to the crucial property of having at least one variable vertex per variable included in $\mathcal{T}$ and due to the fact that a variable occurs in three equations.
Let v_i^0 be the variable vertex which we want to remove. In the next step, we consider switchable equation terminals.

Uncoupling switchable terminals:
Let t be an equation terminal, which is connected to v_i^0 by an black edge in $\mathcal{T}$. In addition, let t be joined with $v_{j_2}^1$ by an edge in $E(\mathcal{G}_{\mathscr{L}})\backslash E(\mathcal{T})$. We remove $\{v_i^0, t\}$ from $E(\mathcal{T})$ and insert the black edge $\{v_{j_2}^1, t\}$ instead. The situation is displayed in Figure 7.4. The obtained solution $\mathcal{T}'$ is still connected, since all paths that contained the edge $\{v_i^0, t\}$ can lead over $v_i^0 - T_m - v_{j_2}^1 - t$. Notice that we have not changed the number of edges. Hence, $\mathcal{T}'$ is a Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$ with the same cost as before.

At this point, we are ready to remove the redundant variable vertex and take care of the possibly existing unswitchable terminal.

Removing redundant variable vertices:
First of all, we remove all red edges $r \in E(\mathcal{T})$ with endpoint v_i^0 from $E(\mathcal{T})$ and insert red edges r' with new endpoint T_m instead. If v_i^0 is connected to t_i, we delete $\{v_i^0, t_i\}$ and add $\{v_i^1, t_i\}$ to $E(\mathcal{T})$. Clearly, those conversions do not affect the connectivity of our solution. By now, v_i^0 is connected only to T_m and the remaining unswitchable terminal t. We delete both black edges, $\{t, v_i^0\}$ and $\{T_m, v_i^0\}$. Finally, we add the red edge $\{t, T_m\}$ to $E(\mathcal{T})$. We note that the solution remains connected. Since we deleted a vertex and an edge, we obtain a Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$. All in all, the cost has not been increased during the whole transformation.■

At this point, we are close to our goal. It remains to remove redundant variable vertices whose corresponding variables occur in equations with exactly three variables. Furthermore, in order to obtain a normed Steiner tree, $\mathcal{T}$ is supposed to contain at least one special vertex for every subgraph $\mathcal{G}(\ell_j^3)$ with ℓ_j^3 in $\mathscr{L}$. The last property will be achieved by means of the following transformation.

Transformation ❹
Let $\mathcal{G}(\ell_j^3)$ be a subgraph for an equation with exactly three variables ℓ_j^3 in $\mathscr{L}$. Furthermore, we assume that there is no special vertex of $\mathcal{G}(\ell_j^3)$ included in $V(\mathcal{T})$. Let $\mathcal{C}$ be the connected component which results from deleting T_j^b from $\mathcal{T}$ with $T_m \in V(\mathcal{C})$. Notice that due to our assumption, the edge r, connecting T_j^b with $\mathcal{C}$ in $\mathcal{T}$, must be red. We remove r from $E(\mathcal{T})$ and add an arbitrary special vertex s_j of $\mathcal{G}(\ell_j^3)$ to $V(\mathcal{T})$. Then, we connect s_j with T_j^b and T_m by black edges. Clearly, $\mathcal{T}'$ is connected. Since we added a vertex and an edge, we obtain a Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$ with the same cost.■

By now, the underlying solution contains at least one special vertex in every subgraph $\mathcal{G}(\ell_j^3)$ with $j \in [m_3]$. In the last transformation, we remove redundant variable vertices corresponding to variables which occur in equations with exactly three variables.

Transformation ❺
Let x_i be a variable in $\mathscr{L}$ whose corresponding variable vertices are both included in $V(\mathcal{T})$. Furthermore, let $\ell_j^3 \equiv x_i \oplus x_{j_2} \oplus x_{j_3} = 0$, $\ell_k^2 \equiv x_i \oplus x_{k_1} = 0$ and $\ell_l^2 \equiv x_i \oplus x_{l_1} = 0$ be the equations, in which x_i occur. Both, $\{T_j^{(1,1)}, v_i^1\}$ and $\{T_j^{(1,0)}, v_i^0\}$ are black edges in $E(\mathcal{G}_{\mathscr{L}})$. As before, we want to remove one of the variable vertices and choose the variable vertex which is connected to at most one unswitchable terminal. Due to the former transformations and the construction of $\mathcal{G}(\ell_j^3)$, we may assume that one of the terminals in $\{T_j^{(1,1)}, T_j^{(1,0)}\}$ is connected to the special vertex s_j by an black edge in $\mathcal{G}_{\mathscr{L}}$. Hence, the existence of a variable vertex with at most one unswitchable terminal is guaranteed. The scenario is displayed in Figure 7.5.

In the following, we are going to uncouple switchable terminals.

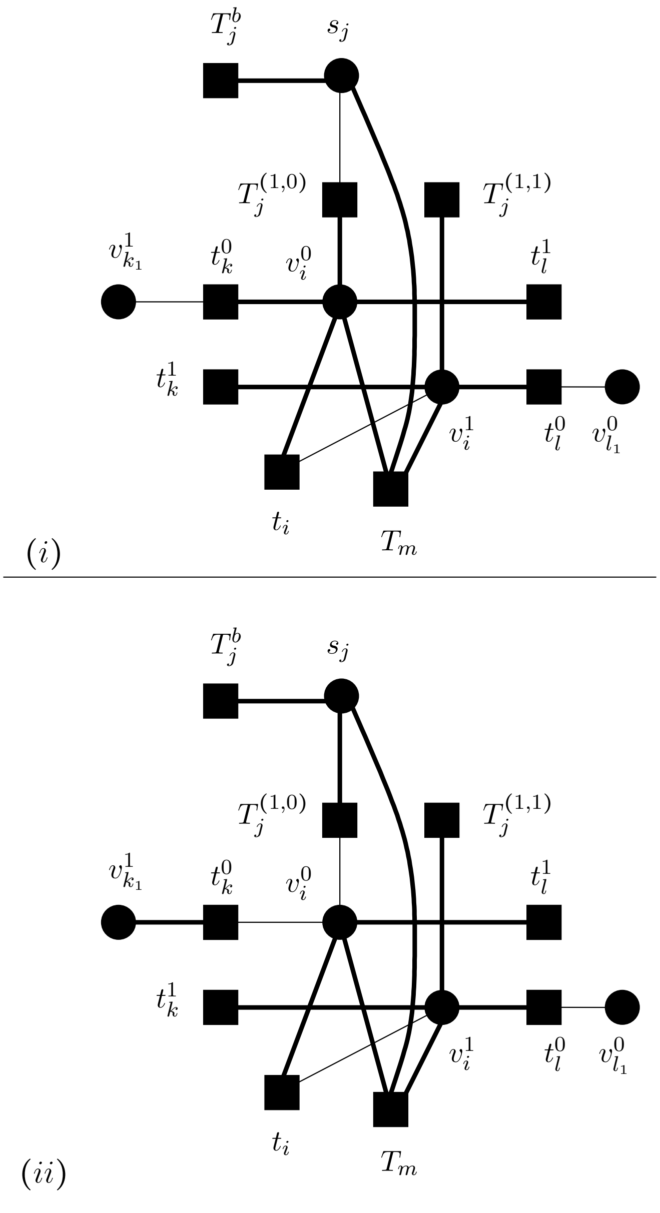


Figure 7.5: Situations before (i) and after (ii) uncoupling switchable terminals. The straight black lines are black edges of $\mathcal{G}_{\mathscr{L}}$, whereas thick lines represent black edges in $E(\mathcal{T})$. Finally, red edges included in $E(\mathcal{T})$ are indicated by dashed lines.

Uncoupling switchable terminals: Let v_i^0 be the variable vertex with at most one unswitchable terminal. Let t be the equation terminal which is connected to v_i^0 by a black edge in $\mathcal{T}$ and to the Steiner vertex v by an black edge in $E(G_{\mathcal{H}})\backslash E(\mathcal{T})$. If t is an equation terminal corresponding to an equation with two variables, we know how to handle this situation due to Transformation ❸. Therefore, we may assume that v is the special vertex s_j contained in $\mathcal{T}$. We remove the edge $\{T_j^{(1,0)}, v_i^0\}$ and insert the black edge $\{T_j^{(1,0)}, s_j\}$ in $E(\mathcal{T})$. The situation is displayed in Figure 7.5. After this conversion, our solution $\mathcal{T}'$ remains connected, since all paths that contained the edge $\{T_j^{(1,0)}, v_i^0\}$ can lead over $v_i^0 - T_m - s_j - T_j^{(1,0)}$. Furthermore, since we have not changed the number of edges, $\mathcal{T}'$ is a Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$ with the same cost as before.

Removing redundant variable vertices: The vertex v_i^0 is possibly connected to at most one equation terminal and its corresponding variable terminal. We notice that we have to deal with a similar situation which we solved in Transformation ❸. Hence, we can proceed analogously in order to obtain a normed Steiner tree $\mathcal{T}'$ for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$ with $c(\mathcal{T}') \leq c(\mathcal{T})$.■

By applying successively Transformation ❶ until ❺ to a solution for the instance $(\mathcal{G}_{\mathscr{L}}, S_{\mathscr{L}})$, we obtain a normed Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$. In addition, none of the defined transformations increases the cost of the actual solution. Since every transformation can be accomplished in polynomial time, we obtain the following lemma.

Lemma 7.8.2

Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem, $(G_{\mathscr{L}}, S_{\mathscr{L}})$ its corresponding instance of the* (1,2)-STEINER TREE *problem and $\mathcal{T}$ a Steiner tree for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$ with cost $c(T)$. It is possible to transform $\mathcal{T}$ in polynomial time into a normed Steiner tree $\mathcal{T}'$ for $S_{\mathscr{L}}$ in $\mathcal{G}_{\mathscr{L}}$ with $c(\mathcal{T}') \leq c(\mathcal{T})$.*

7.9 Proof of the Main Theorem

We now give the proof of Theorem 7.4.1.

Given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem with m_2 equations with two variables, m_3 equations with exactly three variables and n variables

$\{x_1, \ldots, x_n\}$, we construct the corresponding instance $(\mathcal{G}_{\mathscr{L}}, S_{\mathscr{L}})$ of the $(1,2)$-STEINER TREE problem as described in Section 7.6.

• Let $\phi : \{x_1, .., x_n\} \to \{0,1\}$ be an assignment to the variables of $\mathscr{L}$ leaving u equations in $\mathscr{L}$ unsatisfied. Then, we construct the Steiner tree $\mathcal{T}_\phi$ according to the description given in Section 7.7. By Lemma 7.7.1, we know that the cost of $\mathcal{T}_\phi$ is at most $c(\mathcal{T}_\phi) = 2n + 2m_2 + 8m_3 + u$.

• In order to prove the second part of Theorem 7.4.1, we are given a solution $\mathcal{T}$ of an instance $(\mathcal{G}_{\mathscr{L}}, S_{\mathscr{L}})$ of the $(1,2)$-STEINER TREE problem with cost $c(\mathcal{T}) = 2n + 2m_2 + 8m_3 + u$. Then, we apply Transformation ❶ until ❺ to $\mathcal{T}$. According to Lemma 7.8.2, we obtain in polynomial time a normed Steiner tree $\widehat{\mathcal{T}}$ with the properties described in Definition 7.8.1. The normed Steiner tree $\widehat{\mathcal{T}}$, however, enables us to specify the well-defined assignment $\psi_{\widehat{\mathcal{T}}}$ according to Definition 7.8.2. In order to relate the cost of $\widehat{\mathcal{T}}$ to the number of satisfied equations in $\mathscr{L}$ by $\psi_{\widehat{\mathcal{T}}}$, we intend to apply Lemma 7.8.1. For this reason, we construct the assignment star $\mathcal{S}(\widehat{\mathcal{T}})$ of $\widehat{\mathcal{T}}$ according to Definition 7.8.1, for which we analyze the additional cost in subgraphs $\mathcal{G}(\ell)$ for every ℓ in $\mathscr{L}$. By Lemma 7.8.1, we have to pay 2 units for subgraphs $\mathcal{G}(\ell_j^2)$ corresponding to equations with two variables in order to connect the variable vertices in $\mathcal{S}(\widehat{\mathcal{T}})$ with the equation terminals of $\mathcal{G}(\ell_j^2)$ assuming that the assignment $\psi_{\widehat{\mathcal{T}}}$ satisfies the equation ℓ_j^2. For subgraphs $\mathcal{G}(\ell_k^3)$ corresponding to equations with three variables, we have to pay 8 units under the same circumstances. If the underlying equation is left unsatisfied by $\psi_{\widehat{\mathcal{T}}}$, we have to spend one unit more. Hence, we conclude that the number of unsatisfied equations in $\mathscr{L}$ by $\psi_{\widehat{\mathcal{T}}}$ is at most $c(\widehat{\mathcal{T}}) - 2n - 2m_2 - 8m_3 = u$ and the proof of Theorem 7.4.1 follows. ■

CHAPTER 8

The Metric Dimension Problem

In this chapter, we study the approximation hardness of the METRIC DIMENSION problem restricted to graphs with maximum degree B. By constructing an approximation preserving reduction from the VERTEX COVER problem in 4-regular graphs, we prove that for every $B \geq 3$, the METRIC DIMENSION problem in graphs with maximum degree B is **APX**-hard. In addition, it implies the first explicit approximation lower bound for this restricted version of the problem. More precisely, we prove that the METRIC DIMENSION problem in graphs with maximum degree 3 is **NP**-hard to approximate to within any constant approximation ratio less than 353/352. Afterwards, we construct an improved approximation preserving reduction implying that it is **NP**-hard to approximate this restricted version of the problem to within any constant approximation ratio less than 153/152.

8.1 Introduction

A *resolving set* of connected graph $\mathcal{G}$ is a subset $S \subseteq V(\mathcal{G})$ such that for each pair of vertices $\{u, w\} \subseteq V(\mathcal{G})$, there exists some vertex $v \in S$ with $d_{\mathcal{G}}(v, u) \neq d_{\mathcal{G}}(v, w)$, where $d_{\mathcal{G}}(\cdot, \cdot)$ denotes the shortest path metric induced by $\mathcal{G}$. The minimum cardinality of a resolving set is called the *metric dimension* of $\mathcal{G}$, denoted by $dim_M(\mathcal{G})$. The METRIC DIMENSION problemis the following problem: given a connected graph $\mathcal{G}$ and the task is to find a resolving set S for $\mathcal{G}$ with $|S| = dim_M(\mathcal{G})$.

The notion of resolving sets were introduced independently by Harary and Melter [HM76], and by Slater [S75]. Applications of resolving sets arise in various areas including coin weighing problems [SS63], drug discovery [CEJO00], robot navigation [KRR96], network discovery and verification [BEE+05], connected joins in graphs [ST04], and strategies for the Mastermind game [C83]. The METRIC DIMENSION problem has been widely investigated from the graph theoretical point of view [S88a, FGO06, CHM+07, HMP+10, T08, CGH08, CHM+09]. Bailey and Cameron [BC11] note in their survey an interesting connection to graph isomorphism and group theory. In particular, Babai [B80] proved that given a graph $\mathcal{G}$ with n vertices and $dim_M(\mathcal{G}) \leq k$, isomorphism of $\mathcal{G}$ against any graph can be decided in $\mathrm{O}(n^{k+2})$ time.

So far, only a few papers discuss the computational complexity issues of this problem. The **NP**-hardness of the decision version of the METRIC DIMENSION problem was first mentioned in Gary and Johnson [GJ79]. An explicit reduction from the 3SAT problem was given by Khuller, Raghavachari and Rosenfeld [KRR96]. In addition, they design an efficient approximation algorithm for the METRIC DIMENSION problem with approximation ratio $2\ln(n) + \Theta(1)$ based on the well-known greedy algorithm for the SET COVER problem. On the other hand, they gave a polynomial time exact algorithm for the METRIC DIMENSION problem restricted to trees and to graphs with maximum degree 2. Beerliova et al. [BEE+05] proved that it is **NP**-hard to approximate the METRIC DIMENSION problem (which they call the *Network Verification* problem) with an approximation ratio $o(\log(n))$. Hauptmann, Schmied and Viehmann [HSV12] improved the approximation lower bound to $(1-\epsilon)\ln(n)$ for all $\varepsilon > 0$, under the assumption $\mathbf{NP} \notin \mathbf{DTIME}(n^{log(log(n))})$. In the same work, an efficient approximation algorithm for the METRIC DIMENSION problem was given with a matching approximation ratio of $(1 + o(1))\ln(n)$ settling the approximation complexity of the problem.

The METRIC DIMENSION problem restricted to point sets in $\mathbb{R}^d$ was considered in [HSV12] and proved to be solvable in polynomial time, whenever d is constant. Díaz, Pottonen, Serna and van Leeuwen [DPSL11] studied the hardness of planar instances of the METRIC DIMENSION problem. They showed that the restricted version of the problem is **NP**-hard, whereas outerplanar instances are solvable in polynomial time.

By constructing an approximation preserving reduction from the VERTEX COVER problem in 4-regular graphs, Hauptmann, Schmied and Viehmann [HSV12] proved that for all $B \geq 3$, the METRIC DIMENSION problem in graphs with maximum degree B is **APX**-hard. The reduction implies that it is **NP**-hard to approximate the METRIC DIMENSION problem in graphs with maximum degree 3 within any constant approximation ratio better than 353/352. Subsequently, Hartung and Nichterlein [HN12] gave an improved inapproximability result which implies that it is **NP**-hard to approximate the METRIC DIMENSION problem in graphs with maximum degree 3 with an approximation ratio $o(\log(n))$.

In this work, we present the approximation preserving reduction from the VERTEX COVER problem in 4-regular graphs to the METRIC DIMENSION problem in graphs with maximum degree 3 due to Hauptmann, Schmied and Viehmann [HSV12]. Based on the inapproximability results due to Chlebik and Chlebikova [CC06], we prove that the METRIC DIMENSION problem in graphs with maximum degree 3 is **NP**-hard to approximate within any constant approximation ratio better than 353/352. Afterwards, we construct an improved approximation preserving reduction yielding an approximation lower bound of 153/152. We point out that both inapproximability results are prior to the result due to Hartung and Nichterlein [HN12].

8.2 Outline of this Chapter

The chapter is organized as follows. In Section 8.4, we formulate our main results. In Section 8.5, we construct our first approximation preserving reduction. In Section 8.6, we give an improved approximation lower bound for the METRIC DIMENSION problem in graphs with maximum degree 3.

8.3 Notations and Definitions

Before we formulate our contributions, we are going to fix the notation used in this chapter and provide some definitions.

Given a connected graph $\mathcal{G}$, we say that a vertex $v \in V(\mathcal{G})$ *resolves*, *splits* or *distinguishes* a pair $\{u, w\} \in \binom{V(\mathcal{G})}{2}$ if $d_{\mathcal{G}}(v, u) \neq d_{\mathcal{G}}(v, w)$ holds. Furthermore, we say that a set $S \subseteq V(\mathcal{G})$ resolves all pairs $p \in P \subseteq \binom{V(\mathcal{G})}{2}$ if for all $p \in P$, there exists some $s \in S$ splitting p. A subset $S \subseteq V(\mathcal{G})$ of vertices is said to be *resolving* for $\mathcal{G}$ if S resolves every pair $p \in \binom{V(\mathcal{G})}{2}$. The minimum cardinality of a set S, that is resolving for $\mathcal{G}$, is defined as the *metric dimension* of $\mathcal{G}$, denoted by $dim_M(\mathcal{G})$.

According to Definition 4.1.1, the METRIC DIMENSION problem is defined as follows.

Definition 8.3.1 (METRIC DIMENSION problem)

Instances: *A connected graph* $\mathcal{G}$

Solutions: $S \subseteq V(\mathcal{G})$ *such that for all pairs* $\{v, w\} \in \binom{V(\mathcal{G})}{2}$,
there exists some $s \in S$ *with* $d_{\mathcal{G}}(s, v) \neq d_{\mathcal{G}}(s, w)$

Task: *Minimize* $|S|$

For a connected graph $\mathcal{G}$ and a non-empty set $S \subseteq V(\mathcal{G})$, we introduce the notion of distance classes.

Definition 8.3.2 (Distance Classes)
Let $\mathcal{G}$ *be a connected graph and* $S \neq \varnothing$ *a subset of* $V(\mathcal{G})$*. The sets* $V_1, \dots, V_k$ *are called the distance classes induced by* S *in* $\mathcal{G}$ *if the following conditions hold.*

(i) *The sets* $V_1, \dots, V_k$ *define a partition of* $V(\mathcal{G})$.

(ii) *For all* $i \in [k]$, *all pairs* $\{u, w\} \in \begin{pmatrix} V_i \\ 2 \end{pmatrix}$ *and all* $t \in S$, *we have that* $d_{\mathcal{G}}(u, t) = d_{\mathcal{G}}(u, t)$.

(iii) S *resolves all pairs* $p \in \left\{ \{u, w\} \in \begin{pmatrix} V(\mathcal{G}) \\ 2 \end{pmatrix} \middle| \, u \in V_i, w \in V_j, \{i, j\} \in \begin{pmatrix} k \\ 2 \end{pmatrix} \right\}$.

Let $\mathcal{G}$ be a connected graph, $S \neq \varnothing$ a subset of $V(\mathcal{G})$ and $V_1, \dots, V_k$ the distance classes induced by S in $\mathcal{G}$. Then, we say that V_i is *unresolved* by S if $|V_i| \geq 2$ holds.

8.4 Our Contributions

We present our first result concerning the approximation hardness of the METRIC DIMENSION problem restricted to graphs with maximum degree 3, which we are going to prove in the subsequent section.

Theorem 8.4.1
Given a 4*-regular graph* $\mathcal{G}$, *it is possible to construct in polynomial time an*

instance $\mathcal{G}^M$ *of the* METRIC DIMENSION *problem with the following properties.*

(i) *The constructed graph* $\mathcal{G}^M$ *has maximum degree* 3.

(ii) *Given a vertex cover* C *of* $\mathcal{G}$, *it is possible to construct in polynomial time a resolving set of* $\mathcal{G}^M$ *with size at most* $|C| + |E(\mathcal{G}^M)| + |V(\mathcal{G}^M)|$.

(iii) *From every resolving set* R *of* $\mathcal{G}^M$, *it is possible to construct in polynomial time a vertex cover of* $\mathcal{G}$ *of size at most* $|R| - |E(\mathcal{G})| - |V(\mathcal{G})|$.

In order to prove that the METRIC DIMENSION problem restricted to graphs with maximum degree 3 is **NP**-hard to approximate with some constant, we will use the following inapproximability result due to Chlebik and Chlebiková [CC06].

Theorem 8.4.2 ([CC06])
Let $\mathcal{G}$ *be a* 4*-regular graph. For every* $\delta \in (0, \frac{1}{2})$, *the following cases* (i) *and* (ii) *are* **NP**-*hard to decide.*

(i) *Every vertex cover of* $\mathcal{G}$ *has size at least* $|V(\mathcal{G})| \left(\dfrac{53 - 2\delta}{100} \right)$.

(ii) *The size of a minimum vertex cover of* $\mathcal{G}$ *is at most* $|V(\mathcal{G})| \left(\dfrac{52 + 2\delta}{100} \right)$.

By means of Theorem 8.4.1, we are going to prove the following statement.

Corollary 8.4.1
For every $B \geq 3$, *the* METRIC DIMENSION *problem restricted to graphs with maximum degree* B *is* **APX**-*hard and* **NP**-*hard to approximate to within any constant approximation ratio less than* $353/352$.

Proof of Corollary 8.4.1.
For a constant $\varepsilon > 0$, we select $\delta \in \left(0, \frac{1}{2}\right)$ such that the following holds.

$$\frac{353 - 2\delta}{352 + 2\delta} \leq \frac{353}{352} - \varepsilon$$

Given an instance $\mathcal{G}$ of the VERTEX COVER problem in 4-regular graphs described in Theorem 8.4.2, we construct in polynomial time an instance of the METRIC DIMENSION problem with the properties guaranteed by Theorem 8.4.1. According to Theorem 8.4.2, either the metric dimension of the constructed graph $\mathcal{G}^M$ is at least

$$|E(\mathcal{G})| + |V(\mathcal{G})| + |V(\mathcal{G})|\left(\frac{53 - 2\delta}{100}\right) = 3 \cdot |V(\mathcal{G})| + |V(\mathcal{G})|\left(\frac{53 - 2\delta}{100}\right)$$

or $dim_M(\mathcal{G})$ is bounded from above by

$$|E(\mathcal{G})| + |V(\mathcal{G})| + |V(\mathcal{G})|\left(\frac{52 + 2\delta}{100}\right) = |V(\mathcal{G})|\left(\frac{352 + 2\delta}{100}\right).$$

Furthermore, we know that it is **NP**-hard to decide which case above holds. Therefore, we conclude that it is **NP**-hard to approximate the METRIC DIMENSION problem in graphs with maximum degree 3 to within any constant approximation ratio less than $(353/352 - \varepsilon)$ for every constant $\varepsilon > 0$. Recall from Theorem 5.6.2 that the VERTEX COVER problem in 4-regular graphs is **APX**-hard. Accordingly, it implies the **APX**-hardness of the METRIC DIMENSION problem in graphs with maximum degree 3. ■

Afterwards, in Section 8.6, we will construct an approximation preserving reduction implying an improved approximation lower bound. We now formulate our second result.

Theorem 8.4.3
Given a 4-regular graph $\mathcal{G}$, it is possible to construct in polynomial time an instance $\mathcal{G}^I$ of the METRIC DIMENSION *problem with the following properties.*

(i) *The corresponding graph $\mathcal{G}^I$ has maximum vertex degree* 3.

(ii) *Given a vertex cover C of $\mathcal{G}$, then, it is possible to construct an resolving set $R^I_{\mathcal{G}}$ of $\mathcal{G}^I$ with size at most $|C| + |V(\mathcal{G})|$.*

(iii) *From every resolving set R of $\mathcal{G}^I$, it is possible to construct in polynomial time a vertex cover $C^I_{\mathcal{G}}$ of $\mathcal{G}$ with size at most $|R| - |V(\mathcal{G})|$.*

Analogously to the proof of Corollary 8.4.1, we combine Theorem 8.4.2 with Theorem 8.4.3 and obtain the following approximation hardness result.

Corollary 8.4.2
For every $B \geq 3$, the METRIC DIMENSION *problem restricted to graphs with maximum degree B is* **NP***-hard to approximate to within any constant approximation ratio less than* $153/152$.

8.5 APX-Hardness and Explicit Lower Bounds

Before we start to describe our approximation preserving reduction, we first give an outline of the construction and try to build some intuition.

8.5.1 High-Level View of the Reduction

In order to reduce the VERTEX COVER problem to the METRIC DIMENSION problem, we have to convert a covering problem into a splitting problem. It will be accomplished by introducing pairs of nodes for every element that needs to be covered. Moreover, these pairs can only be distinguished by special vertices representing a vertex cover of the original graph.

Given a 4-regular graph $\mathcal{G}$ as an instance of VERTEX COVER problem, we introduce for each vertex $v \in V(\mathcal{G})$ and each edge $e \in E(\mathcal{G})$ an associated graph $\mathcal{G}_v^M$ and $\mathcal{G}_e^M$, respectively. These associated graphs will be connected and build the corresponding instance $\mathcal{G}^M$ of the METRIC DIMENSION problem. Every graph $\mathcal{G}_e^M$ contains a pair of vertices p_e that represents the edge e of the original graph that is supposed to be covered. The only vertices of $\mathcal{G}^M$ that are able to distinguish the pair p_e can be associated to vertices $a \in V(\mathcal{G})$ with $a \in e$.

8.5.2 Constructing $\mathcal{G}^M$ from a 4-Regular Graph $\mathcal{G}$

Given a 4-regular graph $\mathcal{G}$ as an instance of the VERTEX COVER problem, we are going to define the corresponding instance $\mathcal{G}^M$ of the METRIC DIMENSION problem. As we will see, the construction of $\mathcal{G}^M$ can be accomplished in polynomial time and the degree of every vertex $v \in V(\mathcal{G}^M)$ is bounded from above by 3. At the end of this section, we prove some important properties of the corresponding graph $\mathcal{G}^M$. Let us start with the description of $\mathcal{G}^M$.

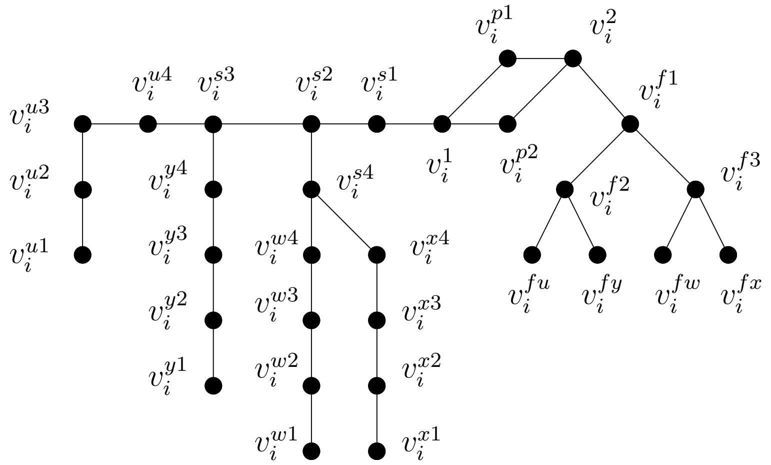


Figure 8.1: Subgraph $\mathcal{G}_i^M$ with $i \in V(\mathcal{G})$ and $N_{\mathcal{G}}(i) = \{u, y, w, x\}$.

Construction of the Corresponding Graph $\mathcal{G}^M$

Let $\mathcal{G}$ be a 4-regular graph. For every vertex $v \in V(\mathcal{G})$ and every edge $e \in E(\mathcal{G})$, we introduce the corresponding subgraphs $\mathcal{G}_v^M$ and $\mathcal{G}_e^M$, respectively. Then, $\mathcal{G}^M$ is defined by connecting the subgraphs $\mathcal{G}_v^M$ and $\mathcal{G}_e^M$ in an appropriate way specified later on.

In the following, we refer to the subgraph $\mathcal{G}_v^M$ with $v \in V(\mathcal{G})$ as the *vertex graph* of v and to $\mathcal{G}_e^M$ as the *edge graph* of $e \in E(\mathcal{G})$. Let us give the description of the subgraphs $\mathcal{G}_x^M$ with $x \in V(\mathcal{G}) \cup E(\mathcal{G})$ starting with the vertex graphs.

Vertex Graph $\mathcal{G}_v^M$

Let $i \in V(\mathcal{G})$ be a vertex in $\mathcal{G}$ and $N_{\mathcal{G}}(i) = \{\, u, y, w, x \,\}$ the set of its neighbours in $\mathcal{G}$. Then, we introduce the subgraph $\mathcal{G}_i^M$ of i specified in Figure 8.1.

In the remainder, we refer to v_i^{p1} and v_i^{p2} as the *main vertices* of $\mathcal{G}_i^M$. For notational simplicity, we introduce for every vertex $i \in V(\mathcal{G})$, the sets

$$D_{\mathcal{G}}(i) = \{v_i^{\alpha k}, v_i^{sk} \mid \alpha \in N_{\mathcal{G}}(i), k \in [4]\} \text{ and } D_{\mathcal{G}}^{-}(i) = D_{\mathcal{G}}(i) \backslash \{v_i^{x1} \mid x \in N_{\mathcal{G}}(i)\}.$$

Finally, we remark that the approximation preserving properties of the construction are independent of the order of appearance of the vertices u, y, w and x.

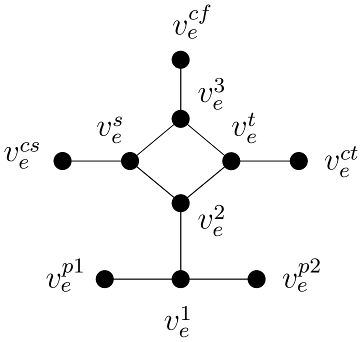


Figure 8.2: Subgraph $\mathcal{G}_e^M$ with $e \in E(\mathcal{G})$.

Edge Graph $\mathcal{G}_e^M$

Let $e \in E(\mathcal{G})$ be an edge in $\mathcal{G}$. Then, we define $\mathcal{G}_e^M$ as displayed in Figure 8.2.

We refer to v_e^{p1} and v_e^{p2} as the *main vertices* of $\mathcal{G}_e^M$. In addition, we introduce the *split pairs* of $\mathcal{G}_e^M$ defined by $p_e^1 = \{v_e^{cs}, v_e^{ct}\}$ and $p_e^2 = \{v_e^s, v_e^t\}$. For notational simplicity, for every $e \in E(\mathcal{G})$, we introduce the set $D_{\mathcal{G}}(e) = p_e^1 \cup p_e^2$. Finally, we denote by $R_m(\mathcal{G})$ the set of the *first main vertices* of $\mathcal{G}^M$ given by

$$R_m(\mathcal{G}) = \{v_x^{p1} \mid x \in V(\mathcal{G}) \cup E(\mathcal{G})\}.$$

The disjoint union of the vertices in $V(\mathcal{G}_x^M)$ with $x \in V(\mathcal{G}) \cup E(\mathcal{G})$ already defines the vertex set $V(\mathcal{G}^M)$ of $\mathcal{G}^M$. Nevertheless, we still have not specified all edges of $\mathcal{G}^M$. So far, we only have

$$\bigcup_{x \in V(\mathcal{G}) \cup E(\mathcal{G})} E(\mathcal{G}_x^M) \subset E(\mathcal{G}^M).$$

The remaining edges are needed to connect the subgraphs in order to form the connected graph $\mathcal{G}^M$. We are going to describe how the subgraphs $\mathcal{G}_x^M$ with $x \in V(\mathcal{G}) \cup E(\mathcal{G})$ are connected appropriately.

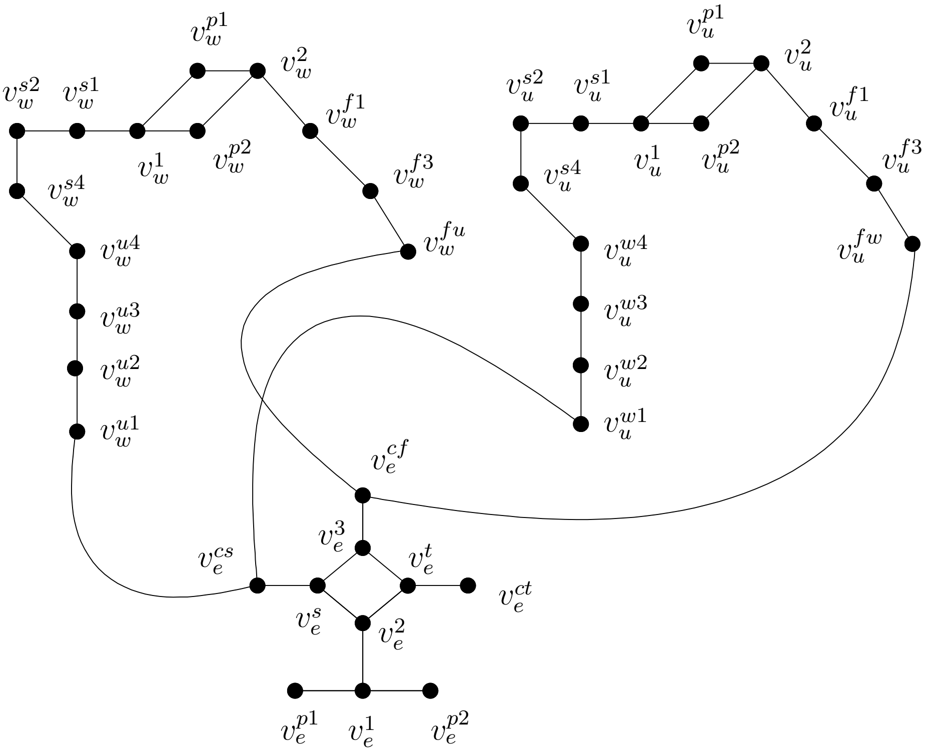


Figure 8.3: Connecting $\mathcal{G}_u^M$, $\mathcal{G}_w^M$ and $\mathcal{G}_e^M$, where $e = \{w, u\}$.

Let $e = \{u, w\} \in E(\mathcal{G})$ be an edge in $\mathcal{G}$. Then, we connect the subgraphs $\mathcal{G}_u^M$ and $\mathcal{G}_w^M$ both with $\mathcal{G}_{\{u,w\}}^M$ by adding the edges $\left\{v_u^{w1}, v_{\{u,w\}}^{cs}\right\}$, $\left\{v_u^{fw}, v_{\{u,w\}}^{cf}\right\}$, $\left\{v_w^{u1}, v_{\{u,w\}}^{cs}\right\}$ and $\left\{v_w^{fu}, v_{\{u,w\}}^{cf}\right\}$ to $E(\mathcal{G}^M)$. A part of the construction is displayed in Figure 8.3.

By implementing the former construction for every edge $e \in E(\mathcal{G})$, we obtain in this way the graph $\mathcal{G}^M$. Since the degree of every vertex of $\mathcal{G}$ is bounded from above by 4, we see that the maximum degree of $\mathcal{G}^M$ is bounded by 3. Finally, we note that given $\mathcal{G}$, the construction of $\mathcal{G}^M$ can be accomplished in polynomial time.

Properties of the Corresponding Graph $\mathcal{G}^M$

In the following, we prove some crucial properties of the subgraphs $\mathcal{G}_x^M$ with $x \in V(\mathcal{G}) \cup E(\mathcal{G})$. In particular, we will see that every resolving set of $\mathcal{G}^M$ includes at least one main vertex of $\mathcal{G}_x^M$ for each $x \in V(\mathcal{G}) \cup E(\mathcal{G})$. Let us give the precise statement.

Lemma 8.5.1

Let $\mathcal{G}$ be a 4-regular graph, $\mathcal{G}^M$ its corresponding graph and R a resolving set of $\mathcal{G}^M$. Then, for every $x \in V(G) \cup E(G)$, we have that

$$R \cap \{v_x^{p1}, v_x^{p2}\} \neq \varnothing.$$

Proof of Lemma 8.5.1.

Let $\mathcal{G}$ be a 4-regular graph and R a resolving set of the corresponding graph $\mathcal{G}^M$. Given an edge $e \in E(\mathcal{G})$, we consider the subgraph $\mathcal{G}_e^M$ in $\mathcal{G}^M$ and the pair $\{v_e^{p2}, v_e^{p1}\}$ of vertices. Notice that for every $v \in V(\mathcal{G}^M)\backslash\{v_e^{p2}, v_e^{p1}\}$, we have $d_{\mathcal{G}^M}(v_e^{p1}, v) = d_{\mathcal{G}^M}(v_e^{p2}, v)$. Accordingly, the only vertices resolving this pair are v_e^{p2} and v_e^{p1} themselves. Therefore, we need to include either v_e^{p2} or v_e^{p1} in a resolving set of $\mathcal{G}^M$.

Finally, we note that if we have given a vertex graph $\mathcal{G}_v^M$ with $v \in V(\mathcal{G})$, we may use a similar argumentation. ■

Next, we are going to prove another crucial property of the edge graph $\mathcal{G}_e^M$. The following lemma enables us to concentrate on resolving only one of the split pairs of a given edge graph.

Lemma 8.5.2

Let $\mathcal{G}$ be a 4-regular graph and e an edge in $\mathcal{G}$. If a vertex $v \in V(\mathcal{G}^M)$ distinguishes the pair p_e^1, then, v resolves both split pairs of $\mathcal{G}_e^M$.

Proof of Lemma 8.5.2.

In order to prove Lemma 8.5.2, we proceed by case analysis. Let $\mathcal{G}$ be a 4-regular graph and $e \in E(\mathcal{G})$. Let us start with vertices $v \in V(\mathcal{G}_e^M)$.

Case $(v \in V(\mathcal{G}_e^M))$**:**

Since we have $d_{\mathcal{G}^M}(v_e^s, v) = d_{\mathcal{G}^M}(v, v_e^t)$ and $d_{\mathcal{G}^M}(v_e^{cs}, v) = d_{\mathcal{G}^M}(v, v_e^{ct})$ for any vertex

$$v \in \left\{\ v_e^{p1},\ v_e^{p2},\ v_e^1,\ v_e^2,\ v_e^{cf},\ v_e^3\ \right\},$$

neither p_e^1 nor p_e^2 can be distinguished by v. Hence, the claim holds for these vertices. The same holds for $v \in p_e^1 \cup p_e^2$, because every $v \in p_e^1 \cup p_e^2$ splits both pairs. Therefore, it remains to consider vertices $v \in V(\mathcal{G}^M)\backslash V(\mathcal{G}_e^M)$.

Case ($v \in V(\mathcal{G}^M)\backslash V(\mathcal{G}_e^M)$)**:**
Let us assume that there is a vertex $v \in V(\mathcal{G}^M)\backslash V(\mathcal{G}_e^M)$ such that $d_{\mathcal{G}^M}(v, v_e^s) \neq d_{\mathcal{G}^M}(v, v_e^t)$ holds, as otherwise, we have nothing to prove. We claim that there is a shortest path $\mathcal{P}_{s-e}^v$ from v to v_e^s, in which the vertex v_e^{cf} is not contained. The former statement will be proved by contradiction: Let us assume that all shortest paths from v to v_e^t all include the vertex v_e^{cf}. But this leads to a contradiction, since we obtain $d_{\mathcal{G}^M}(v, v_e^s) = d_{\mathcal{G}^M}(v, v_e^t)$. Consequently, we may assume that there is a path $\mathcal{P}_{t-e}^v$ from v to v_e^t not including v_e^{cf}. It implies that $\mathcal{P}_{t-e}^v$ must contain the vertices v_e^s and v_e^3. Let the path $\mathcal{P}_{t-e}^v$ be given by

$$\mathcal{P}_{t-e}^v \ = \ v - v_1 - \ldots - v_k - v_e^{cs} - v_e^s - v_e^3 - v_e^t.$$

The situation is displayed in Figure 8.4. But then, $\mathcal{P}_{s-e}^v = v - v_1 - \ldots - v_k - v_e^{cs} - v_e^s$ is a shortest path from v to v_e^s not containing v_e^{cf} contradicting our assumption of non-existence.

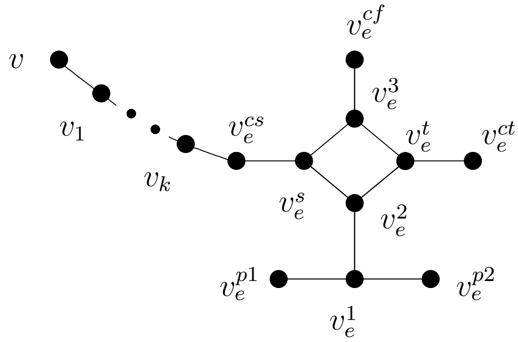


Figure 8.4: The path $\mathcal{P}_{t-e}^v = v - v_1 - \ldots - v_k - v_e^{cs} - v_e^s - v_e^3 - v_e^t$.

Since we know that there exists a shortest path from v to v_e^s not including v_e^{cf} and $d_{\mathcal{G}^M}(v, v_e^s) \neq d_{\mathcal{G}^M}(v, v_e^t)$, we conclude that $d_{\mathcal{G}^M}(v, v_e^s) < d_{\mathcal{G}^M}(v, v_e^t)$. But this means that we have $d_{\mathcal{G}^M}(v, v_e^{cs}) < d_{\mathcal{G}^M}(v, v_e^{ct})$ as well and the proof of Lemma 8.5.2 follows. ■

8.5.3 Constructing a Resolving Set of $\mathcal{G}^M$

Given a 4-regular graph $\mathcal{G}$ and a vertex cover C of $\mathcal{G}$, we are going to construct the resolving set $R(C)$ of the corresponding graph $\mathcal{G}^M$ based on the vertex cover C.

We first give the definition of the set $R(C)$. Afterwards, we prove that $R(C)$ is indeed resolving for $\mathcal{G}^M$.

Definition 8.5.1 (Resolving Set $R(C)$)
Let $\mathcal{G}$ be a 4-regular graph, $\mathcal{G}^M$ the corresponding graph constructed as described in Section 8.5.2. Given a vertex cover C of $\mathcal{G}$, the resolving set $R(C)$ is defined as follows.

$$R(C) = R_m(\mathcal{G}) \cup \{v_u^{s1} \mid u \in C\}$$

In the remainder of this section, we will prove that the former defined set $R(C)$ is resolving for $\mathcal{G}^M$.

Lemma 8.5.3
Let $\mathcal{G}$ be a 4-regular graph and C a vertex cover of $\mathcal{G}$. Then, the vertex set $R(C)$ is resolving for $\mathcal{G}^M$.

In what follows, it will be convenient to separate the proof of Lemma 8.5.3 into two parts. First of all, we show that the set $R_m(\mathcal{G})$, defined by

$$R_m(\mathcal{G}) = \{v_x^{p1} \mid x \in V(\mathcal{G}) \cup E(\mathcal{G})\},$$

resolves every pair of vertices in $\mathcal{G}^M$ except $p_e^1 = \{v_e^{cs}, v_e^{ct}\}$ and $p_e^2 = \{v_e^s, v_e^t\}$ with $e \in E(\mathcal{G})$. At this point, we recall that according to Lemma 8.5.2, we only have to take care of the remaining pairs p_e^2 in the sense that every vertex $v \in V(\mathcal{G}^M)$ resolving p_e^2 distinguishes also the pair p_e^1. Finally, we prove that given a vertex $u \in V(\mathcal{G})$ and an edge $e \in E(\mathcal{G})$, v_u^{s1} distinguishes p_e^2 if and only if u is contained in e.

Since $R_m(\mathcal{G})$ is a subset of $R(C)$ and C a vertex cover of $\mathcal{G}$, we conclude that $R(C)$ is resolving for $\mathcal{G}^M$. Let us start with the following Lemma.

Lemma 8.5.4

Let $\mathcal{G}$ be a 4-regular graph and $\mathcal{G}^M$ its corresponding graph. Then, the set $R_m(\mathcal{G})$ resolves every pair $p \in \binom{V(\mathcal{G}^M)}{2}$ except $p \in \{\, p_e^1,\, p_e^2 \mid e \in E(\mathcal{G})\,\}$.

Proof of Lemma 8.5.4.
Let $\mathcal{G}$ be a 4-regular graph and $\mathcal{G}^M$ its corresponding graph. First of all, we prove that the set $R_m(\mathcal{G})$ resolves all pairs

$$p \in \binom{V(\mathcal{G}^M)}{2} \setminus \left(\bigcup_{e \in E(\mathcal{G})} \{\, p_e^1,\, p_e^2 \,\} \right).$$

Afterwards, we show that all pairs $p \in \{\, p_e^1,\, p_e^2 \mid e \in E(\mathcal{G})\}$ remain unresolved by $R_m(\mathcal{G})$. The first part will be accomplished by case analysis on pairs $p \in \binom{V(\mathcal{G}^M)}{2} \setminus \big(\bigcup_{e \in E(\mathcal{G})} \{p_e^1, p_e^2\} \big)$ starting with the case $p \in \binom{V(\mathcal{G}_e^M)}{2} \setminus \{p_e^1, p_e^2\}$ for a fixed $e \in E(\mathcal{G})$.

1.Case $\big(p \in \binom{V(\mathcal{G}_e^M)}{2} \setminus \{p_e^1, p_e^2\}$ **with** $e \in E(\mathcal{G})\big)$**:**
The distance classes induced by $\{v_e^{p1}\}$ in $\mathcal{G}_e^M$ are given by

$$\{v_e^{p1}\}, \quad \{v_e^1\}, \quad \{v_e^2, v_e^{p2}\}, \quad \{v_e^s, v_e^t\}, \quad \{v_e^3, v_e^{cs}, v_e^{ct}\} \quad \text{and} \quad \{v_e^{cf}\}.$$

Finally, this distance classes are splitted by v_i^{p1} with $i \in e$ into

$$\{v_e^{p1}\}, \quad \{v_e^1\}, \quad \{v_e^2\}, \quad \{v_e^{p2}\}, \quad \{v_e^s, v_e^t\}, \quad \{v_e^{cs}, v_e^{ct}\} \quad \{v_e^3\}, \quad \text{and} \quad \{v_e^{cf}\}.$$

2.Case $\big(p \in \binom{V(\mathcal{G}_i^M)}{2}$ **with** $i \in V(\mathcal{G})\big)$**:**
Let $i \in V(\mathcal{G})$ be a vertex with neighbours u, y, w and x in $\mathcal{G}$. Furthermore, we have that $e = \{w, i\}$ and $a = \{u, i\}$. The corresponding subgraphs $\mathcal{G}_i^M$, $\mathcal{G}_e^M$ and $\mathcal{G}_a^M$ are connected as displayed in Figure 8.5. Let us consider the distance classes induced by $\{v_i^{p1}\}$ in $\mathcal{G}_i^M$ given below.

$$\{v_i^{p1}\},\ \{\, v_i^1,\, v_i^2 \,\},\ \{\, v_i^{f1},\, v_i^{p2},\, v_i^{s1} \,\},\ \{\, v_i^{f2},\, v_i^{f3},\, v_i^{s2} \,\},$$

$$\{\, v_i^{fu},\, v_i^{fy},\, v_i^{fw},\, v_i^{fx},\, v_i^{s3},\, v_i^{s4} \,\},\ \{\, v_i^{u4},\, v_i^{y4},\, v_i^{w4},\, v_i^{x4} \,\},\ \{\, v_i^{u3},\, v_i^{y3},\, v_i^{w3},\, v_i^{x3} \,\}$$

$$\{\, v_i^{u2},\, v_i^{y2},\, v_i^{w2},\, v_i^{x2} \,\} \text{ and } \{\, v_i^{u1},\, v_i^{y1},\, v_i^{w1},\, v_i^{x1} \,\}.$$

By including v_e^{p1} in our resolving set, the remaining unresolved distance classes are

$$\{\,v_i^{fy},\, v_i^{fu}\,\},\ \{\,v_i^{y4},\, v_i^{u4}\},\ \{\,v_i^{y3},\, v_i^{u3}\,\},\ \{\,v_i^{y2},\, v_i^{u2}\,\}\ \text{ and }\{\,v_i^{y1},\, v_i^{u1}\,\}.$$

Finally, we see that these remaining pairs can be resolved by v_a^{p1}.

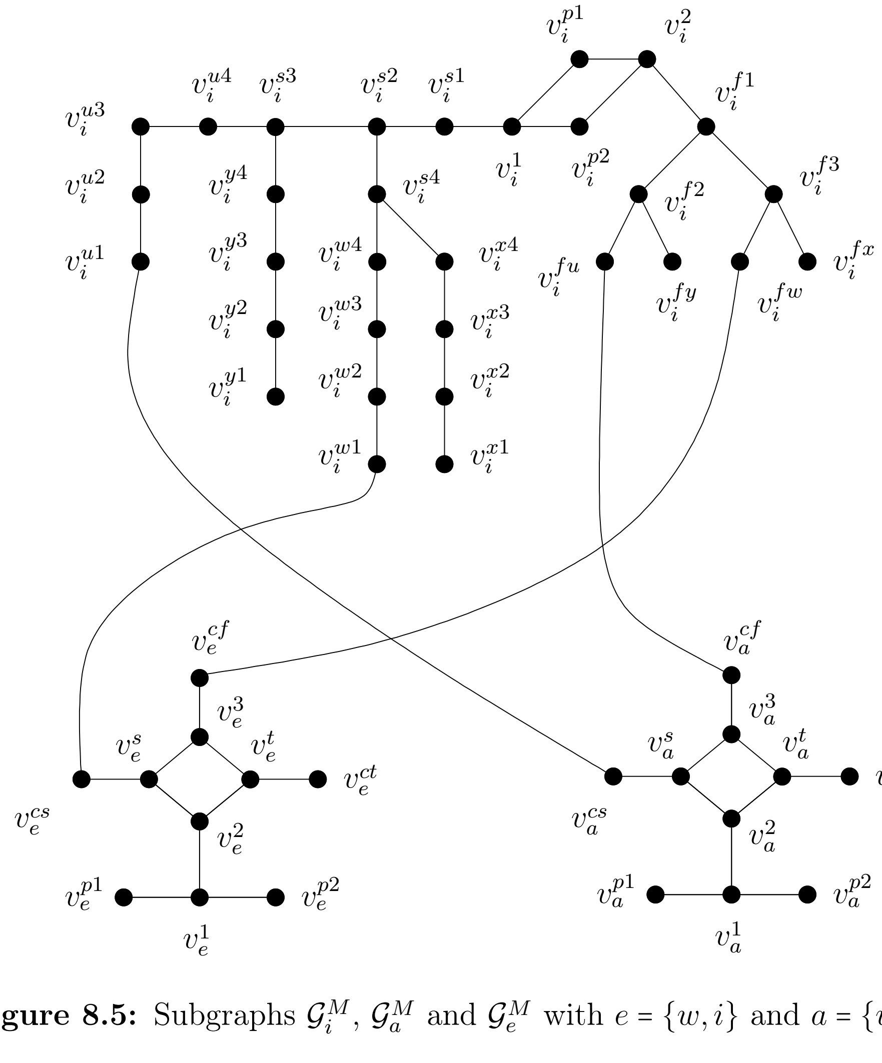


Figure 8.5: Subgraphs $\mathcal{G}_i^M$, $\mathcal{G}_a^M$ and $\mathcal{G}_e^M$ with $e = \{w, i\}$ and $a = \{u, i\}$.

3.Case $\big(\{x, y\}$, where $x \in V(\mathcal{G}_e^M)$, $y \in V(\mathcal{G}_{e'}^M)$ and $\{e, e'\} \subseteq E(\mathcal{G})\big)$: Notice that for every $x \in V(\mathcal{G}_e^M)$ and $y \in V(\mathcal{G}_{e'}^M)$, we have $d_{\mathcal{G}^M}(x, v_e^{p1}) \leq 5$ and $d_{\mathcal{G}^M}(y, v_e^{p1}) \geq 9$. Hence, we can choose v_e^{p1} to distinguish all these pairs.

4.Case $\big(\{x,y\}$, where $x \in V(\mathcal{G}_e^M)$, $y \in V(\mathcal{G}_i^M)$, $i \in V(\mathcal{G})$ and $e \in E(\mathcal{G})\big)$:
From the former case, we know that $d_{\mathcal{G}^M}(x, v_e^{p1}) \le 5$ holds. If we have $d_{\mathcal{G}^M}(y, v_e^{p1}) > 5$ for every $y \in V(\mathcal{G}_i^M)$, we have nothing to prove.

Therefore, we may assume that there is a vertex $v_i^{w1} \in V(\mathcal{G}_i^M)$ such that $\{v_e^{cs}, v_i^{w1}\} \in E(\mathcal{G}^M)$ holds. But then, v_e^{p1} leaves only the pair $\{v_i^{w1}, v_e^{cf}\}$ unresolved. The remaining pair can be distinguished by v_i^{p1} since we obtain $d_{\mathcal{G}^M}(v_e^{cf}, v_i^{p1}) = 5$ and $d_{\mathcal{G}^M}(v_i^{w1}, v_i^{p1}) = 8$.

5.Case $\big(\{x,y\}$, where $x \in V(\mathcal{G}_j^M)$, $y \in V(\mathcal{G}_i^M)$ and $\{i,j\} \subseteq V(\mathcal{G})\big)$:
Let $i \in V(\mathcal{G})$ be a vertex in $\mathcal{G}$ and $\{u, y, w, x\}$ its neighbors in $\mathcal{G}$. Then, we introduce the set of vertices X_6 defined as follows.

$$X_6 = \left\{v_i^{yk}, v_i^{uk}, v_i^{wk}, v_i^{xk} \mid k \in [3]\right\}$$

Notice that we have $d_{\mathcal{G}^M}(x, v_i^{p1}) \le 5$ for all $x \in V(\mathcal{G}_i^M) \backslash X_6$ and $d_{\mathcal{G}^M}(y, v_i^{p1}) \ge 6$ for all $y \in V(\mathcal{G}_j^M)$. We illustrated the corresponding situation in Figure 8.6. Hence, the pairs that we have to consider are $\{x,y\}$ with $x \in X_6$ and $y \in V(\mathcal{G}_j^M)$. But for the former described pairs, we obtain $d_{\mathcal{G}^M}(y, v_j^{p1}) \le 8$ and $d_{\mathcal{G}^M}(x, v_j^{p1}) \ge 9$.

Finally, we have to show that all pairs p_e^1 and p_e^2 with $e \in E(\mathcal{G})$ remain unresolved in order to finish the proof of Lemma 8.5.4. For this reason, we consider the graph $\mathcal{G}_e^M$ for a fixed $e \in E(\mathcal{G})$. Let $\{e_i \in E(\mathcal{G}) \mid i \in [6]\}$ be the set of edges with $e_i \cap e \neq \varnothing$ and $e_i \neq e$ for all $i \in [6]$. Notice that for every $i \in [6]$, the shortest path from $v_{e_i}^{p1}$ to v_e^s contains the vertex v_e^{cf}. The same holds for all shortest paths from $v_{e_i}^{p1}$ to $v \in \{v_e^{ct}, v_e^{cs}, v_e^t\}$. Since we have $d_{\mathcal{G}^M}(v_e^{cf}, v_e^s) = d_{\mathcal{G}^M}(v_e^{cf}, v_e^t)$ and $d_{\mathcal{G}^M}(v_e^{cf}, v_e^{cs}) = d_{\mathcal{G}^M}(v_e^{cf}, v_e^{ct})$, we conclude that $d_{\mathcal{G}^M}(v_{e_i}^{p1}, v_e^s) = d_{\mathcal{G}^M}(v_{e_i}^{p1}, v_e^t)$ and $d_{\mathcal{G}^M}(v_{e_i}^{p1}, v_e^{cs}) = d_{\mathcal{G}^M}(v_{e_i}^{p1}, v_e^{ct})$ for all $i \in [6]$. With this fact in mind, we are able to prove inductively that all shortest paths from $v_{e'}^{p1}$ with $e' \neq e$ to $v \in \{v_e^s, v_e^{ct}, v_e^{cs}, v_e^t\}$ lead over the vertex v_e^{cf}. Hence, the vertices $v \in \{v_e^{p1} \mid e \in E(\mathcal{G})\}$ leave every split pair of $\mathcal{G}_e^M$ unresolved.

The remaining vertices to be analyzed are v_i^{p1} with $i \in V(\mathcal{G})$:
Notice that given a vertex v_i^{p1} with $i \in e$, we have that $d_{\mathcal{G}^M}(v_i^{p1}, v_e^s) = d_{\mathcal{G}^M}(v_i^{p1}, v_e^t) = 7$ and $d_{\mathcal{G}^M}(v_i^{p1}, v_e^{cs}) = d_{\mathcal{G}^M}(v_i^{p1}, v_e^{ct}) = 8$.

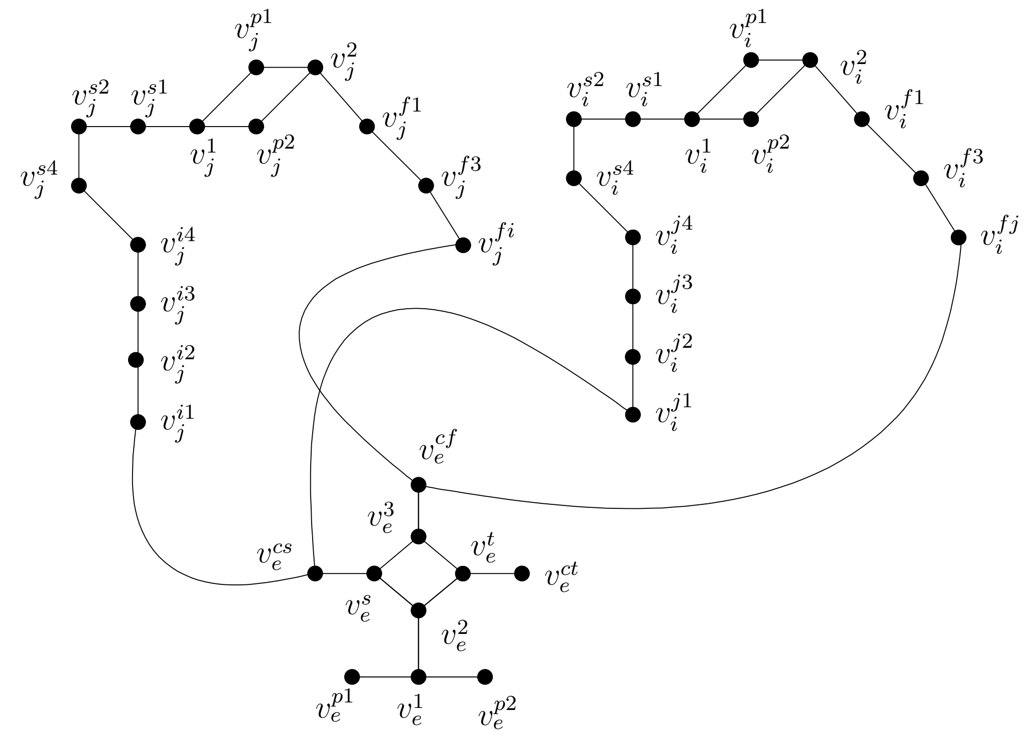


Figure 8.6: 5.Case $\big(\{x, y\}$, where $x \in V(\mathcal{G}_j^M)$, $y \in V(\mathcal{G}_i^M)$ and $i \neq j\big)$.

Therefore, let $e_1 \neq e$ be an edge in $\mathcal{G}$ and $j \in e_1 \backslash e$ a vertex in $\mathcal{G}$. Then, the shortest path from v_j^{p1} to v_e^s can be divided into the shortest path from v_j^{p1} to $v_{e_1}^{cf}$ and the shortest path from $v_{e_1}^{cf}$ to v_e^s. The same holds for shortest paths from v_j^{p1} to $v \in \{v_e^{ct}, v_e^{cs}, v_e^t\}$. From the former case, we know that $d_{\mathcal{G}^M}(v_{e_1}^{cf}, v_e^s) = d_{\mathcal{G}^M}(v_{e_1}^{cf}, v_e^t)$ and $d_{\mathcal{G}^M}(v_{e_1}^{cf}, v_e^{cs}) = d_{\mathcal{G}^M}(v_{e_1}^{cf}, v_e^{ct})$ holds. Thus, v_j^{p1} cannot resolve any split pair of $\mathcal{G}_e^M$. By a similar argument, we conclude that the same holds for all $v \in \{v_j^{p1} \mid j \in V(\mathcal{G}) \backslash e\}$ finishing the proof. ■

Thus far, we know that the set $R_m(\mathcal{G})$ resolves all pairs $p \in \binom{V(\mathcal{G}^M)}{2}$ except $p \in \{p_e^1, p_e^2 \mid e \in E(\mathcal{G})\}$. Consequently, we want to know which vertices are able to distinguish the remaining pairs. In order to resolve this question, we are going to prove the following lemma.

Lemma 8.5.5

Let $\mathcal{G}$ be a 4-regular graph and e an edge in $\mathcal{G}$. Then, the vertex $v_i^{s1} \in V(\mathcal{G}^M)$

resolves the pair p_e^2 if and only if $i \in e$ holds.

Proof of Lemma 8.5.5.
Let $\mathcal{G}$ be a 4-regular graph and $e \in E(\mathcal{G})$. We are going to analyze two cases starting with $i \in e$:
In the case $(i \in e)$, we have $d_{\mathcal{G}^M}(v_i^{s1}, v_e^s) = 8$ and $d_{\mathcal{G}^M}(v_i^{s1}, v_e^t) = 9$, which by definition means that the vertex v_i^{s1} resolves p_e^2.

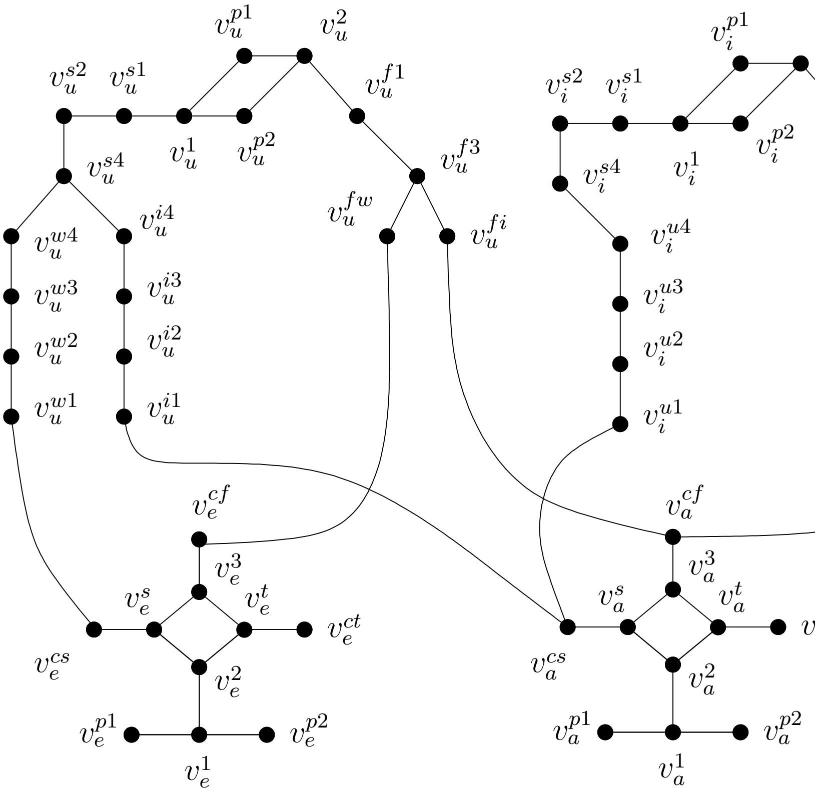


Figure 8.7: Case $(i \neq e)$ with $a = \{i, u\}$ and $e = \{u, w\}$ in Lemma 8.5.5.

Now, let us consider the case $(i \notin e)$:
We are going to analyze the situation displayed in Figure 8.7. In this situation, we have $d_{\mathcal{G}^M}(v_i^{s1}, v_e^s) = d_{\mathcal{G}^M}(v_i^{s1}, v_e^t) = 13$. Notice that a shortest path from v_i^{s1} to both vertices v_e^t and v_e^s lead over the edge $\{v_e^{cf}, v_e^3\}$ causing that the considered pair remains unresolved. More generally, a similar argumentation as in the proof of Lemma 8.5.4 leads to the fact that shortest paths

from every v_i^{s1} with $i \notin e$ to both, v_e^t and v_e^s, contain the vertex v_e^{cf}. Hence, these vertices cannot resolve p_e^2. ■

At this point, we are ready to give the proof of Lemma 8.5.3.

Proof of Lemma 8.5.3.
Let $\mathcal{G}$ be a 4-regular graph and C a vertex cover of $\mathcal{G}$. By definition, $R_m(\mathcal{G})$ is a subset of $R(C)$. Thus, by Lemma 8.5.4, we know that the only pairs which could be unresolved are p_e^1 and p_e^2 for every $e \in E(\mathcal{G})$.

On the other hand, the vertices $i \in V(\mathcal{G})$ with $v_i^{s1} \in R(C)\backslash R_m(\mathcal{G})$ form a vertex cover of the original graph $\mathcal{G}$. Therefore, Lemma 8.5.5 applies to $R(C)\backslash R_m(\mathcal{G})$ resolving all pairs p_e^2 with $e \in E(\mathcal{G})$.

According to Lemma 8.5.2, the same vertices in $R(C)\backslash R_m(\mathcal{G})$ also split all remaining pairs p_e^1 with $e \in E(\mathcal{G})$ and the proof of Lemma 8.5.3 follows. ■

8.5.4 Constructing a Vertex Cover from a Resolving Set

Given a 4-regular graph $\mathcal{G}$ and a resolving set R of $\mathcal{G}^M$, we are going to construct the corresponding vertex cover $C_\mathcal{G}(R)$ of the original graph $\mathcal{G}$ based on the resolving set R.

Let us first give the definition of the corresponding vertex cover $C_\mathcal{G}(R)$. Afterwards, we prove that $C_\mathcal{G}(R)$ is indeed a vertex cover of the graph $\mathcal{G}$.

Definition 8.5.2 (Vertex Cover $C_\mathcal{G}(R)$)
Let $\mathcal{G}$ be a 4-regular graph with $V(\mathcal{G}) = \{1,\dots,n\}$ and R a resolving set of $\mathcal{G}^M$. Given R, the vertex cover $C_\mathcal{G}(R)$ is defined as follows.

$$C_\mathcal{G}(R) = \left(\bigcup_{\substack{\{i,j\}\in E(\mathcal{G}) \\ i<j}} \{\, i \mid R \cap D(\{i,j\}) \neq \varnothing \,\} \right) \cup \left(\bigcup_{v \in V(\mathcal{G})} \{\, v \mid D(v) \cap R \neq \varnothing \,\} \right)$$

According to Lemma 8.5.1, we know that any resolving set R of $\mathcal{G}^M$ must contain at least one main vertex of every subgraph $\mathcal{G}_x^M$ with $x \in V(\mathcal{G}) \cup E(\mathcal{G})$. Let us fix a $x \in V(\mathcal{G}) \cup E(\mathcal{G})$. Then, for every $v \in V(\mathcal{G}^M)\backslash\{v_x^{p2}, v_x^{p1}\}$, we have $d_{\mathcal{G}^M}(v, v_x^{p1}) = d_{\mathcal{G}^M}(v, v_x^{p2})$. Thus, the vertices v_x^{p1} and v_x^{p2} resolve the same pairs in $\mathcal{G}^M$. Moreover, even by including both vertices, it will not change the number of resolved pairs. Consequently, we may assume without loss of generality that $R_m(\mathcal{G}) \subseteq R$ holds.

Recall from Lemma 8.5.4 that $R_m(\mathcal{G})$ resolves all $p \in \binom{V(\mathcal{G}^M)}{2}$ except p_e^1 and p_e^2 with $e \in E(\mathcal{G})$. Hence, we have to determine which vertices are able to distinguish the remaining pairs. According to Lemma 8.5.2, it suffices to identify the particular vertices resolving p_e^2 with $e \in E(\mathcal{G})$.

At this point, we provide the following lemma whose proof will be given later on in this section.

Lemma 8.5.6
Let $\mathcal{G}$ be a 4-regular graph and $\{u, w\}$ be an edge in $\mathcal{G}$. Then, $v \in V(\mathcal{G}^M)$ resolves the pair $p^2_{\{u,w\}}$ if and only if

$$v \in \left[D_{\mathcal{G}}(\{\, u,\, w\}) \cup D^{-}_{\mathcal{G}}(u) \cup D^{-}_{\mathcal{G}}(w) \cup \{\, v_u^{w1},\, v_w^{u1}\}\right].$$

Given an edge $e = \{u, w\} \in E(\mathcal{G})$, according to Lemma 8.5.6, the only vertices resolving the pair $p^2_{\{u,w\}}$ are subsets of $V(\mathcal{G}_u^M)$, $V(\mathcal{G}_w^M)$ and $V(\mathcal{G}_e^M)$. Since R is resolving for $\mathcal{G}^M$ and by definition of $C_{\mathcal{G}}(R)$, for every $e \in E(\mathcal{G})$, there exist at least one vertex $i \in \big(e \cap C_{\mathcal{G}}(R)\big)$. In summary, we obtain the following statement.

Lemma 8.5.7
Let $\mathcal{G}$ be a 4-regular graph and R a resolving set of $\mathcal{G}^M$. Given R, the set of vertices $C_{\mathcal{G}}(R)$ can be constructed in polynomial time and is a vertex cover of $\mathcal{G}$.

In order to complete the proof of Lemma 8.5.7, it remains to give the proof of Lemma 8.5.6.

Proof of Lemma 8.5.6.
Let $\mathcal{G}$ be a 4-regular graph and $e = \{w, u\} \in E(\mathcal{G})$. We are going to determine which vertices of $\mathcal{G}^M$ are able to resolve p_e^2. Let us first consider vertices $v \in V(\mathcal{G}_e^M)$: Notice that all $v \in D(\{w, u\}) = p_e^1 \cup p_e^2$ possess this property, whereas v_e^{p1}, v_e^{p2}, v_e^1, v_e^2, v_e^3 and v_e^{cf} cannot resolve p_e^2.

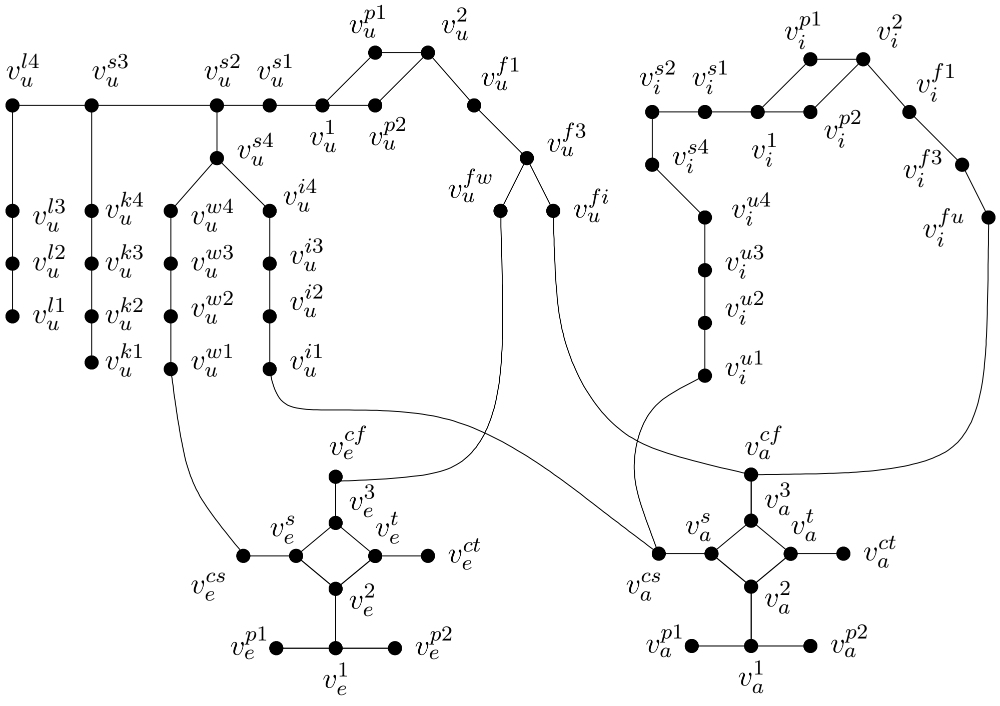


Figure 8.8: Graphs $\mathcal{G}_e$, $\mathcal{G}_u$, $\mathcal{G}_a$ with $a = \{i, u\}$ and $\{w, u\}, a \in E(\mathcal{G})$.

Hence, we are left to analyze the vertices in $V(\mathcal{G}^M) \backslash V(\mathcal{G}_e^M)$:
We are going to analyze the situation displayed in Figure 8.8. The neighborhood of u in $\mathcal{G}$ is given by $N_{\mathcal{G}}(u) = \{l, k, w, i\}$. Furthermore, we use $a = \{i, u\}$. Note that all vertices in $D^-_{\mathcal{G}}(u)$ are able to distinguish the pair p_e^2. On the other hand, for v_u^{i1}, we obtain $d_{\mathcal{G}^M}(v_e^s, v_u^{i1}) = 10 = d_{\mathcal{G}^M}(v_e^t, v_u^{i1})$. Since we have $d_{\mathcal{G}^M}(v_e^s, v) = 12$ and $d_{\mathcal{G}^M}(v_e^t, v) = 12$ for each $v \in \{v_u^{l1}, v_u^{k1}\}$, both, v_u^{l1} and v_u^{k1} cannot split p_e^2. For v_u^{w1}, we obtain $d_{\mathcal{G}^M}(v_e^s, v_u^{w1}) = 2 \neq d_{\mathcal{G}^M}(v_e^t, v_u^{w1}) = 4$. Furthermore, we see that shortest paths from

$$v \in \{v_u^1, v_u^{p1}, v_u^{p2}, v_u^2, v_u^{f1}, v_u^{f2}, v_u^{f3}, v_u^{fl}, v_u^{fk}, v_u^{fw}, v_u^{fi}\}$$

to $w \in \{v_e^t, v_e^s\}$ all contain v_e^{cf} and conclude that v cannot resolve p_e^2. By symmetry, we infer that the only vertices in $V(\mathcal{G}_w^M)$ splitting p_e^2 are $v \in D^-_{\mathcal{G}}(w)$ and v_w^{u1}.

Let us consider the vertex v_a^{cs}. Then, all shortest paths from v_a^{cs} to v_e^t as well as to v_e^s contain the vertex v_e^{cf} and in particular, we obtain $d_{\mathcal{G}^M}(v_e^s, v_a^{cs}) = 9$ and $d_{\mathcal{G}^M}(v_e^t, v_a^{cs}) = 9$.

For the remaining vertices, we argue analogously to the proof of Lemma 8.5.5 that all shortest paths to v_e^t and v_e^s contain the vertex v_e^{cf}. In other words, these vertices cannot distinguish the pair p_e^2.

In summary, the only vertices resolving p_e^2 are contained in the set

$$D_{\mathcal{G}}^{-}(u) \cup D_{\mathcal{G}}^{-}(w) \cup \left\{ v_w^{u1}, v_u^{w1} \right\} \cup D_{\mathcal{G}}(e).$$

■

At this point, we are ready to give the proof of Theorem 8.4.1.

8.5.5 Proof of Theorem 8.4.1

Given a 4-regular graph $\mathcal{G}$ as an instance of the VERTEX COVER problem, we construct in polynomial time the corresponding graph $\mathcal{G}^M$ as described in Section 8.5.2. Recall that the maximum degree of $\mathcal{G}^M$ is bounded by 3. Since $\mathcal{G}$ is connected, $\mathcal{G}^M$ inherits this property due to its construction.
(i) Let C be a vertex cover of $\mathcal{G}$. Then, we construct in polynomial time the set $R_{\mathcal{G}}(C)$ as described in Section 8.5.3. According to Lemma 8.5.3, $R_{\mathcal{G}}(C)$ is a resolving set of $\mathcal{G}^M$. Then, the metric dimension of $\mathcal{G}^M$ is at most

$$dim_M(\mathcal{G}^M) \;\le\; |R_{\mathcal{G}}(C)| \;=\; |C| + |E(\mathcal{G})| + |V(\mathcal{G})|.$$

(ii) On the other hand, given a resolving set R of $\mathcal{G}^M$, we construct in polynomial time the vertex cover $C_{\mathcal{G}}(R)$ of the original graph $\mathcal{G}$ by applying Lemma 8.5.7. From Lemma 8.5.1, we know that every resolving set of $\mathcal{G}^M$ contains at least one main vertex of every subgraph $\mathcal{G}_x^M$ with $x \in E(\mathcal{G}) \cup V(\mathcal{G})$. Moreover, for every vertex in the set

$$R \backslash \left(\bigcup_{x \in E(\mathcal{G}) \cup V(\mathcal{G})} \{ v_x^{p1}, v_x^{p1} \} \right),$$

$C_{\mathcal{G}}(R)$ contains at most one vertex of $V(\mathcal{G})$. Therefore, we conclude that

$$|C_{\mathcal{G}}(R)| \;\le\; |R| - \sum_{x \in E(\mathcal{G}) \cup V(\mathcal{G})} \left| \{ v_x^{p1}, v_x^{p1} \} \cap R \right| \;\le\; |R| - |E(\mathcal{G})| - |V(\mathcal{G})|$$

and the proof of Theorem 8.4.1 follows. ■

8.6 Improved Approximation Lower Bound

In this section, we give an improved approximation preserving reduction from the VERTEX COVER problem in 4-regular graphs to the METRIC DIMENSION problem restricted to graphs with maximum degree 3. Given a 4-regular graph $\mathcal{G}$, we construct the corresponding graph $\mathcal{G}^I$ with its subgraphs $\mathcal{G}^I_e$ and $\mathcal{G}^I_v$ having improved properties and less vertices compared to the subgraphs $\mathcal{G}^M_v$ and $\mathcal{G}^M_e$ in the previous section. In particular, a resolving set S of the graph $\mathcal{G}^I$ with $|S| = dim_M(\mathcal{G}^I)$ contains only one vertex of every subgraph $\mathcal{G}^I_v$ with $v \in V(\mathcal{G})$ in contrast to the previous construction, in which we needed one vertex per each subgraph $\mathcal{G}^M_x$ with $x \in E(\mathcal{G}) \cup V(\mathcal{G})$. Moreover, we will give the proof of Theorem 8.4.3.

8.6.1 Constructing $\mathcal{G}^I$ from a 4-regular Graph $\mathcal{G}$

Given a 4-regular graph $\mathcal{G}$ as an instance of the VERTEX COVER problem, we are going to construct the corresponding graph $\mathcal{G}^I$ with maximum degree 3 as an instance of the METRIC DIMENSION problem. In addition, we prove some important properties of the corresponding graph $\mathcal{G}^I$. Let us start with the description of the graph $\mathcal{G}^I$.

Construction of $\mathcal{G}^I$

Let $\mathcal{G}$ be a 4-regular graph. For convenience, we split the neighbourhood of every vertex $v \in V(\mathcal{G})$ into two sets of equal size, denoted by $N^1_{\mathcal{G}}(v)$ and $N^2_{\mathcal{G}}(v)$. Given $\mathcal{G}$, the corresponding graph $\mathcal{G}^I$ consists of the subgraphs $\mathcal{G}^I_v$ and $\mathcal{G}^I_e$ for every vertex $v \in V(\mathcal{G})$ and every edge $e \in E(\mathcal{G})$. We refer to them as the *vertex* and the *edge* graph, respectively. In the following, we are going to specify the subgraphs $\mathcal{G}^I_x$ with $x \in V(\mathcal{G}) \cup E(\mathcal{G})$ beginning with $\mathcal{G}^I_v$.

Vertex Graph $\mathcal{G}^I_v$

Let $i \in V(\mathcal{G})$ be a vertex and $N_{\mathcal{G}}(i) = N^1_{\mathcal{G}}(i) \cup N^2_{\mathcal{G}}(i)$ be the set of its neighbours in $\mathcal{G}$, where $N^1_{\mathcal{G}}(i) = \{y, x\}$ and $N^2_{\mathcal{G}}(i) = \{u, w\}$. The corresponding vertex graph $\mathcal{G}_i^I$ is displayed in Figure 8.9. We refer to v_i^{p1} and v_i^{p2} as the *main vertices of* $\mathcal{G}_i^I$. Furthermore, we introduce the set of vertices $R_{\mathcal{G}}^1$ defined by $R_{\mathcal{G}}^1 = \{v_i^{p1} \mid i \in V(\mathcal{G})\}$.

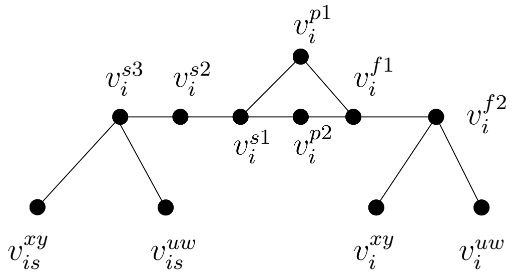


Figure 8.9: Subgraph $\mathcal{G}_i^I$ with $i \in V(\mathcal{G})$, $N_{\mathcal{G}}^1(i) = \{x, y\}$ and $N_{\mathcal{G}}^2(i) = \{u, w\}$.

Edge Graph $\mathcal{G}_e^I$

Let $e \in E(\mathcal{G})$ be an edge in $\mathcal{G}$ joining the vertices u and w. Then, $\mathcal{G}_e^I$ is defined as displayed in Figure 8.10. Furthermore, we introduce the *split pair* of $\mathcal{G}_e^I$ given by $p_e = \{v_e^s, v_e^t\}$. For a vertex $i \in V(\mathcal{G})$ with $N_{\mathcal{G}}^1(i) = \{v, w\}$ and $N_{\mathcal{G}}^2(i) = \{x, y\}$, we define the set $D(i)$ as follows.

$$D(i) = \left\{ v_i^{s2},\ v_i^{s1},\ v_i^{s3},\ v_{is}^{vw},\ v_{is}^{xy},\ v_{\{i,x\}}^{il},\ v_{\{i,y\}}^{il},\ v_{\{i,w\}}^{il},\ v_{\{i,v\}}^{il} \;\middle|\; l \in [2] \right\}$$

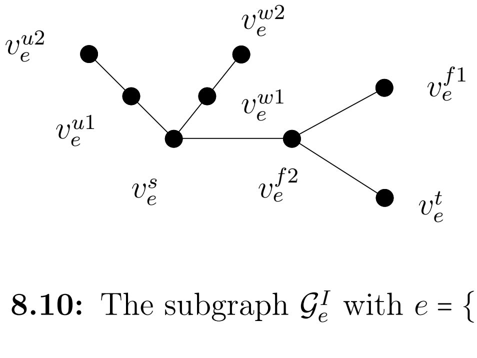


Figure 8.10: The subgraph $\mathcal{G}_e^I$ with $e = \{u, w\}$.

As mentioned, the disjoint union of the vertices in $V(\mathcal{G}_x^I)$ with $x \in V(\mathcal{G}) \cup$

$E(\mathcal{G})$ defines the vertex set of $\mathcal{G}^I$ meaning

$$V(\mathcal{G}^I) = \bigcup_{x \in V(\mathcal{G}) \cup E(\mathcal{G})} V(\mathcal{G}^I_x).$$

We now describe how the subgraphs are connected in order to form $\mathcal{G}^I$.
Let $\{u,w\} \in E(\mathcal{G})$ be an edge in $\mathcal{G}$. In addition, we let $N^1_{\mathcal{G}}(u) = \{y,w\}$ and $N^1_{\mathcal{G}}(w) = \{z,u\}$. Then, we connect the subgraphs $\mathcal{G}^I_u$ and $\mathcal{G}^I_w$ both with $\mathcal{G}^I_{\{u,w\}}$ by adding the edges $\{v^{yw}_{us}, v^{u2}_{\{u,w\}}\}$, $\{v^{yw}_{u}, v^{f1}_{\{u,w\}}\}$, $\{v^{zu}_{ws}, v^{w2}_{\{u,w\}}\}$ and $\{v^{zu}_{w}, v^{f1}_{\{u,w\}}\}$ to $E(\mathcal{G}^I)$. By implementing the former construction for every edge $e \in E(\mathcal{G})$, we obtain in this way the graph $\mathcal{G}^I$ with maximum vertex degree 3.

Properties of the Corresponding Graph $\mathcal{G}^I$

First of all, we are going to prove that for each $v \in V(\mathcal{G})$, every resolving set of $\mathcal{G}^I$ includes at least one main vertex of $\mathcal{G}^I_v$.

Lemma 8.6.1
Let $\mathcal{G}$ be a 4-regular graph and R a resolving set of $\mathcal{G}^I$. Then, for every $v \in V(\mathcal{G})$, there is at least one main vertex of $\mathcal{G}^I_v$ included in R.

Proof of Lemma 8.6.1.
Let $\mathcal{G}$ be a 4-regular graph, $i \in V(\mathcal{G})$ and R a resolving set of $\mathcal{G}^I$. We note that for every vertex $v \in V(\mathcal{G}^I) \setminus \{v^{p1}_i, v^{p2}_i\}$, we have $d_{\mathcal{G}^I}(v^{p1}_i, v) = d_{\mathcal{G}^I}(v^{p2}_i, v)$. Thus, in order to resolve the pair $\{v^{p1}_i, v^{p2}_i\}$, at least one vertex $v \in \{v^{p1}_i, v^{p2}_i\}$ is contained in R. ■

8.6.2 Constructing a resolving set from a vertex cover

Let $\mathcal{G}$ be a 4-regular graph and $\mathcal{G}^I$ the corresponding graph constructed as described in the previous section. Given a vertex cover C of $\mathcal{G}$, we are going to construct the resolving set $R^I_{\mathcal{G}}(C)$ of $\mathcal{G}^I$ based on C.

The crucial difference between the sets $R^I_{\mathcal{G}}(C)$ and $R_{\mathcal{G}}(C)$ from Definition 8.5.1 is that $R^I_{\mathcal{G}}(C)$ includes only vertices corresponding to elements contained in $V(\mathcal{G})$ and in C. It entails a decreased size of the resolving set implying an improved approximation preserving reduction. Let us first define the set $R^I_{\mathcal{G}}(C)$ given a vertex cover C of $\mathcal{G}$. Afterwards, we are going to prove that $R^I_{\mathcal{G}}(C)$ is indeed resolving for $\mathcal{G}^I$.

Definition 8.6.1 (Resolving Set $R^I_{\mathcal{G}}(C)$)
Let $\mathcal{G}$ be a 4-regular graph, $\mathcal{G}^I$ the corresponding graph constructed as described in Section 8.6.1 and C a vertex cover of $\mathcal{G}$. Then, the resolving set $R^I_{\mathcal{G}}(C)$ is defined as follows.

$$R^I_{\mathcal{G}}(C) = \{v^{p1}_v, v^{s1}_u \mid v \in V(\mathcal{G}), u \in C\}$$

The remaining part of this section is devoted to prove that the set $R^I_{\mathcal{G}}(C)$ is resolving for $\mathcal{G}^I$. Let us formulate the precise statement.

Lemma 8.6.2
Let $\mathcal{G}$ be a 4-regular graph, $\mathcal{G}^I$ the corresponding graph constructed as described in Section 8.6.1 and C a vertex cover of $\mathcal{G}$. Then, the set $R^I_{\mathcal{G}}(C)$ is resolving for $\mathcal{G}^I$.

Let us describe the structure of the proof of Lemma 8.6.2.
First of all, we show that the set $R^1_{\mathcal{G}}$, defined by $R^1_{\mathcal{G}} = \{v^{p1}_w \mid w \in V(\mathcal{G})\}$, resolves every pair of vertices of $\mathcal{G}^I$ except the split pair $p_e = \{v^s_e, v^t_e\}$ for each $e \in E(\mathcal{G})$.

Then, we prove that v^{s1}_u with $u \in e$ resolves p_e. Since $R^1_{\mathcal{G}}$ is a subset of $R^I_{\mathcal{G}}(C)$ and C is a vertex cover of $\mathcal{G}$, we conclude that $R^I_{\mathcal{G}}(C)$ is a resolving set of $\mathcal{G}^I$. Let us proceed to prove the following statement.

Lemma 8.6.3
Let $\mathcal{G}$ be a 4-regular graph. The set $R^1_{\mathcal{G}}$ resolves every pair $p \in \binom{V(\mathcal{G}^I)}{2}$ except $p \in \{\, p_e \mid e \in E(\mathcal{G}) \,\}$.

Proof of Lemma 8.6.3.
In the first part of the proof of Lemma 8.6.3, we show that any pair in

$$\binom{V(\mathcal{G}^I)}{2} \setminus \left(\bigcup_{e \in E(\mathcal{G})} \{p_e\} \right)$$

can be distinguished by some vertex $w \in R^1_{\mathcal{G}}$. We proceed by case analysis starting with $p \in \binom{V(\mathcal{G}_e^I)}{2}\backslash\{p_e\}$ for a fixed $e \in E(\mathcal{G})$:

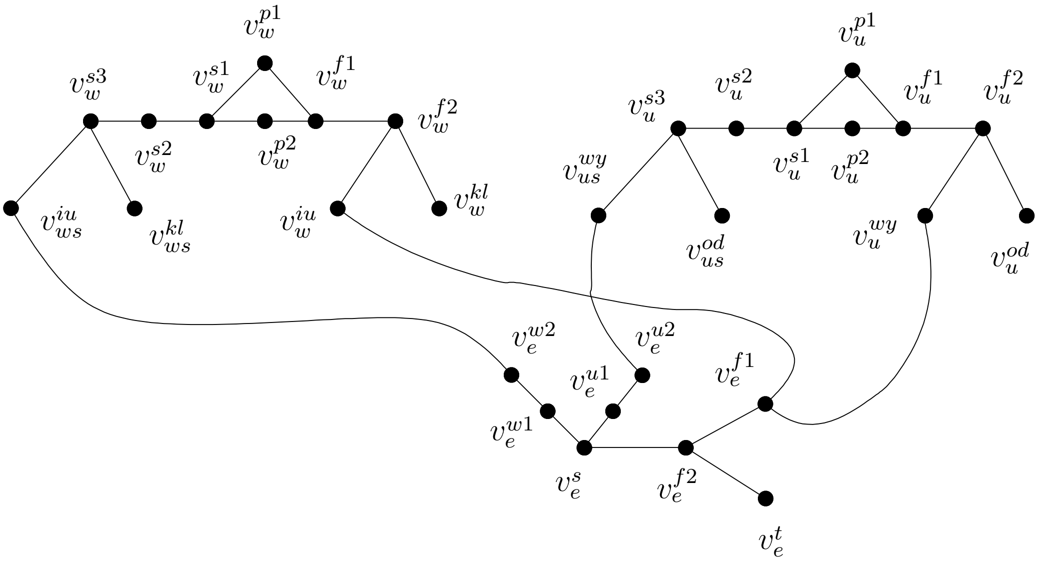


Figure 8.11: Subgraphs $\mathcal{G}_e^I$, $\mathcal{G}_u^I$ and $\mathcal{G}_w^I$ with $e = \{u, w\}$.

1.Case $\left(p \in \binom{V(\mathcal{G}_e^I)}{2}\backslash\{p_e\}\right.$ with $\left.e \in E(\mathcal{G})\right)$:

Let $\{u, w\}$ be an edge in $\mathcal{G}$ and $\mathcal{G}^I_{\{u,w\}}$ its edge graph. Furthermore, we let $N^1_{\mathcal{G}}(w) = \{i, u\}$, $N^2_{\mathcal{G}}(w) = \{k, l\}$, $N^1_{\mathcal{G}}(u) = \{w, y\}$ and $N^2_{\mathcal{G}}(u) = \{o, d\}$. This situation is displayed in Figure 8.11. Then, we consider the following distance classes induced by $\{v_u^{p1}\}$ in $\mathcal{G}_e^I$.

$$\{v_e^{f1}\}, \quad \{v_e^{f2}, v_e^{u2}\}, \quad \{v_e^s, v_e^t, v_e^{u1}\}, \quad \{v_e^{w2}\}, \quad \{v_e^{w1}\}$$

By taking v_w^{p1} into account, we can resolve all distance classes except $\{v_e^t, v_e^s\}$.

2.Case $\left(p \in \binom{V(\mathcal{G}_i^I)}{2}\right.$ with $\left.i \in V(\mathcal{G})\right)$:

Let $i \in V(\mathcal{G})$ be a vertex and $N_{\mathcal{G}}(i) = N^1_{\mathcal{G}}(i) \cup N^2_{\mathcal{G}}(i)$ its neighbours in $\mathcal{G}$, where $N^1_{\mathcal{G}}(i) = \{x, y\}$ and $N^2_{\mathcal{G}}(i) = \{u, w\}$. Then, the following distance classes are induced by $\{v_i^{p1}\}$ in $\mathcal{G}_i^I$.

$$\{v_i^{p1}\}, \quad \{v_i^{f1}, v_i^{s1}\}, \quad \{v_i^{p2}, v_i^{f2}, v_i^{s2}\}, \quad \{v_i^{xy}, v_i^{uw}, v_i^{s3}\}, \quad \{v_{is}^{xy}, v_{is}^{uw}\}$$

We note that the remaining unresolved distance classes can be resolved by v_u^{p1}.

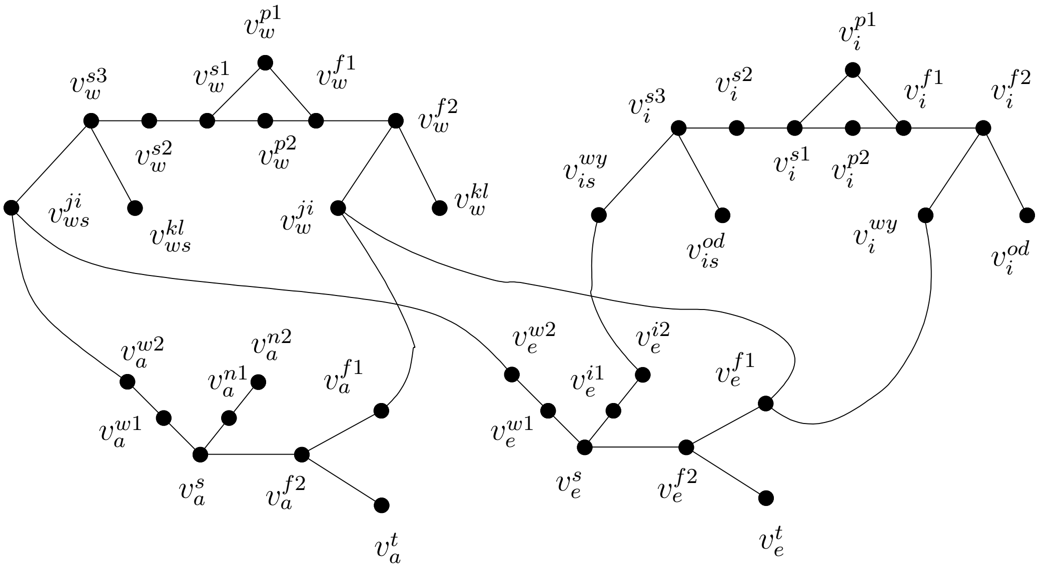


Figure 8.12: Case $(e \cap a \neq \emptyset)$ with $e = \{i, w\}$ and $w \in e \cap a$.

3.Case $\big(\{x, z\}$, where $x \in V(\mathcal{G}_e)$, $z \in V(\mathcal{G}_a)$ and $\{e, a\} \subseteq E(\mathcal{G})\big)$:
Let e and a be edges in $\mathcal{G}$. We begin with the case $(e \cap a \neq \emptyset)$. Then, we define $e = \{i, w\}$ and $w \in e \cap a$. The corresponding situation is displayed in Figure 8.12. The unresolved distance classes induced by $\{v_i^{p1}\}$ in $\mathcal{G}_e^I$ and $\mathcal{G}_a^I$ are

$$\{v_e^{f2}, v_e^{i2}\},\ \{v_e^t, v_e^s, v_a^{f1}, v_e^{i1}\},\ \{v_a^{f2}, v_e^{w1}\},\ \{v_a^s, v_a^t, v_e^{w2}\},$$

$$\{v_a^{w1}, v_a^{n1}\} \text{ and } \{v_a^{w2}, v_a^{n2}\}.$$

By considering v_w^{p1}, we are left with the unresolved distance classes p_e and p_a.

In the case $(e \cap a = \emptyset)$, we define $e = \{u, w\}$ and $a = \{c, y\}$. Then, there is a vertex $j \in \{c, y\}$ such that for all $x \in V(\mathcal{G}_e^I)$, we have $d_{\mathcal{G}^I}(v_j^{p1}, x) \geq 8$. On the other hand, for all $z \in V(\mathcal{G}_a^I)$, we obtain $d_{\mathcal{G}^I}(v_j^{p1}, z) \leq 8$. Note that there could be only one remaining unresolved pair $\{x, z\}$ with $x \in V(\mathcal{G}_e^I)$ and $z \in V(\mathcal{G}_a^I)$. By taking v_u^{p1} into account, this last pair can be resolved by v_u^{p1}.

4.Case $\big(\{x,y\}$, where $x \in V(\mathcal{G}^I_e)$, $y \in V(\mathcal{G}^I_i)$, $i \in V(\mathcal{G})$ and $e \in E(\mathcal{G})\big)$:
In case of $i \notin e$, we obtain $d_{\mathcal{G}^I}(v_i^{p1}, x) \leq 4$ and $d_{\mathcal{G}^I}(v_i^{p1}, y) \geq 6$ for all $x \in V(\mathcal{G}^I_i)$ and $y \in V(\mathcal{G}^I_e)$.

Therefore, we may assume that we have $e = \{i,k\}$. Let $N_{\mathcal{G}}(i)$ be partitioned by $N^1_{\mathcal{G}}(i) = \{k,l\}$ and $N^2_{\mathcal{G}}(i) = \{u,w\}$. Again, we analyze the distance classes induced by v_i^{p1}. The remaining unresolved distance classes are $\{v_{is}^{kl}, v_e^{f1}\}$ and $\{v_{is}^{uw}, v_e^{f1}\}$. But, these pairs can be resolved by v_k^{p1}.

5.Case $\big(\{x,y\}$, where $x \in V(\mathcal{G}^I_j)$, $y \in V(\mathcal{G}^I_i)$ and $\{i,j\} \subseteq V(\mathcal{G})\big)$:
Let i and j be different vertices in $\mathcal{G}$. Then, we note that for all $x \in V(\mathcal{G}^I_i)$ and $y \in V(\mathcal{G}^I_j)$, we have that $d_{\mathcal{G}^I}(x, v_i^{p1}) \leq 4$ and $d_{\mathcal{G}^I}(y, v_i^{p1}) \geq 5$.

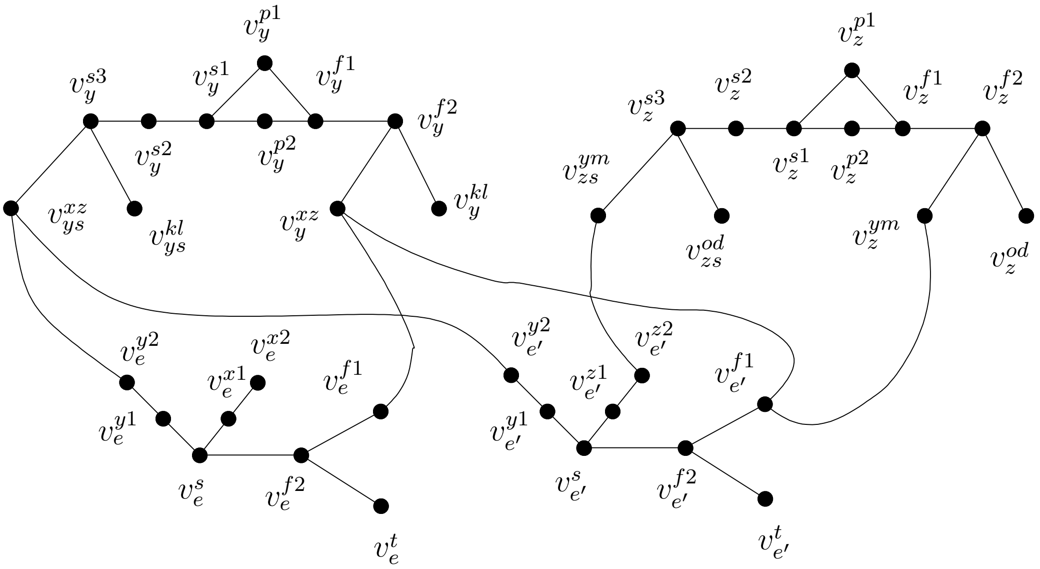


Figure 8.13: Case $(e \cap e' \neq \varnothing)$ with $e = \{y,x\}$ and $e' = \{z,y\}$.

Hence, we can distinguish all considered pairs finishing the first part of the proof. In the second part, we will prove that all pairs p_e with $e \in E(\mathcal{G})$ remain unresolved by $R^1_{\mathcal{G}}$. For this reason, we consider the graph $\mathcal{G}^I_e$ for a fixed $e = \{x,y\} \in E(\mathcal{G})$. We notice that $d_{\mathcal{G}^I}(v_e^t, v_x^{p1}) = d_{\mathcal{G}^I}(v_e^s, v_x^{p1}) = 6$ holds. Thus,

v_x^{p1} cannot resolve p_e. By symmetry, the same is true for v_y^{p1}.

Let $e' = \{y,z\} \in E(\mathcal{G})$ be an edge in $\mathcal{G}$ sharing a vertex with e. Furthermore, let the neighborhood of z be defined by $N_{\mathcal{G}}^1(z) \cup N_{\mathcal{G}}^2(z)$, where $N_{\mathcal{G}}^1(z) = \{y,m\}$ and $N_{\mathcal{G}}^2(z) = \{o,d\}$. Then, we conclude that the shortest path $\mathcal{P}_{t-e}^{p1-z}$ from v_z^{p1} to v_e^t in $\mathcal{G}^I$ is given by

$$\mathcal{P}_{t-e}^{p1-z} = v_z^{p1} - v_z^{f1} - v_z^{f2} - v_z^{ym} - v_{e'}^{f1} - v_y^{xz} - v_e^{f1} - v_e^{f2} - v_e^t.$$

The situation is displayed in Figure 8.13. We keep in mind that $\mathcal{P}_{t-e}^{p1-z}$ contains the vertex v_e^{f1}. The same holds for the shortest path from v_z^{p1} to v_e^s. More generally, all the shortest paths from a vertex v_y^{p1} with $y \in V(\mathcal{G})$ to v_e^s or v_e^t possess this property, which can be proved inductively. Consequently, we conclude that this pair cannot be resolved by any v_y^{p1} with $y \in V(\mathcal{G})$. In summary, the set $R_{\mathcal{G}}^1$ leaves all pairs p_e with $e \in E(\mathcal{G})$ unresolved. ■

Thus far, we know that the set $R_{\mathcal{G}}^1$ resolves all $p \in \binom{V(\mathcal{G}^I)}{2}$ except pairs $p \in \{p_e \mid e \in E(\mathcal{G})\}$. Therefore, we want to know which vertices are able to distinguish the remaining pairs. A step towards this goal is the following lemma.

Lemma 8.6.4

Let $\mathcal{G}$ be a 4-regular graph and $\mathcal{G}^I$ its corresponding graph. For a fixed $e \in E(\mathcal{G})$, the vertex v_i^{s1} with $i \in V(\mathcal{G})$ resolves the pair p_e if and only if $i \in e$ holds.

Proof of Lemma 8.6.4.

Let $\mathcal{G}$ be a 4-regular graph, $i \in V(\mathcal{G})$ a vertex and $e \in E(\mathcal{G})$ an edge in $\mathcal{G}$. We begin with the case $i \in e$. Then, we obtain $d_{\mathcal{G}^I}(v_i^{s1}, v_e^s) = 6$ and $d_{\mathcal{G}^I}(v_i^{s1}, v_e^t) = 7$ implying that the pair p_e is indeed resolved by v_i^{s1}.

Next, we consider the case $(i \notin e)$. Let us analyze the situation displayed in Figure 8.14. We let $e = \{y,a\}$ and $e' = \{i,y\}$. In this situation, we have $d_{\mathcal{G}^I}(v_i^{s1}, v_e^s) = d_{\mathcal{G}^I}(v_i^{s1}, v_e^t) = 9$. Notice that shortest paths from v_i^{s1} to v_e^t and v_e^s lead over the edge $\{v_e^{f1}, v_e^{f2}\}$ causing that the considered pairs remain unresolved. Similarly to the proof of Lemma 8.6.3, it can be proved that shortest paths form every v_i^{s1} with $i \notin e$ to v_e^t and v_e^s contain the edge $\{v_e^{f2}, v_e^{f1}\}$. Accordingly, vertices v_i^{s1} with $i \notin e$ leave the pair p_e unresolved. ■

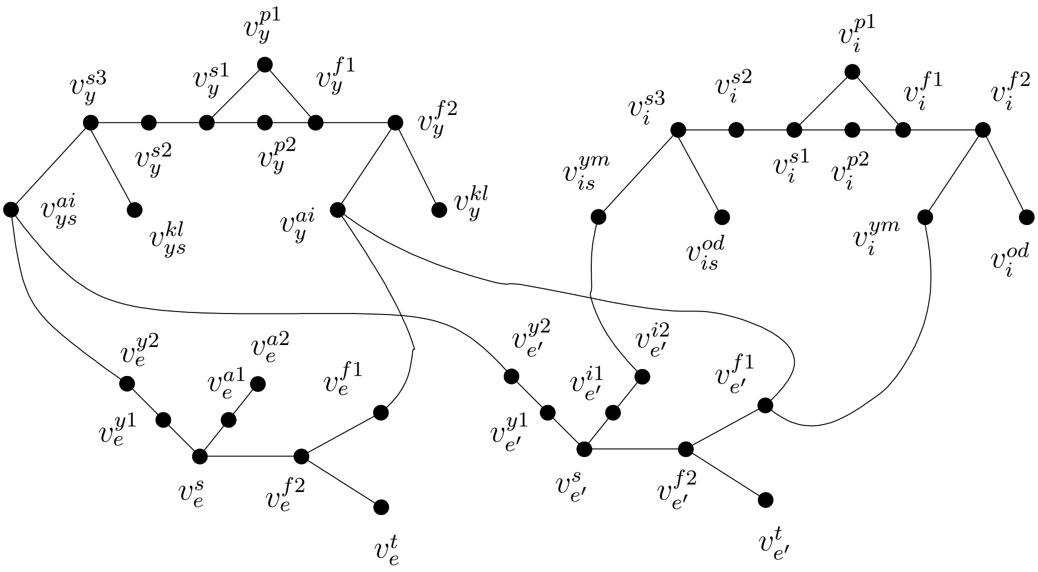


Figure 8.14: Case ($i \notin e$), where $e = \{y, a\}$ and $e' = \{i, y\}$.

At this point, we are ready to give the proof of Lemma 8.6.2.

Proof of Lemma 8.6.2.
Let $\mathcal{G}$ be a 4-regular graph and C a vertex cover of $\mathcal{G}$. Recall that by definition of $R^I_{\mathcal{G}}(C)$, we have $R^1_{\mathcal{G}} \subseteq R^I_{\mathcal{G}}(C)$. From Lemma 8.6.3, we know that $R^1_{\mathcal{G}}$ resolves all pairs of vertices in $\mathcal{G}^I$ except p_e with $e \in E(\mathcal{G})$.

On the other hand, the corresponding vertices $i \in V(\mathcal{G})$ with $v_i^{s1} \in R^I_{\mathcal{G}}(C) \backslash R^1_{\mathcal{G}}$ form a vertex cover of the original graph $\mathcal{G}$. Therefore, Lemma 8.6.4 applies to $R^I_{\mathcal{G}}(C) \backslash R^1_{\mathcal{G}}$ resolving the remaining pairs p_e with $e \in E(\mathcal{G})$. ∎

8.6.3 Constructing the Corresponding Vertex Cover

Given a 4-regular graph $\mathcal{G}$, its corresponding graph $\mathcal{G}^I$ and a resolving set R of $\mathcal{G}^I$, we construct in polynomial time a vertex cover $C^I_{\mathcal{G}}(R)$ of the original graph $\mathcal{G}$ based on R. Let us first define the corresponding vertex cover $C^I_{\mathcal{G}}(R)$ given a resolving set R of $\mathcal{G}^I$.

Definition 8.6.2 (Vertex Cover $C^I_{\mathcal{G}}(R)$)
Let $\mathcal{G}$ be a 4-regular graph with $V(\mathcal{G}) = \{1,\dots,n\}$, $\mathcal{G}^I$ its corresponding graph and R a resolving set of $\mathcal{G}^I$. The set of vertices $C^I_{\mathcal{G}}(R)$ is defined as follows.

$$C^I_{\mathcal{G}}(R) = \left(\bigcup_{\substack{\{i,j\}\in E(\mathcal{G})\\ i<j}} \left\{ i \,\middle|\, R \cap p_{\{i,j\}} \neq \varnothing \right\} \right) \cup \left(\bigcup_{k \in V(\mathcal{G})} \{ k \mid D(k) \cap R \neq \varnothing \} \right)$$

In the remainder of this section, we will prove that the former defined set $C^I_{\mathcal{G}}(R)$ is indeed a vertex cover of the original graph $\mathcal{G}$. More precisely, we are going to prove the following statement.

Lemma 8.6.5
Let $\mathcal{G}$ be a 4-regular graph, $\mathcal{G}^I$ its corresponding graph and R a resolving set of $\mathcal{G}^I$. Given R, the set $C^I_{\mathcal{G}}(R)$ is a vertex cover of $\mathcal{G}$ and is constructible in polynomial time.

Given a resolving set R of $\mathcal{G}^I$, it is straightforward to construct the set $C^I_{\mathcal{G}}(R)$ in polynomial time. Hence, it remains to be proved that $C^I_{\mathcal{G}}(R)$ is a vertex cover of $\mathcal{G}$. According to Lemma 8.6.1, a resolving set R of $\mathcal{G}^I$ contains at least one main vertex of the every graph $\mathcal{G}^I_v$ with $v \in V(\mathcal{G})$. Moreover, we may assume that every resolving set contains $R^1_{\mathcal{G}}$ as a subset. This can be deduced from the fact that for every $x \in V(\mathcal{G}^I)\backslash\{v_i^{p1}, v_i^{p2}\}$ with $i \in V(\mathcal{G})$, we have $d_{\mathcal{G}^I}(v_i^{p1}, x) = d_{\mathcal{G}^I}(v_i^{p2}, x)$. As a matter of fact, even when including both vertices, v_i^{p1} and v_i^{p2}, we resolve the same number of pairs in $\mathcal{G}^I$. By Lemma 8.6.3, the remaining unresolved pairs $p \in \binom{V(\mathcal{G}^I)}{2}$ are exactly p_e with $e \in E(\mathcal{G})$. It gives rise to the question which vertices in $\mathcal{G}^I$ are able to distinguish the remaining pairs. The following lemma resolves this question.

Lemma 8.6.6
Let $\mathcal{G}$ be a 4-regular graph and $\mathcal{G}^I$ its corresponding graph. Given an edge $\{u,w\} \in E(\mathcal{G})$, then, the vertex $v \in V(\mathcal{G}^I)$ resolves the pair $p_{\{u,w\}}$ if and only if $v \in \left(p_{\{u,w\}} \cup D(u) \cup D(w)\right)$.

Proof of Lemma 8.6.6.
Let $\mathcal{G}$ be a 4-regular graph and $\{u,w\} \in E(\mathcal{G})$ an edge in $\mathcal{G}$. Furthermore, let

the neighborhood of u in $\mathcal{G}$ be given by $N^1_{\mathcal{G}}(u) = \{w, x\}$ and $N^2_{\mathcal{G}}(u) = \{z, y\}$. Then, from Lemma 8.6.3, we know that v^{p1}_u, v^{p2}_u, v^{p1}_w and v^{p2}_w cannot resolve p_e. On the other hand, we see that the vertices $v \in p_e$ are able to distinguish p_e. According to Lemma 8.6.4, v^{s1}_u and v^{s1}_w, both split the pair p_e. It implies that every vertex along the shortest path $\mathcal{P}^{s1-u}_{s-e}$ from v^{s1}_u to v^s_e possesses this property, where $\mathcal{P}^{s1-u}_{s-e}$ is defined as follows.

$$\mathcal{P}^{s1-u}_{s-e} = v^{s1}_u - v^{s2}_u - v^{s3}_u - v^{wx}_{us} - v^{u2}_e - v^{u1}_e - v^s_e$$

The situation is displayed in Figure 8.15. For convenience, we use the abbreviations $a = \{u, x\}$ and $b = \{u, y\}$. Then, we note that $d_{\mathcal{G}^I}(v^{u1}_a, v^s_e) = 5$ and $d_{\mathcal{G}^I}(v^{u1}_a, v^t_e) = 7$ holds. Accordingly, all vertices along the shortest path $\mathcal{P}^{u1-a}_{s-e}$ from v^{u1}_a to v^s_e resolve the pair p_e, where

$$\mathcal{P}^{u1-a}_{s-e} = v^{u1}_a - v^{u2}_a - v^{wx}_{us} - v^{u2}_e - v^{u1}_e - v^s_e.$$

By contrast, the vertex v^s_a cannot distinguish the pair p_e since we have $d_{\mathcal{G}^I}(v^{u1}_a, v^s_e) = 6$ and $d_{\mathcal{G}^I}(v^{u1}_a, v^t_e) = 6$. For the vertex v^{y1}_b, we observe that $d_{\mathcal{G}^I}(v^{u1}_b, v^s_e) = 7$ and $d_{\mathcal{G}^I}(v^{u1}_b, v^t_e) = 9$ implying that every vertex on the shortest path $\mathcal{P}^{u1-b}_{s-e}$ from v^{u1}_b to v^s_e splits the pair p_e, where

$$\mathcal{P}^{u1-b}_{s-e} = v^{u1}_b - v^{u2}_b - v^{zy}_{us} - v^{s3}_u - v^{wx}_{us} - v^{u2}_e - v^{u1}_e - v^s_e.$$

On the other hand, we have $d_{\mathcal{G}^I}(v^s_b, v^s_e) = 8$ and $d_{\mathcal{G}^I}(v^s_b, v^t_e) = 8$. A similar situation holds for the vertex v^{u1}_c with $c = \{u, z\}$ yielding $d_{\mathcal{G}^I}(v^{u1}_c, v^s_e) = 7$ and $d_{\mathcal{G}^I}(v^{u1}_c, v^t_e) = 9$. Therefore, we conclude that vertices $v \in D(u)$ all resolve the pair p_e. By symmetry, the same holds for vertices $v \in D(w)$. On the other hand, for the vertices

$$v \in \left\{ v^{f1}_u, v^{f2}_u, v^{f1}_w, v^{f2}_w, v^{wx}_u, v^{zy}_u, v^{pq}_w, v^{ut}_w \;\middle|\; p, q \in N^1_{\mathcal{G}}(w),\ u, t \in N^2_{\mathcal{G}}(w) \right\},$$

we obtain $d_{\mathcal{G}^I}(v, v^s_e) = d_{\mathcal{G}^I}(v, v^t_e)$. Analogously, for all vertices w having the property that a shortest path $\mathcal{P}^w_{s-e}$ from w to v^s_e in $\mathcal{G}^I$ is including v^{wx}_u, we obtain $d_{\mathcal{G}^I}(w, v^s_e) = d_{\mathcal{G}^I}(w, v^t_e)$.

By induction, it can be proved that all vertices

$$v \in V(\mathcal{G}^I) \backslash \big(p_e \cup D(u) \cup D(w)\big)$$

possess this property. In summary, we conclude that the only vertices resolving the pair p_e are $v \in (p_e \cup D(u) \cup D(w))$. ■

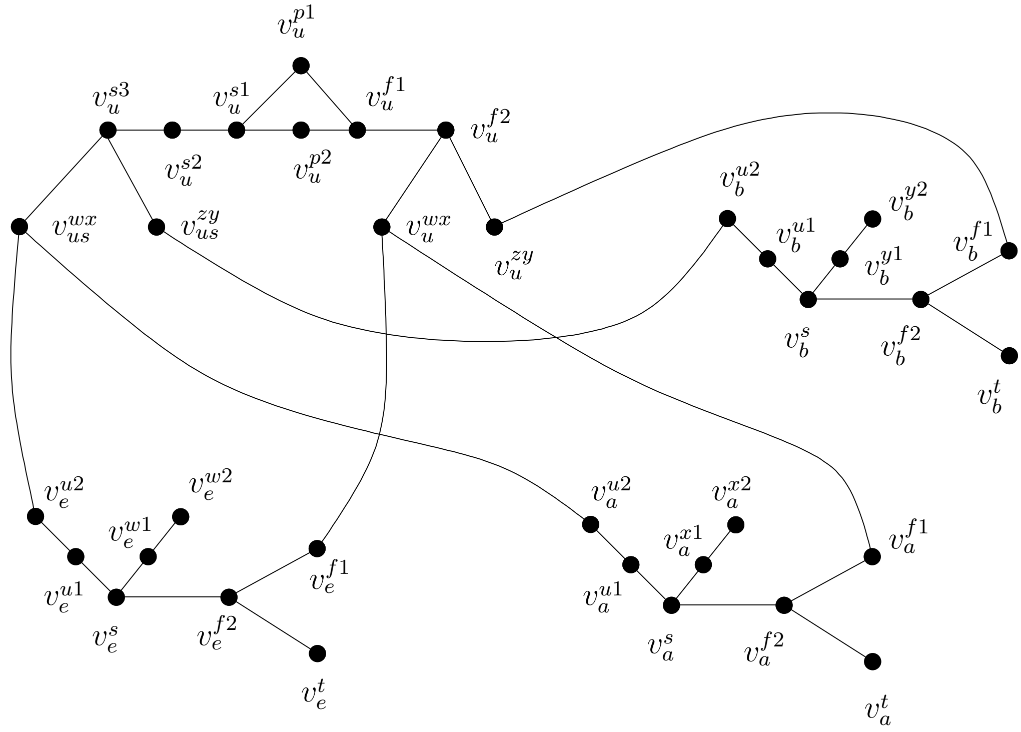


Figure 8.15: Subgraphs $\mathcal{G}_u^I$, $\mathcal{G}_e^I$, $\mathcal{G}_a^I$ and $\mathcal{G}_b^I$, where $a = \{u, x\}$ and $b = \{u, y\}$.

After having resolved which vertices can distinguish the pairs p_e with $e \in E(\mathcal{G})$, we are ready to give the proof of Lemma 8.6.5.

Proof of Lemma 8.6.5.

Let $\mathcal{G}$ be a 4-regular graph and R a resolving set of $\mathcal{G}^I$. From Lemma 8.6.1, we know that any resolving set R of $\mathcal{G}^I$ includes at least one main vertex of every subgraph $\mathcal{G}_v^I$ with $v \in V(\mathcal{G})$. According to Lemma 8.6.3, the vertices in $R_{\mathcal{G}}^1$ resolve every pair in $\mathcal{G}^I$ except p_e with $e \in E(\mathcal{G})$. By applying Lemma 8.6.6, for every $e = \{u, w\} \in E(\mathcal{G})$, there exists a vertex $i_r \in \left(C_{\mathcal{G}}^I(R) \cap e\right)$ corresponding to the vertex $r \in R \cap \left(p_e \cup D(u) \cup D(w)\right)$ and the proof of Lemma 8.6.5 follows. ■

8.6.4 Proof of Theorem 8.4.3

Given a 4-regular graph $\mathcal{G}$, we construct in polynomial time the corresponding graph $\mathcal{G}^I$ as described in Section 8.6.1. Note that the maximum degree of $\mathcal{G}^I$ is 3.

(i) Let C be a vertex cover of the original graph $\mathcal{G}$. Given C, we construct in polynomial time the corresponding resolving set $R^I_{\mathcal{G}}(C)$. According to Lemma 8.6.2, $R^I_{\mathcal{G}}(C)$ is indeed resolving for $\mathcal{G}^I$ and the metric dimension of $\mathcal{G}^I$ can be bounded from above as follows.

$$dim_M(\mathcal{G}^I) \le |R^I_{\mathcal{G}}(C)| = |C| + |R^1_{\mathcal{G}}| = |C| + |V(\mathcal{G})|$$

(ii) On the other hand, given a resolving set R of $\mathcal{G}^I$, we are able to construct in polynomial time the set $C^I_{\mathcal{G}}(R)$. By Lemma 8.6.5, we know that $C^I_{\mathcal{G}}(R)$ is a vertex cover of $\mathcal{G}$. Furthermore, we know that every resolving set of $\mathcal{G}^I$ contains at least one main vertex of every subgraph $\mathcal{G}^I_x$ with $x \in V(\mathcal{G})$. Furthermore, for each vertex $w \in \left(R\backslash\{v^{p1}_v, v^{p2}_v \mid v \in V(\mathcal{G})\}\right)$, $C^I_{\mathcal{G}}(R)$ contains at most one vertex $u_w \in V(\mathcal{G})$. We conclude that the size of the set $C^I_{\mathcal{G}}(R)$ can be bounded from above by

$$|C^I_{\mathcal{G}}(R)| \le |R| - \sum_{i \in V(\mathcal{G})} |R \cap \left\{v^{p1}_i, v^{p2}_i\right\}| \le |R| - |V(\mathcal{G})|$$

and the proof of Theorem 8.4.3 follows. ■

8.7 Bibliographic Notes

Some parts of the material presented in this chapter are based on the paper [HSV12]. In particular, the proof of Theorem 8.4.1 appeared in [HSV12].

CHAPTER 9

The Shortest Superstring and Related Problems

In this chapter, we study the approximation hardness of the SHORTEST SUPERSTRING, the MAXIMUM COMPRESSION and the MAXIMUM ASYMMETRIC TRAVELING SALESPERSON (MAX-ATSP) problem. We introduce a new reduction method that produces strongly restricted instances of the SHORTEST SUPERSTRING problem, in which the maximal orbit size is 8 (with no character appearing more than 8 times) and all given strings having length exactly 4. Based on this reduction method, we are able to improve the best up to now known approximation lower bound for the SHORTEST SUPERSTRING problem and the MAXIMUM COMPRESSION problem by an order of magnitude. In particular, our first reduction implies an inapproximability threshold of 345/344 for the SHORTEST SUPERSTRING problem and 207/206 for the MAXIMUM COMPRESSION problem. By designing more efficient gadgets, we improve the corresponding bounds to 333/332 and 204/203, respectively. The results imply also an improved approximation lower bound for the MAX-ATSP problem.

9.1 Introduction

The SHORTEST SUPERSTRING problem is the following problem: given a finite set S of strings and we would like to construct their shortest superstring, which is the shortest possible string such that every string in S is a proper substring of it.

The task of computing a shortest common superstring appears in a wide variety of application related to computational biology (see. e.g. [L88] and [L90]). Intuitively, short superstrings preserve important biological structure and are good models of the original DNA sequence. In context of computational biology, DNA sequencing is the important task of determining the sequence of nucleotides in a molecule of DNA. The DNA can be seen as a double-stranded sequence of four types of nucleotides represented by the alphabet $\{A, C, G, T\}$. Identifying those strings for different molecules is an important step towards understanding the biological functions of the molecules. However, with current laboratory methods, it is quite impossible to extract a long molecule directly as a whole. In fact, biochemists split millions of identical molecules into pieces each typically containing at most 500 nucleotides. Then, from sometimes millions of these fragments, one has to compute the superstring representing the whole molecule.

Interested in computational aspects of this problem, Maier and Storer [MS77] proved that the decision version of the SHORTEST SUPERSTRING problem is **NP**-complete. However, there are many applications that involve relatively simple classes of strings. Motivated by those applications, many authors have investigated whether the SHORTEST SUPERSTRING problem becomes polynomial time solvable under various restrictions to the set of instances. Gallant, Maier and Storer [GMS80] proved that this problem in the exact setting is still **NP**-complete for strings of length three and polynomial time solvable for strings of length two. On the other hand, Timkovskii [T90] studied the SHORTEST SUPERSTRING problem under restrictions to the orbit size of the letters in the alphabet. The orbit size of a letter is the number of its occurrences in the strings of $\mathcal{S}$. Timkovskii proved that this problem restricted to instances with maximal orbit size two is polynomial time solvable. He raised the question about the status of the problem with maximal orbit size k for any constant $k \geq 3$. Middendorf [M94] proved that the restricted problem, in which instances $\mathcal{S}$ have maximal orbit size six and the length of all strings in $\mathcal{S}$ equals four, is **NP**-hard. Furthermore, we mention that the SHORTEST SUPERSTRING problem remains **NP**-hard for strongly restricted instances, such as

- all strings have length three and the maximal orbit size is eight [M94],
- the size of the alphabet of the instance is exactly two [GMS80], and
- all strings are of the form $10p10q$, where $p, q \in \mathbb{N}$ [M98].

In order to cope with the exact computation intractability, approximation algorithms were designed to deal with this problem. In 1990, Li [L90] gave a polynomial time approximation algorithm for this problem with an approximation ratio $\mathrm{O}(\log n)$. The first polynomial time approximation algorithm for the problem with a constant approximation ratio was given by

Blum et al. [BJL+94] achieving an approximation ratio 3. This factor was improved in a series of papers yielding approximation ratios of 2.89 by Teng and Yao [TT93], 2.84 by Czumaj, Gasieniec, Piotrow and Rytter [CGPR97], 2.80 by Kosaraju, Park and Stein [KPS94], 2.75 by Armen and Stein [AS95], 2.67 by Armen and Stein [AS96]; 2.60 by Breslauer, Jiang and Jiang [BJJ97] and 2.5 by Sweedyk [S99]. Kaplan, Lewenstein, Shafrir and Sviridenko [KLSS05] and Paluch, Elbassioni and van Zuylen [PEZ12] designed an efficient approximation algorithm for the MAX-ATSP problem (see Definition 9.4.3) with approximation ratio 1.5. By using a black-box reduction due to Breslauer, Jiang, and Jiang [BJJ97], the approximation algorithms mentioned above for the MAX-ATSP problem yields an approximation ratio 2.5 for the SHORTEST SUPERSTRING problem. Both, especially the approximation algorithm given by Paluch, Elbassioni and van Zuylen [PEZ12], are significantly simpler than the approximation algorithm for the SHORTEST SUPERSTRING problem due to Sweedyk [S99]. More recently, Mucha [M13] broke the long-standing bound of 2.5 and developed an efficient approximation algorithm for the SHORTEST SUPERSTRING problem with approximation ratio 2.48. For the special case, when all input strings have length exactly γ, the problem can be approximated within $(\gamma^2+\gamma-4)/(4\gamma-6)$ improving on the general bound of 2.48 for all $\gamma \in \{3,\dots,7\}$ (cf. [GKM13]).

On the lower bound side, Blum et al. [BJL+94] proved that the SHORTEST SUPERSTRING problem is **APX**-hard. However, the constructed reduction produces instances with arbitrarily large alphabets. Ott [O99] gave the first explicit approximation hardness result and proved that the problem is **APX**-hard even if the size of the alphabet is two. Moreover, he established an inapproximability threshold of 17246/17245 for instances over a binary alphabet. In 2005, Vassilevska [V05] improved this bound to 1217/1216 by using a natural construction. The constructed instances of the SHORTEST SUPERSTRING problem have maximal orbit size 20 and the length of the strings is exactly 4.

In this chapter, we prove that even instances of the SHORTEST SUPERSTRING problem with maximal orbit size 8 and all strings having length 4 are **NP**-hard to approximate to within any factor less than 333/332.

Maximum Compression Problem

Given a collection of strings $\mathcal{S}$, we want to find a superstring for $\mathcal{S}$ with maximum compression, which is the difference between the sum of the lengths of the given strings and the length of the superstring.

In the exact setting, we note that an optimal solution to the SHORTEST SUPERSTRING problem is an optimal solution to this problem, but the approximate solutions can differ significantly in the sense of approximation ratio. The MAXIMUM COMPRESSION problem arises in various data compression problems (cf. [SS82], [S88b] and [MJ75]). The decision version of the problem is **NP**-complete [MS77]. Tarhio and Ukkonen [TU88] and Turner [T89] gave efficient approximation algorithms for the problem with approximation ratio 2. The best known approximation upper bound is 1.5 [KLSS05] by reducing the MAXIMUM COMPRESSION problem to the MAX-ATSP problem defined below.

On the approximation lower bound side, Blum et al. [BJL+94] proved that the MAXIMUM COMPRESSION problem is **APX**-hard. The first explicit approximation lower bound was given by Ott [O99], who proved that it is **NP**-hard to approximate the problem to within any factor less than 11217/11216. This hardness result was improved by Vassilevska [V05] implying an approximation lower bound of 1072/1071.

In this chapter, we prove that it is **NP**-hard to approximate the MAXIMUM COMPRESSION problem to within any constant factor less than 204/203.

Maximum Asymmetric Traveling Salesperson (MAX-ATSP) Problem

Given a complete directed graph $\mathcal{G}$ and a weight function w assigning each edge of $\mathcal{G}$ a non-negative weight, the task is to find a closed tour of maximum total weight visiting every vertex of $\mathcal{G}$ exactly once.

Since the MAX-ATSP problem is **APX**-hard [PY93], there is little hope for polynomial time approximation algorithms with arbitrary good precision. Besides being an interesting problem on its own, we are interested in designing good approximation algorithms for the MAX-ATSP problem since it implies good approximations for a number of related problems. For example, it was proved by Breslauer, Jiang and Jiang [BJJ97] that an approximation algorithm for the MAX-ATSP problem with approximation ratio α entails an approximation algorithm for the SHORTEST SUPERSTRING problem with approximation ratio $\big(7/2 - 3/2 \cdot (1/\alpha)\big)$. In addition, Kaplan, Lewenstein, Shafrir and Sviridenko [KLSS05] proved that it implies an approximation algorithm for the MAXIMUM COMPRESSION problem with approximation ratio α.

The first efficient approximation algorithm for the MAX-ATSP problem with guaranteed approximation performance is due to Fisher, Nemhauser, and Wolsey [FNW79] and achieves an approximation ratio 2. This approximation upper bound was improved in a series of papers yielding approximation ratios of 1.66 by Kosaraju, Park, and Stein [KPS94], 1.63 by Bläser [B02], 1.60 by Lewenstein and Sviridenko [LS03] and 1.50 by Kaplan, Lewenstein, Shafrir and Sviridenko [KLSS05]. More recently, Paluch, Elbassioni and van Zuylen [PEZ12] gave a simpler approximation algorithm compared to the one due to Kaplan, Lewenstein, Shafrir and Sviridenko [KLSS05] with the same approximation ratio.

On the approximation hardness side, Engebretsen [E03] proved that it is **NP**-hard to approximate the MAX-ATSP problem within any constant approximation ratio better than 2804/2803. The approximation lower bound for this problem was improved to 320/319 by Engebretsen and Karpinski [EK06].

In this chapter, we prove that approximating the MAX-ATSP problem to within any constant factor less than 204/203 is **NP**-hard.

9.2 The Proof Methods and Summary of Results

The results of this chapter depend on a new reduction method for proving approximation hardness of the SHORTEST SUPERSTRING and related problems. This reduction method defines for each problem so called parity gadgets that on the one hand, are simulating the variables of a well-suited bounded occurrence CSP and on the other hand, transmit the parity information to the gadgets that are simulating the linear equation mod 2 with exactly three variables of the CSP. The crucial point of the reduction is that we make essential use of the underlying structure of the constraints in the CSP, which are induced by a 3-regular amplifier graph. In Chapter 10, we will extend this method to TSP problems and we believe that it could be a more widely useful method for improving the approximation lower bounds of other problems.

In Table 9.1, we summarize our approximation lower bounds as compared with previous inapproximability results.

Problem	Previously known	Our Results
SHORTEST SUPERSTRING problem	1217/1216 [V05]	333/332
MAXIMUM COMPRESSION problem	1072/1071 [V05]	204/203
MAX – ATSP problem	320/319 [EK06]	204/203

Table 9.1: Comparison of known explicit lower bounds and our results.

9.3 Outline of this Chapter

This chapter is organized as follows. In Section 9.5, we formulate our main results. In Section 9.6, we describe the properties of our first reduction and its implications. In Section 9.7, we give a high-level view of the reduction. In Section 9.8, we define the special instances of the SHORTEST SUPERSTRING problem. In Section 9.9, we construct a superstring from an assignment to a given instance of the MAX-HYBRID-LIN2 problem (cf. Definition 4.9.1). In Section 9.10, we give the description the corresponding assignment given a superstring and prove our first result. In Section 9.11, we construct an improved reduction and give the proof of our main results.

9.4 Preliminaries

As we will deal with strings, we are going to introduce some basic terminology used in this context. In addition, we introduce some notation related to tours in directed graphs.

Basic String Terminology and Problem Specification

Given two strings $v = v_1 \dots v_n$ and $w = w_1 \dots w_m$ over a finite alphabet Σ, v is said to be a *substring* of w if $m \geq n$ and there exists a $j \in \{0, \dots, n - m\}$ such that for all $i \in [n]$, we have $v_i = w_{j+i}$. A string v is said to be a *prefix* of w if v is a substring of w and $w = v_1 \dots v_n \, w_{n+1} \dots w_m$. We say that v is a *suffix* of w if there is a prefix w' of w such that $w = w'_1 \dots w'_{m-n} v_1 \dots v_n$.

Furthermore, we define the *overlap* of w and v, denoted $ov(w, v)$, as the longest suffix of w that is also a prefix of v. Also, we define the prefix of v with respect to w, denoted $pref(v, w)$, as the string v' such that $v = v' \; ov(v, w)$. w is said to be a *superstring* of v if v is a substring of w. Given a finite set of strings $S = \{s_1, \dots, s_n\}$, a string s is a superstring for S if s is a superstring of every s_i in S.

According to Definition 4.1.1, the SHORTEST SUPERSTRING problem is defined as follows.

Definition 9.4.1 (SHORTEST SUPERSTRING problem)
Instances: *A set of strings* S
Solutions: *A superstring* s *for* S
Task: *Minimize* $|s|$

Given a superstring s for a collection of strings S, the *compression* of s with respect to S, denoted *comp*(S, s), is defined as follows.

$$comp(S, s) = \sum_{s_i \in S} |s_i| - |s|.$$

By means of the notion of compression, we are going to specify the MAXIMUM COMPRESSION problem.

Definition 9.4.2 (MAXIMUM COMPRESSION problem)

Instances:	*A set of strings S*
Solutions:	*A superstring s for S*
Task:	*Maximize $comp(S, s)$*

Given a finite set of strings S over a finite alphabet, we introduce the notion of the *maximal orbit size* of S being the maximum number of occurrences of a letter in S.

MAX-ATSP Problem

Given a complete directed graph $\mathcal{G}$, a *tour* σ in $\mathcal{G}$ is formally a subset of $V(\mathcal{G}) \times V(\mathcal{G})$ inducing a directed cycle in $\mathcal{G}$ that visits each vertex in $\mathcal{G}$ only once. Given a weight function $w : V(\mathcal{G}) \times V(\mathcal{G}) \to \mathbb{Q}_{\geq 0}$ and a tour σ in $\mathcal{G}$, the *length* or *cost* of σ in $(\mathcal{G}, w)$ is defined as follows.

$$\text{length of } \sigma \text{ in } (\mathcal{G}, w) \;=\; \sum_{a \in \sigma} w(a)$$

According to Definition 4.1.1, the MAX-ATSP problem is specified as follows.

Definition 9.4.3 (MAX-ATSP problem)

Instances:	*A complete directed graph $\mathcal{G}$ and* *a weight function $w : A(\mathcal{G}) \to \mathbb{Q}_{\geq 0}$*
Solutions:	*A tour σ in $\mathcal{G}$*
Task:	*Maximize the length of σ in $(\mathcal{G}, w)$*

In order to relate the MAXIMUM COMPRESSION problem to the MAX-ATSP problem, we are going to introduce the overlap graph for a given collection of strings.

Definition 9.4.4 (Overlap Graph)
Given a finite set of strings $S = \{s_1, \ldots, s_n\}$ over a finite alphabet Σ such that no s_i is a substring of a s_j for all $i \neq j$. We define the overlap graph $\mathcal{G}_S^{ov}$ with weight function $w_S^{ov} : A(\mathcal{G}_S^{ov}) \to \mathbb{N}_0$ as follows.

$$V(\mathcal{G}_S^{ov}) = S, \;\; A(\mathcal{G}_S^{ov}) = S \times S \backslash \{(s_i, s_i) \mid i \in [n]\} \;\; \textit{and} \;\; w_S^{ov}(s_i, s_j) = |ov(s_i, s_j)|$$

Given a collection of strings $S = \{s_1, \ldots, s_n\}$ as an instance of the MAXIMUM COMPRESSION problem, we define an instance $\mathcal{G}_S$ of the MAX-ATSP problem by adding a special vertex s_{n+1} to the overlap graph $\mathcal{G}_S^{ov}$ and constructing the weight function $w_S : V(\mathcal{G}_S) \times V(\mathcal{G}_S) \to \mathbb{N}_{\geq 0}$ with

$$w_S(s_i, s_j) = \begin{cases} w_S^{ov}(s_i, s_j) & \text{if } \{i, j\} \subseteq [n], \\ 0 & \text{else.} \end{cases}$$

Then, the optimal compression of a superstring for S is identical to the maximum length of a tour in $\mathcal{G}_S$. This simple approximation preserving reduction was used by Kaplan, Lewenstein, Shafrir and Sviridenko [KLSS05] in order to obtain an improved approximation algorithm for the MAXIMUM COMPRESSION problem.

Lemma 9.4.1 ([KLSS05])
A polynomial time approximation algorithm for the MAX-ATSP *problem with approximation ratio α implies a polynomial time approximation algorithm for the* MAXIMUM COMPRESSION *problem with approximation ratio α.*

9.5 Our Contribution

We now formulate the results of our first approach.

Theorem 9.5.1
Given an instance $\mathcal{L}$ of the MAX-HYBRID-LIN2 *problem with n wheels, m_2 equations with two variables and m_3 equations with exactly three variables with the properties described in Theorem 4.9.1, we construct in polynomial time an instance $\mathcal{S}_{\mathcal{L}}$ of the* SHORTEST SUPERSTRING *problem and* MAXIMUM COMPRESSION *problem with the following properties:*

(*i*) *If there exists an assignment ϕ to the variables of $\mathcal{L}$ which leaves u equations in $\mathcal{L}$ unsatisfied, then, there exist a superstring for $\mathcal{S}_{\mathcal{L}}$ with length at most $5m_2 + 16m_3 + 7n + u$.*

(*ii*) *From every superstring s for $S_{\mathcal{L}}$ with length $|s| = 5m_2 + 16m_3 + u + 7n$,* *we can construct in polynomial time an assignment* ψ_s *to the variables of* $\mathscr{L}$ *that leaves at most* u *equations in* $\mathscr{L}$ *unsatisfied.*

(iii) *If there exists an assignment* ϕ *to the variables of* $\mathscr{L}$ *which leaves* u *equations unsatisfied, then, there exist a superstring with compression at least* $3m_2 + 12m_3 - u + 5n$.

(iv) *From every superstring* s *for* $S_{\mathscr{L}}$ *with compression* $3m_2 + 12m_3 - u + 5n$, *we can construct in polynomial time an assignment* ψ_s *to the variables of* $\mathscr{L}$ *that leaves at most* u *equations in* $\mathscr{L}$ *unsatisfied.*

(v) *The maximal orbit size of the instance* $\mathcal{S}_{\mathscr{L}}$ *is* 8 *and the length of each string in* $\mathcal{S}_{\mathscr{L}}$ *is* 4.

The former theorem can be used to derive an explicit approximation lower bound for the SHORTEST SUPERSTRING problem by reducing instances of the MAX-HYBRID-LIN2 problem of the form described in Theorem 4.9.1 to the SHORTEST SUPERSTRING problem.

Corollary 9.5.1
It is **NP**-*hard to approximate the* SHORTEST SUPERSTRING *problem to within any constant approximation ratio less than* 333/332.

Proof of Corollary 9.5.1.
For a given $\varepsilon > 0$, we choose constants $k \in \mathbb{N}$ and $\delta \in (0, \frac{1}{2})$ such that the following holds.

$$\frac{333 - \delta}{332 + \delta + \dfrac{42}{k}} \geq \frac{333}{332} - \varepsilon$$

Given an instance $\mathscr{L}_3$ of the MAX-E3LIN2 problem, we generate k copies of $\mathscr{L}_3$ and produce an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem. Then, we construct the corresponding instance $\mathcal{S}_{\mathscr{L}}$ of the SHORTEST SUPERSTRING problem with the properties described in Theorem 9.5.1. We conclude according to Theorem 4.9.1 that there exist a superstring for $\mathcal{S}_{\mathscr{L}}$ with length at most

$$5 \cdot 60\nu k + 16 \cdot 2\nu k + \delta\nu k + 7n \leq \left(332 + \delta + \frac{7n}{k\nu}\right)\nu k \leq \left(332 + \delta + \frac{7 \cdot 6}{k}\right)\nu k$$

or the length of a superstring for $\mathcal{S}_{\mathscr{L}}$ is bounded from below by

$$5 \cdot 60\nu k + 16 \cdot 2\nu k + (1-\delta)\nu k + 7n \;\geq\; (332 + (1-\delta))\nu k \;\geq\; (333-\delta)\nu k.$$

From Theorem 4.9.1, we know that the two cases above are **NP**-hard to distinguish. Hence, for every $\varepsilon > 0$, it is **NP**-hard to find a solution to the SHORTEST SUPERSTRING problem with an approximation ratio

$$\frac{333-\delta}{332+\delta+\dfrac{42}{k}} \;\geq\; \frac{333}{332} - \varepsilon$$

and the proof of Corollary 9.5.1 follows. ■

Analogously, Theorem 9.5.1 can be used to derive an approximation lower bound for the MAXIMUM COMPRESSION problem.

Corollary 9.5.2
It is **NP***-hard to approximate the* MAXIMUM COMPRESSION *problem to within any constant approximation ratio less than* 204/203*.*

By applying Lemma 9.4.1, we obtain the following hardness result for the MAX-ATSP problem.

Corollary 9.5.3
It is **NP***-hard to approximate the* MAX-ATSP *problem to within any constant approximation ratio less than* 204/203*.*

9.6 The First Reduction

In this section, we present the proof of a slightly weaker result by using a more intuitive approach. In particular, it uses strings with length 6 simulating equations with exactly three variables. In section 9.11, we will introduce smaller gadgets for equations with exactly three variables implying the claimed inapproximability results.

Let us state the properties of our first approach.

Theorem 9.6.1

Given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 *problem with n wheels, m_2 equations with two variables and m_3 equations with exactly three variables with the properties described in Theorem 4.9.1, we construct in polynomial time an instance $\mathcal{S}_{\mathscr{L}}$ of the* SHORTEST SUPERSTRING *problem and* MAXIMUM COMPRESSION *problem with the following properties:*

(*i*) *If there exists an assignment ϕ to the variables of $\mathscr{L}$ which leaves u equations in $\mathscr{L}$ unsatisfied, then, there exist a superstring s_ϕ for $\mathcal{S}_{\mathscr{L}}$ with length at most $5m_2 + 22m_3 + 7n + u$.*

(*ii*) *From every superstring s for $S_{\mathscr{L}}$ with length $|s| = 5m_2 + 22m_3 + u + 7n$, we can construct in polynomial time an assignment ψ_s to the variables of $\mathscr{L}$ that leaves at most u equations in $\mathscr{L}$ unsatisfied.*

(*iii*) *If there exists an assignment ϕ to the variables of $\mathscr{L}$ which leaves u equations unsatisfied, then, there exist a superstring for $\mathcal{S}_{\mathscr{L}}$ with compression at least $3m_2 + 14m_3 - u + 5n$.*

(*iv*) *From every superstring s for $S_{\mathscr{L}}$ with compression $3m_2 + 14m_3 - u + 5n$, we can construct in polynomial time an assignment ψ_s to the variables of $\mathscr{L}$ that leaves at most u equations in $\mathscr{L}$ unsatisfied.*

(*v*) *The maximal orbit size of the instance $\mathcal{S}_{\mathscr{L}}$ is eight and the length of a string in $\mathcal{S}_{\mathscr{L}}$ is bounded by six.*

Combining Theorem 4.9.1 with Theorem 9.6.1, we obtain the following explicit lower bound for the SHORTEST SUPERSTRING problem.

Corollary 9.6.1

It is **NP**-*hard to approximate the* SHORTEST SUPERSTRING *problem to within any constant approximation ratio less than* $345/344$.

Before we proceed to the proof of Theorem 9.6.1, we describe the reduction from a high-level view and try to build some intuition.

9.7 The High-Level View of the Reduction

In order to build some intuition, let us first give the high-level view of the reduction. Given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem (cf. Definition 4.9.1), we want to transform $\mathscr{L}$ into an instance of the SHORTEST SUPERSTRING problem. Fortunately, the special structure of the linear equations in the MAX-HYBRID-LIN2 problem is particularly well-suited for our reduction, since a part of the equations with two variables form a cycle and every variable occurs exactly three times. For every equation $\ell_{i+1} \equiv x_i \oplus x_{i+1} = 0$ included in this cycle, we introduce a set $S(\ell_{i+1})$ containing two strings, which can be aligned advantageously in two natural ways. If those fragments, corresponding to two successively following equations $x_{i-1} \oplus x_i = 0$ and $x_i \oplus x_{i+1} = 0$, use the same natural alignment, we are able to overlap them by one letter. From a high level view, we can construct an associated superstring for each wheel in $\mathscr{L}$, which contains the aligned strings.

We will define for every equation ℓ in $\mathscr{L}$ an associated set of strings $S(\ell)$ and the corresponding natural alignments. The instance $\mathcal{S}_{\mathscr{L}}$ of the SHORTEST SUPERSTRING problem is defined as the union of all sets $S(\ell)$. Due to the construction of the sets $S(\ell)$, there is a particular way to interpret an alignment of the strings in $S(\ell)$ included in the resulting superstring as an assignment to the variables in the instance of the MAX-HYBRID-LIN2 problem. The major challenge in the proof of correctness is to prove that every superstring for $\mathcal{S}_{\mathscr{L}}$ can be interpreted as an assignment to the variables of $\mathscr{L}$ with the property that the number of satisfied equations is connected to the length of the superstring.

9.8 Description of the Instance $\mathcal{S}_{\mathscr{L}}$ given $\mathscr{L}$

Given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem, we are going to construct the corresponding instance $\mathcal{S}_{\mathscr{L}}$ of the SHORTEST SUPERSTRING problem. Furthermore, we introduce some useful notation.

For every equation ℓ in $\mathscr{L}$, we define a set $S(\ell)$ of corresponding strings. The corresponding instance $\mathcal{S}_{\mathscr{L}}$ of the SHORTEST SUPERSTRING problem is given by $\mathcal{S}_{\mathscr{L}} = \bigcup_{\ell \text{ in } \mathscr{L}} S(\ell)$. The strings in the set $S(\ell)$ differ by the type of equation ℓ in $\mathscr{L}$. In particular, we distinguish four types of equations contained in $\mathscr{L}$.

- Cycle equations
- Matching equations
- Wheel border equations
- Equations with exactly three variables

Let us begin with the description of the strings corresponding to wheel border equations.

Strings Corresponding to Wheel Border Equations

Given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem, a wheel $\mathcal{W}_x$ in $\mathscr{L}$ and its wheel border equation $\ell_1^x \equiv x_1 \oplus x_n = 0$, we introduce six associated strings, that are all included in the set $S(\ell_1^x)$. Due to the construction of the wheel $\mathcal{W}_x$, the variable x_n is a contact variable. This means that x_n appears in an equation ℓ_j^3 with exactly three variables. The strings in the set $S(\ell_1^x)$ differ by the type of equation ℓ_j^3. We begin with the case $\ell_j^3 \equiv x_n \oplus y \oplus z = 0$.

The string $L_x C_x^l$ is used as the initial part of the superstring corresponding to this wheel, whereas $C_x^r R_x$ is used as the end part. Furthermore, we introduce strings that represent an assignment that sets either the variable x_1 to 0 or the variable x_n to 1. The corresponding two strings are

$$C_x^l x_1^{m0} x_n^{l1} C_x^r \qquad \text{and} \qquad x_n^{l1} C_x^r C_x^l x_1^{m0}.$$

Finally, we define the last two strings of the set $S(\ell_1^x)$ given by

$$C_x^l x_1^{r1} x_n^{m0} C_x^r \qquad \text{and} \qquad x_n^{m0} C_x^r C_x^l x_1^{r1}$$

having a similar interpretation. Both pairs of strings can be overlapped by two letters. Those natural alignments have a crucial influence during the process of constructing a superstring. For this reason, we introduce a notion for this alignments. By the 0-*alignment* of the strings in $S(\ell_1^x)$, we refer to the following alignment of the four strings. In the following, $(\downarrow)$ will denote the overlapping of the strings.

$$\begin{array}{ccc} C_x^l x_1^{m0} x_n^{l1} C_x^r \quad \text{and} \quad x_n^{l1} C_x^r C_x^l x_1^{m0} & & C_x^l x_1^{r1} x_n^{m0} C_x^r \quad \text{and} \quad x_n^{m0} C_x^r C_x^l x_1^{r1} \\ \downarrow & & \downarrow \\ C_x^l x_1^{m0} x_n^{l1} C_x^r C_x^l x_1^{m0} & \text{and} & x_n^{m0} C_x^r C_x^l x_1^{r1} x_n^{m0} C_x^r \end{array}$$

On the other hand, we the define the 1-*alignment* of the strings in $S(\ell_1^x)$ as follows.

$$\begin{array}{ccc} C_x^l x_1^{m0} x_n^{l1} C_x^r \quad \text{and} \quad x_n^{l1} C_x^r C_x^l x_1^{m0} & & C_x^l x_1^{r1} x_n^{m0} C_x^r \quad \text{and} \quad x_n^{m0} C_x^r C_x^l x_1^{r1}. \\ \downarrow & & \downarrow \\ x_n^{l1} C_x^r C_x^l x_1^{m0} x_n^{l1} C_x^r & \text{and} & C_x^l x_1^{r1} x_n^{m0} C_x^r C_x^l x_1^{r1} \end{array}$$

Both ways to join the four strings are called 0/1-*alignments*.

After having described how the strings corresponding to $S(\ell_1^x)$ in case of $\ell_j^3 \equiv x_n \oplus y \oplus z = 0$ are defined, we are going to deal with the case $\ell_j^3 \equiv x_n \oplus y \oplus z = 1$. As before, we use $L_x C_x^l$ as the initial part of the superstring corresponding to this wheel, whereas $C_x^r R_x$ is used as the end part. Furthermore, we define the remaining four strings contained in $S(\ell_1^x)$ as follows.

$$C_x^l x_1^{m0} x_n^{m1} C_x^r \qquad x_n^{m1} C_x^r C_x^l x_1^{m0}$$

and

$$C_x^l x_1^{r1} x_n^{l0} C_x^r \qquad x_n^{l0} C_x^r C_x^l x_1^{r1}$$

Both pairs of strings can be overlapped by two letters. We introduce a notation for these alignments.

$$\begin{array}{ccc}
C_x^l x_1^{m0} x_n^{m1} C_x^r \quad \text{and} \quad x_n^{m1} C_x^r C_x^l x_1^{m0} & & C_x^l x_1^{r1} x_n^{l0} C_x^r \quad \text{and} \quad x_n^{l0} C_x^r C_x^l x_1^{r1}. \\
\downarrow & & \downarrow \\
C_x^l x_1^{m0} x_n^{m1} C_x^r C_x^l x_1^{m0} & \text{and} & x_n^{l0} C_x^r C_x^l x_1^{r1} x_n^{l0} C_x^r
\end{array}$$

The former introduced alignment is called the 0-*alignment* of the strings in $S(\ell_1^x)$. On the other hand, we the define the 1-*alignment* of the strings in $S(\ell_1^x)$ as follows.

$$\begin{array}{ccc}
C_x^l x_1^{m0} x_n^{m1} C_x^r \quad \text{and} \quad x_n^{m1} C_x^r C_x^l x_1^{m0} & & C_x^l x_1^{r1} x_n^{l0} C_x^r \quad \text{and} \quad x_n^{l0} C_x^r C_x^l x_1^{r1}. \\
\downarrow & & \downarrow \\
x_n^{m1} C_x^r C_x^l x_1^{m0} x_n^{m1} C_x^r & \text{and} & C_x^l x_1^{r1} x_n^{l0} C_x^r C_x^l x_1^{r1}
\end{array}$$

In the remainder, we refer to both ways to overlap the four strings as 0/1-*alignments.*

Strings Corresponding to Matching Equations

Let $\mathcal{W}_x$ be a wheel in $\mathscr{L}$ and $M(\mathcal{W}_x)$ its associated perfect matching. Let $\{i,j\}$ be an edge in $M(\mathcal{W}_x)$ and $\ell_{\{i,j\}}^x \equiv x_i \oplus x_j = 0$ the associated matching equation. We now define the corresponding set $S\big(\ell_{\{i,j\}}^x\big)$ consisting of two strings. Assuming $i < j$, we introduce two strings of the form

$$x_j^{r0} x_j^{l0} x_i^{r1} x_i^{l1} \qquad \text{and} \qquad x_i^{r1} x_i^{l1} x_j^{r0} x_j^{l0}$$

corresponding to the matching equation. There are two ways to align those two strings to obtain an overlap of two letters. In the remainder, we refer to those alignments as 0/1-*alignments.* (The brace notation indicate original strings).

$$x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1} \qquad \text{and} \qquad x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}$$

$$\swarrow \qquad \searrow$$

$$\overbrace{x_j^{r0}x_j^{l0}\rlap{$\underbrace{\phantom{x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}}}_{x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}}$}x_i^{r1}x_i^{l1}}^{x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}}x_j^{r0}x_j^{l0} \qquad\qquad \overbrace{x_i^{r1}x_i^{l1}\rlap{$\underbrace{\phantom{x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}}}_{x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}}$}x_j^{r0}x_j^{l0}}^{x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}}x_i^{r1}x_i^{l1}$$

The first way to overlap the strings is called the 0-*alignment*, whereas the second one is called the 1-*alignment.* Next, we describe the strings corresponding to wheel equations.

Strings Corresponding to Cycle Equations

Let $\mathcal{W}_x$ be a wheel in $\mathscr{L}$ and $M(\mathcal{W}_x)$ its associated matching. Furthermore, let $\{i, j\}$ and $\{i+1, j'\}$ be both contained in $M(\mathcal{W}_x)$. Assuming $i < j$, we introduce the corresponding strings for $x_i \oplus x_{i+1} = 0$. If $i+1 < j'$, we have

$$x_i^{l1}x_{i+1}^{r1}x_i^{m0}x_{i+1}^{m0} \qquad \text{and} \qquad x_i^{m0}x_{i+1}^{m0}x_i^{l1}x_{i+1}^{r1}.$$

Otherwise $(i+1 > j')$, we use

$$x_i^{l1}x_{i+1}^{m1}x_i^{m0}x_{i+1}^{r0} \qquad \text{and} \qquad x_i^{m0}x_{i+1}^{r0}x_i^{l1}x_{i+1}^{m1}.$$

In case of $(i > j$ and $i+1 > j')$, we use

$$x_i^{m1}x_{i+1}^{m1}x_i^{l0}x_{i+1}^{r0} \qquad \text{and} \qquad x_i^{l0}x_{i+1}^{r0}x_i^{m1}x_{i+1}^{m1}.$$

Finally, if $(i > j$ and $i+1 < j')$ holds, we introduce

$$x_i^{m1}x_{i+1}^{r1}x_i^{l0}x_{i+1}^{m0} \qquad \text{and} \qquad x_i^{l0}x_{i+1}^{m0}x_i^{m1}x_{i+1}^{r1}.$$

Let x_i be a variable in $\mathscr{L}$ that occurs in an equation ℓ_j^3 with three variables. We now define the corresponding strings for the equations $x_{i-1} \oplus x_i = 0$ and

$x_i \oplus x_{i+1} = 0$. We assume that $\{i-1, j\}$ and $\{i+1, j'\}$ are both included in $M(\mathcal{W}_x)$. Furthermore, we assume $i-1 < j$ and $i+1 < j'$. If the equation ℓ_j^3 is of the form $x_i \oplus y \oplus z = 0$, we introduce

$$x_{i-1}^{l1} x_i^{r1} x_{i-1}^{m0} x_i^{m0} \qquad \text{and} \qquad x_{i-1}^{m0} x_i^{m0} x_{i-1}^{l1} x_i^{r1}.$$

for $x_{i-1} \oplus x_i = 0$. Furthermore, for $x_i \oplus x_{i+1} = 0$, we use the strings

$$x_i^{l1} x_{i+1}^{r1} x_i^{m0} x_{i+1}^{m0} \qquad \text{and} \qquad x_i^{m0} x_{i+1}^{m0} x_i^{l1} x_{i+1}^{r1}.$$

On the other hand, if the equation ℓ_j^3 is of the form $x_i \oplus y \oplus z = 1$, we introduce

$$x_{i-1}^{l1} x_i^{m1} x_{i-1}^{m0} x_i^{r0} \qquad \text{and} \qquad x_{i-1}^{m0} x_i^{r0} x_{i-1}^{l1} x_i^{m1}.$$

corresponding to the equation $x_{i-1} \oplus x_i = 0$. For $x_i \oplus x_{i+1} = 0$, we use the strings

$$x_i^{m1} x_{i+1}^{r1} x_i^{l0} x_{i+1}^{m0} \qquad \text{and} \qquad x_i^{l0} x_{i+1}^{m0} x_i^{m1} x_{i+1}^{r1}.$$

Analogously, we introduce the notation of 0/1-alignments for the strings in $S(\ell_{i+1}^x)$. For the strings,

$$x_i^{m1} x_{i+1}^{m1} x_i^{l0} x_{i+1}^{r0} \qquad \text{and} \qquad x_i^{l0} x_{i+1}^{r0} x_i^{m1} x_{i+1}^{m1},$$

we define the following alignments as 0/1-alignments.

$$\overbrace{x_i^{m1} x_{i+1}^{m1} \rlap{$\underbrace{x_i^{l0} x_{i+1}^{r0} x_i^{m1} x_{i+1}^{m1}}_{x_i^{l0} x_{i+1}^{r0} x_i^{m1} x_{i+1}^{m1}}$} \phantom{x_i^{l0} x_{i+1}^{r0}}}^{x_i^{m1} x_{i+1}^{m1} x_i^{l0} x_{i+1}^{r0}} \phantom{x_i^{m1} x_{i+1}^{m1}} \qquad \text{and} \qquad \overbrace{x_i^{l0} x_{i+1}^{r0} \rlap{$\underbrace{x_i^{m1} x_{i+1}^{m1} x_i^{l0} x_{i+1}^{r0}}_{x_i^{m1} x_{i+1}^{m1} x_i^{l0} x_{i+1}^{r0}}$} \phantom{x_i^{m1} x_{i+1}^{m1}}}^{x_i^{l0} x_{i+1}^{r0} x_i^{m1} x_{i+1}^{m1}} \phantom{x_i^{l0} x_{i+1}^{r0}}$$

The the former alignment is called the 1-alignment and the latter one is called the 0-alignment. Next, we describe the strings corresponding to equations with three variables.

Strings Corresponding to Equations with Three Variables

We now concentrate on equations with exactly three variables. Let ℓ_j^3 be an equation with three variables in $\mathscr{L}$. For every equation ℓ_j^3, we define two corresponding sets $S^A(\ell_j^3)$ and $S^B(\ell_j^3)$, both containing exactly three strings.

The set $S(\ell_j^3)$ is defined by the union $S^A(\ell_j^3) \cup S^B(\ell_j^3)$. We distinguish whether ℓ_j^3 is of the form $x \oplus y \oplus z = 1$ or $x \oplus y \oplus z = 0$. The description starts with the former case.

An equation of the form $x \oplus y \oplus z = 0$ is represented by $S^A(\ell_j^3)$ containing the strings

$$x^{r1}A_j^1x^{l1}y^{r1}A_j^2y^{l1} \qquad y^{r1}A_j^2y^{l1}x^{m0}A_j^3C_j \qquad x^{m0}A_j^3C_jx^{r1}A_j^1x^{l1}$$

and by $S^B(\ell_j^3)$ including the following strings.

$$x^{r1}B_j^1x^{l1}z^{r1}B_j^2z^{l1} \qquad z^{r1}B_j^2z^{l1}C_jB_j^3x^{m0} \qquad C_jB_j^3x^{m0}x^{r1}B_j^1x^{l1}$$

The strings in the set $S^A(\ell_j^3)$ can be aligned in a cyclic fashion in order to obtain different fragments, which we will use in our reduction. Every specific alignment possesses its own abbreviation given below.

$$x^{r1}A_j^1x^{l1}y^{r1}A_j^2y^{l1} \quad y^{r1}A_j^2y^{l1}x^{m0}A_j^3C_j \quad x^{m0}A_j^3C_jx^{r1}A_j^1x^{l1}$$

$$\downarrow$$

$$\mathrlap{\phantom{x^{r1}A_j^1x^{l1}}\underbrace{\phantom{y^{r1}A_j^2y^{l1}x^{m0}A_j^3C_j}}_{y^{r1}A_j^2y^{l1}x^{m0}A_j^3C_j}}\overbrace{x^{r1}A_j^1x^{l1}y^{r1}A_j^2y^{l1}}^{x^{r1}A_j^1x^{l1}y^{r1}A_j^2y^{l1}}\overbrace{x^{m0}A_j^3C_jx^{r1}A_j^1x^{l1}}^{x^{m0}A_j^3C_jx^{r1}A_j^1x^{l1}} \equiv x^{r1}A_jx^{l1} \text{ called } x^1\text{- alignment}$$

$$\downarrow$$

$$\mathrlap{\phantom{y^{r1}A_j^2y^{l1}}\overbrace{\phantom{x^{m0}A_j^3C_jx^{r1}A_j^1x^{l1}}}^{x^{m0}A_j^3C_jx^{r1}A_j^1x^{l1}}}\underbrace{y^{r1}A_j^2y^{l1}x^{m0}A_j^3C_j}_{y^{r1}A_j^2y^{l1}x^{m0}A_j^3C_j}\underbrace{x^{r1}A_j^1x^{l1}y^{r1}A_j^2y^{l1}}_{x^{r1}A_j^1x^{l1}y^{r1}A_j^2y^{l1}} \equiv y^{r1}A_jy^{l1} \text{ called } y^1\text{- alignment}$$

$$\downarrow$$

$$\mathrlap{\phantom{x^{m0}A_j^3C_j}\overbrace{\phantom{x^{r1}A_j^1x^{l1}y^{r1}A_j^2y^{l1}}}^{x^{r1}A_j^1x^{l1}y^{r1}A_j^2y^{l1}}}\underbrace{x^{m0}A_j^3C_jx^{r1}A_j^1x^{l1}}_{x^{m0}A_j^3C_jx^{r1}A_j^1x^{l1}}\underbrace{y^{r1}A_j^2y^{l1}x^{m0}A_j^3C_j}_{y^{r1}A_j^2y^{l1}x^{m0}A_j^3C_j} \equiv x^{m0}A_jC_j \text{ called left-}x^0\text{- alignment}$$

The strings in $S^B(\ell_j^3)$ can also be aligned in a cyclic fashion. For each of the fragments, we are going to define an abbreviation.

$$x^{r1}B_j^1x^{l1}z^{r1}B_j^2z^{l1} \quad z^{r1}B_j^2z^{l1}C_jB_j^3x^{m0} \quad C_jB_j^3x^{m0}x^{r1}B_j^1x^{l1}$$

$$\downarrow$$

$$\underbrace{C_jB_j^3x^{m0}\mathrlap{\overbrace{\phantom{x^{r1}B_j^1x^{l1}z^{r1}B_j^2z^{l1}}}^{x^{r1}B_j^1x^{l1}z^{r1}B_j^2z^{l1}}}x^{r1}B_j^1x^{l1}}_{C_jB_j^3x^{m0}x^{r1}B_j^1x^{l1}}\underbrace{z^{r1}B_j^2z^{l1}C_jB_j^3x^{m0}}_{z^{r1}B_j^2z^{l1}C_jB_j^3x^{m0}} \equiv C_jB_jx^{m0} \text{ called right-}x^0\text{- alignment}$$

$$\downarrow$$

$$\overbrace{x^{r1}B_j^1x^{l1}\mathrlap{\underbrace{\phantom{z^{r1}B_j^2z^{l1}C_jB_j^3x^{m0}}}_{z^{r1}B_j^2z^{l1}C_jB_j^3x^{m0}}}z^{r1}B_j^2z^{l1}}^{x^{r1}B_j^1x^{l1}z^{r1}B_j^2z^{l1}}\overbrace{C_jB_j^3x^{m0}x^{r1}B_j^1x^{l1}}^{C_jB_j^3x^{m0}x^{r1}B_j^1x^{l1}} \equiv x^{r1}B_jx^{l1} \text{ called } x^1\text{- alignment}$$

$$\downarrow$$

$$\underbrace{z^{r1}B_j^2z^{l1}\mathrlap{\overbrace{\phantom{C_jB_j^3x^{m0}x^{r1}B_j^1x^{l1}}}^{C_jB_j^3x^{m0}x^{r1}B_j^1x^{l1}}}C_jB_j^3x^{m0}}_{z^{r1}B_j^2z^{l1}C_jB_j^3x^{m0}}\underbrace{x^{r1}B_j^1x^{l1}z^{r1}B_j^2z^{l1}}_{x^{r1}B_j^1x^{l1}z^{r1}B_j^2z^{l1}} \equiv z^{r1}B_jz^{l1} \text{ called } z^1\text{- alignment}$$

The strings in $S^B(\ell_j^3)$ and $S^A(\ell_j^3)$ can be overlapped in a special way that corresponds to assigning the value 0 to x.

$$x^{r1}A_j^1x^{l1}y^{r1}A_j^2y^{l1} \quad y^{r1}A_j^2y^{l1}x^{m0}A_j^3C_j \quad x^{m0}A_j^3C_jx^{r1}A_j^1x^{l1}$$

$$z^{r1}B_j^2z^{l1}C_jB_j^3x^{m0} \quad C_jB_j^3x^{m0}x^{r1}B_j^1x^{l1} \quad x^{r1}B_j^1x^{l1}z^{r1}B_j^2z^{l1}$$

$$\downarrow$$

$$\underbrace{x^{m0}A_j^3C_j\mathrlap{\overbrace{\phantom{x^{r1}A_j^1x^{l1}y^{r1}A_j^2y^{l1}}}^{x^{r1}A_j^1x^{l1}y^{r1}A_j^2y^{l1}}}x^{r1}A_j^1x^{l1}}_{x^{m0}A_j^3C_jx^{r1}A_j^1x^{l1}}\underbrace{y^{r1}A_j^2y^{l1}x^{m0}A_j^3\mathrlap{\overbrace{\phantom{C_jB_j^3x^{m0}x^{r1}B_j^1x^{l1}}}^{C_jB_j^3x^{m0}x^{r1}B_j^1x^{l1}}}C_j}_{y^{r1}A_j^2y^{l1}x^{m0}A_j^3C_j}B_j^3x^{m0}\underbrace{x^{r1}B_j^1x^{l1}\mathrlap{\overbrace{\phantom{z^{r1}B_j^2z^{l1}C_jB_j^3x^{m0}}}^{z^{r1}B_j^2z^{l1}C_jB_j^3x^{m0}}}z^{r1}B_j^2z^{l1}}_{x^{r1}B_j^1x^{l1}z^{r1}B_j^2z^{l1}}C_jB_j^3x^{m0}$$

In the remainder, we call this alignment the x^0-alignment of $S(\ell_j^3)$ and use the abbreviation $x^{m0}C_jx^{m0}$ for this string.

When the equation with three variables is of the form $\ell_j^3 \equiv x \oplus y \oplus z = 1$, the set $S^A(\ell_j^3)$ contains the following strings.

$$x^{r0}A_j^1x^{l0}y^{r0}A_j^2y^{l0} \qquad y^{r0}A_j^2y^{l0}x^{m1}A_j^3C_j \qquad x^{m1}A_j^3C_jx^{r0}A_j^1x^{l0}$$

Furthermore, the set $S^B(\ell_j^3)$ includes three strings defined as follows.

$$x^{r0}B_j^1x^{l0}z^{r0}B_j^2z^{l0} \qquad z^{r0}B_j^2z^{l0}C_jB_j^3x^{m1} \qquad C_jB_j^3x^{m1}x^{r0}B_j^1x^{l0}$$

The corresponding alignments for the strings in $S^A(\ell_j^3)$ and $S^B(\ell_j^3)$ can be defined in an analogous way.

9.9 Constructing Superstrings from Assignments

Given an assignment ϕ to the variables of $\mathscr{L}$, we are going to construct the associated superstring s_ϕ for the instance $\mathcal{S}_{\mathscr{L}}$.

For every equation ℓ in $\mathscr{L}$, we formulate rules for aligning the corresponding strings in $S(\ell)$ according to the assignment ϕ. We start with sets corresponding to wheel border equations and cycle equations. Afterwards, we show how the actual fragments can be overlapped with strings from the sets corresponding to matching equations and equations with three variables. Furthermore, we analyze the relation between the number of satisfied equations by ϕ and the length of the obtained superstring s_ϕ. We begin with the description of the alignment of strings corresponding to wheel border equations in $\mathscr{L}$.

Aligning Strings Corresponding to Wheel Border Equations

Let $\mathcal{W}_x$ be a wheel in $\mathscr{L}$ and $x_1 \oplus x_n = 0$ its wheel border equation. Furthermore, we assume that x_n is contained in a equation with three variables of the form $x_n \oplus y \oplus z = 0$. First, we set the string $L_xC_x^l$ as the initial part of our superstring corresponding to the wheel $\mathcal{W}_x$. Then, we use the $\phi(x_1)$-alignment of the strings

$$C_x^lx_1^{m0}x_n^{m1}C_x^r, \quad x_n^{l1}C_x^rC_x^lx_1^{m0}, \quad C_x^lx_1^{r1}x_n^{m0}C_x^r, \quad \text{and} \quad x_n^{m0}C_x^rC_x^lx_1^{r1}.$$

In this condition, one of the strings s_l can be overlapped from the left side with $L_x C_x^l$ by one letter. The other string s_r will be joined from the right side with $C_x^r R_x$ by one letter. This construction will help us to check whether ϕ assigns the same value to the variable x_n as to x_1. The string s_r can be interpreted as the $\phi(x_1)$-alignment of the strings corresponding to $x_n \oplus x_{n+1} = 0$, since the first letter of s_r is either x_n^{m0} or x_n^{l1}.

The parts corresponding to a wheel border equation with $x_n \oplus y \oplus z = 1$ can be constructed analogously. Next, we are going to align strings corresponding to cycle equations.

Aligning Strings Corresponding to Cycle Equations

Let $x_i \oplus x_{i+1} = 0$ be a cycle equation contained in $\mathscr{L}$. Furthermore, let the corresponding strings be given by

$$x_i^{m0} x_{i+1}^{m0} x_i^{l1} x_{i+1}^{r1} \quad \text{and} \quad x_i^{l1} x_{i+1}^{r1} x_i^{m0} x_{i+1}^{m0}.$$

In dependence of the given assignment ϕ, we use simple alignments to overlap the considered strings. More precisely, we make use of the $\phi(x_{i+1})$-alignment. For every pair of associated strings, we derive an overlap of two letters. We are going to align those fragments with strings corresponding to matching equations and equations with three variables.

Aligning Strings Corresponding to Matching Equations

Let $x_i \oplus x_j = 0$ be a matching equation in $\mathscr{L}$. Let us assume that $i < j$ holds. We define the alignment of the strings in $S(\ell^x_{\{i,j\}})$ according to the value of $\phi(x_{i+1})$. More precisely, we use the $\phi(x_{i+1})$-alignment of the strings

$$x_j^{r0} x_j^{l0} x_i^{r1} x_i^{l1} \quad \text{and} \quad x_i^{r1} x_i^{l1} x_j^{r0} x_j^{l0}.$$

Due to this alignment, we obtain an overlap of two letters. We are going to analyze the length of the resulting superstring in dependence of the assignment ϕ to the variables x_i, x_{i+1}, x_j and x_{j+1}. We start with the case $\phi(x_{i+1}) = \phi(x_{j+1}) = 1$.

Case ($\phi(x_{i+1}) = \phi(x_{j+1}) = 1$):
We use the 1-alignment of the strings $x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}$ and $x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}$. The situation is displayed below. (The two triangle notation $\rhd\rhd$ and $\lhd\lhd$ will be explained hereafter.)

$$b \rhd \rhd \boxed{X_i}\, x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1} \boxed{x_i^{l1}} \lhd \lhd m \rhd \rhd \boxed{X_j}\, x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0} \boxed{x_j^{m1}} \lhd \lhd e$$

$$\downarrow$$

$$b \rhd \rhd \boxed{X_i}\, \mathrlap{\overbrace{\phantom{x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}}}^{x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}}}x_i^{r1}x_i^{l1}\underbrace{x_j^{r0}x_j^{l0}x_i^{r1}\boxed{x_i^{l1}}}_{x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}} \lhd \lhd m \rhd \rhd \boxed{X_j} \boxed{x_j^{m1}} \lhd \lhd e$$

The actual superstring s is denoted by the following sequence.

$$s = b \rhd \rhd \boxed{X_i}\, x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1} \boxed{x_i^{l1}} \lhd \lhd m \rhd \rhd \boxed{X_j}\, x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0} \boxed{x_j^{m1}} \lhd \lhd e$$

The part $\rhd \rhd \boxed{X_i}$ represents a simple alignment of the strings corresponding to $x_{i-1} \oplus x_i = 0$ ending with the letter $X_i \in \{x_i^{m0}, x_i^{r1}\}$, which means

$$\rhd \rhd \boxed{X_i} \in \{x_{i-1}^{m0}x_i^{m0}x_{i-1}^{l1}x_i^{r1}x_{i-1}^{m0}x_i^{m0}, x_{i-1}^{l1}x_i^{r1}x_{i-1}^{m0}x_i^{m0}x_{i-1}^{l1}x_i^{r1}\}.$$

The letter in the box emphasizes the letter which can be used to overlap from the right side with other strings. Furthermore, the string $\boxed{x_i^{l1}} \lhd \lhd$ denotes $x_i^{l1}x_{i+1}^{r1}x_i^{m0}x_{i+1}^{m0}x_i^{l1}x_{i+1}^{r1}$. Analogously, $\rhd \rhd \boxed{X_j}$ is a simple alignment of the strings corresponding to $x_{j-1} \oplus x_j = 0$, where $X_j \in \{x_j^{r0}, x_j^{m1}\}$. Furthermore, we use $\boxed{x_j^{m1}} \lhd \lhd$ to denote $x_j^{m1}x_{j+1}^{m1}x_j^{l0}x_{j+1}^{r0}x_j^{m1}x_{j+1}^{m1}$. Finally, b, m and e are sequences of letters, which we do not specify in detail. They define the remaining parts of the superstring.
If $X_i = x_i^{r1}$ holds, we align $\rhd \rhd \boxed{X_i}$ with $x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}$ to achieve an additional overlap of one letter. An analogous situation holds for $\rhd \rhd \boxed{X_j}$ and $\boxed{x_j^{m1}} \lhd \lhd$. All in all, we obtain an overlap of three letters if $\phi(x_i) = \phi(x_{i+1}) = 1$ and $\phi(x_{j+1}) = \phi(x_j) = 1$ holds. Otherwise, we lose an overlap of one letter per unsatisfied equation.

Case $(\phi(x_{i+1}) = \phi(x_{j+1}) = 0)$:
We use the 0-alignment of the strings $x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}$ and $x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}$.

$$b \triangleright \triangleright \boxed{X_i}\, x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1} \boxed{x_i^{m0}} \triangleleft \triangleleft\, m \triangleright \triangleright \boxed{X_j}\, x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0} \boxed{x_j^{l0}} \triangleleft \triangleleft\, e$$

$$\downarrow$$

$$b \triangleright \triangleright \boxed{X_i}\boxed{x_i^{m0}} \triangleleft \triangleleft\, m \triangleright \triangleright \boxed{X_j}\, \mathrlap{\overbrace{x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}}^{x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}}}x_j^{r0}x_j^{l0}\underbrace{x_i^{r1}x_i^{l1}x_j^{r0}\boxed{x_j^{l0}}}_{x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}} \triangleleft \triangleleft\, e$$

In this case, we use $\boxed{x_i^{m0}} \triangleleft \triangleleft$ as an abbreviation for $x_i^{m0}x_{i+1}^{m0}x_i^{l1}x_{i+1}^{r1}x_i^{m0}x_{i+1}^{m0}$ and $\boxed{x_j^{l0}} \triangleleft \triangleleft$ for $x_j^{l0}x_{j+1}^{r0}x_j^{m1}x_{j+1}^{m1}x_j^{l0}x_{j+1}^{r0}$. If $X_i = x_i^{m0}$ holds, we align $\triangleright \triangleright \boxed{X_i}$ with $\boxed{x_i^{m0}} \triangleleft \triangleleft$ and gain an additional overlap of one letter. An analogous situation holds for $\triangleright \triangleright \boxed{X_j}$ and $\boxed{x_j^{l0}} \triangleleft \triangleleft$. Hence, we obtain an overlap of three letters if $\phi(x_{i+1}) = \phi(x_i) = 0$ and $\phi(x_{j+1}) = \phi(x_j) = 0$ holds. If the corresponding equation with two variables is not satisfied, we lose an overlap of one letter.

Case $(\phi(x_{i+1}) \neq \phi(x_{j+1}) = 1)$:
In this case, we use the 0-alignment of the strings $x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}$ and $x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}$.

$$b \triangleright \triangleright \boxed{X_i}\, x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1} \boxed{x_i^{m0}} \triangleleft \triangleleft\, m \triangleright \triangleright \boxed{X_j}\, x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0} \boxed{x_j^{m1}} \triangleleft \triangleleft\, e$$

$$\downarrow$$

$$b \triangleright \triangleright \boxed{X_i}\boxed{x_i^{m0}} \triangleleft \triangleleft\, m \triangleright \triangleright \boxed{X_j}\boxed{x_j^{m1}} \triangleleft \triangleleft\, e \quad \mathrlap{\overbrace{x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}}^{x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}}}x_j^{r0}x_j^{l0}\underbrace{x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}}_{x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}}$$

We attach $x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}$ at the end of our actual solution s_ϕ without having any overlap with the so far obtained superstring. Notice that we obtain in each case an additional overlap of one letter if the corresponding equation with two variables is satisfied, i.e. $X_i = x_i^{m0}$ and $X_j = x_j^{m1}$.

Case ($\phi(x_{i+1}) \neq \phi(x_{j+1}) = 0$):
According to ϕ, we use the 1-alignment of the strings $x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}$ and $x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}$.

$$b \triangleright \triangleright \boxed{X_i}\, x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}\, \boxed{x_i^{l1}} \triangleleft \triangleleft m \triangleright \triangleright \boxed{X_j}\, x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}\, \boxed{x_j^{l0}} \triangleleft \triangleleft e$$

$$\downarrow$$

$$b \triangleright \triangleright \boxed{X_i}\, \mathrlap{\overbrace{\phantom{x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}}}^{x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}}} x_i^{r1}x_i^{l1}\underbrace{x_j^{r0}x_j^{l0}x_i^{r1}\boxed{x_i^{l1}}}_{x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}} \triangleleft \triangleleft m \triangleright \triangleright \boxed{X_j}\, \boxed{x_j^{l0}} \triangleleft \triangleleft e$$

We join $x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}$ from the right side with $\boxed{x_i^{l1}} \triangleleft \triangleleft$ and obtain an overlap of one letter. This reduces the length of the superstring by one letter independent of the assignment $\phi(x_j)$. In case of $X_i = x_i^{r1}$, we achieve another overlap of one letter, since we are able to align $\triangleright \triangleright \boxed{X_i}$ from the right side with $x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}$. It corresponds to the satisfied equation $x_i \oplus x_{i+1} = 0$. Hence, we obtain at least the same number of overlapped letters as satisfied equations.

As for the next step, we are going to align strings corresponding to equations with three variables.

Aligning Strings Corresponding to Equations with Three Variables

Let ℓ_j^3 be an equation with three variables x, y and z in $\mathscr{L}$. Furthermore, let $x_{i-1} \oplus x = 0$, $x \oplus x_{i+1} = 0$, $y_{j-1} \oplus y = 0$, $y \oplus y_{j+1} = 0$, $z_{k-1} \oplus z = 0$ and $z \oplus z_{k+1} = 0$ be the equations with two variables, in which the variables

x, y and z occur. Given the assignment ϕ to x, y and z, we are going to define the alignment of the corresponding strings. Let us start with equations of the form $\ell_j^3 \equiv x \oplus y \oplus z = 0$. Then, we define the rules for aligning strings in $S^A(\ell_j^3)$ and $S^B(\ell_j^3)$ as follows. In particular, we handle the cases $\big(\phi(x_{i+1}) + \phi(y_{j+1}) + \phi(z_{k+1})\big) \in \{3,2,1,0\}$ separately starting with $\phi(x_{i+1}) + \phi(y_{j+1}) + \phi(z_{k+1}) = 3$.

Case $(\phi(x_{i+1}) + \phi(y_{j+1}) + \phi(z_{k+1}) = 3)$**:**
In this case, we align the strings in $S(\ell_j^3)$ in such a way that we obtain the former introduced strings $y^{r1}A_jy^{l1}$ and $z^{r1}B_jz^{l1}$. The situation, which we want to analyze, is displayed below.

$$b \triangleright \triangleright \boxed{X}\,\boxed{x^{l1}} \triangleleft \triangleleft\, m_1 \triangleright \triangleright \boxed{Y}\; y^{r1}A_jy^{l1}\; \boxed{y^{l1}} \triangleleft \triangleleft\, m_2 \triangleright \triangleright \boxed{Z}\; z^{r1}B_jz^{l1}\; \boxed{z^{l1}} \triangleleft \triangleleft\, e$$

$$\downarrow$$

$$b \triangleright \triangleright \boxed{X}\,\boxed{x^{l1}} \triangleleft \triangleleft\, m_1 \triangleright \triangleright \boxed{Y}\; \underbrace{y^{r1}A_j\boxed{y^{l1}}}_{y^{r1}A_jy^{l1}} \triangleleft \triangleleft\, m_2 \triangleright \triangleright \boxed{Z}\; \overbrace{z^{r1}B_j\boxed{z^{l1}}}^{z^{r1}B_jz^{l1}} \triangleleft \triangleleft\, e$$

Similarly to the situations that we discussed concerning matching equations, we define the actual superstring s in the way described below.

$$s = b \triangleright \triangleright \boxed{X}\,\boxed{x^{l1}} \triangleleft \triangleleft m_1 \triangleright \triangleright \boxed{Y} y^{r1}A_jy^{l1}\boxed{y^{l1}} \triangleleft \triangleleft m_2 \triangleright \triangleright \boxed{Z} z^{r1}B_jz^{l1}\boxed{z^{l1}} \triangleleft \triangleleft e$$

Here, b, m_1, m_2 and e denote parts of s, which we do not specify in detail to emphasize the parts corresponding to the equation with three variables. The string $\boxed{x^{l1}} \triangleleft \triangleleft$ denotes the $\phi(x_{i+1})$-alignment of the strings in $S(\ell_{i+1}^x)$. The strings $\boxed{z^{l1}} \triangleleft \triangleleft$ and $\boxed{y^{l1}} \triangleleft \triangleleft$ are defined analogously. In this situation, we want to analyze the cases $X \in \{x^{r1}, x^{m0}\}$, $Y \in \{y^{r1}, y^{m0}\}$ and $Z \in \{z^{r1}, z^{m0}\}$. We infer that we obtain an overlap of four letters if all equations with two variables are satisfied. Otherwise, we lose an overlap of one letter per unsatisfied equation with two variables.

Case $(\phi(x_{i+1}) + \phi(y_{j+1}) + \phi(z_{k+1}) = 2)$:
Let $\alpha, \gamma \in \{x_{i+1}, y_{j+1}, z_{k+1}\}$ be variables such that $\phi(\gamma) = \phi(\alpha) = 1$ holds. Then, we use the α^1-alignment and γ^1-alignment of the strings in $S^A(\ell_j^3)$ and $S^B(\ell_j^3)$ breaking ties arbitrarily. We display exemplary the situation for $\phi(z_{k+1}) = \phi(x_{i+1}) = 1$.

$$b \rhd \rhd \boxed{X}\, x^{r1}A_jx^{l1}\,\boxed{x^{l1}} \lhd \lhd m_1 \rhd \rhd \boxed{Y}\,\boxed{y^{m0}} \lhd \lhd m_2 \rhd \rhd \boxed{Z}\, z^{r1}B_jz^{l1}\,\boxed{z^{l1}} \lhd \lhd e$$

$$\downarrow$$

$$b \rhd \rhd \boxed{X}\, \underbrace{x^{r1}A_j\boxed{x^{l1}}}_{x^{r1}A_jx^{l1}} \lhd \lhd m_1 \rhd \rhd \boxed{Y}\,\boxed{y^{m0}} \lhd \lhd m_2 \rhd \rhd \boxed{Z}\, \overbrace{z^{r1}B_j\boxed{z^{l1}}}^{z^{r1}B_jz^{l1}} \lhd \lhd e$$

In this case, we achieve an overlap of five letters if all equations with two variables are satisfied. Otherwise, we lose an overlap of one letter per unsatisfied equation with two variables.

Case $(\phi(x_{i+1}) + \phi(y_{j+1}) + \phi(z_{k+1}) = 1)$:
If $\phi(z_{k+1}) + \phi(x_{i+1}) = 1$ holds, we align the strings in $S^B(\ell_j^3)$ and $S^A(\ell_j^3)$ to obtain $x^{r1}A_jx^{l1}$ and $z^{r1}B_jz^{l1}$. Otherwise, we make use of the strings $x^{r1}B_jx^{l1}$ and $y^{r1}A_jy^{l1}$. We display the situation for $\phi(y_{j+1}) = 1$.

$$b \rhd \rhd \boxed{X}\, x^{r1}B_jx^{l1}\boxed{x^{m0}} \lhd \lhd m_1 \rhd \rhd \boxed{Y}\, y^{r1}A_jy^{l1}\boxed{y^{l1}} \lhd \lhd m_2 \rhd \rhd \boxed{Z}\,\boxed{z^{m0}} \lhd \lhd e$$

$$\downarrow$$

$$b \rhd \rhd \boxed{X}\,\boxed{x^{m0}} \lhd \lhd m_1 \rhd \rhd \boxed{Y}\, \overbrace{y^{r1}A_j\boxed{y^{l1}}}^{y^{r1}A_jy^{l1}} \lhd \lhd m_2 \rhd \rhd \boxed{Z}\,\boxed{z^{m0}} \lhd \lhd e\, x^{r1}B_jx^{l1}$$

Notice that we obtain an overlap of four letters if the equations with two variables are satisfied, i.e. $X = x^{m0}$, $Z = z^{m0}$ and $Y = y^{r1}$. Otherwise, we lose an overlap of one letter per unsatisfied equation with two variables.

Case $(\phi(x_{i+1}) + \phi(y_{j+1}) + \phi(z_{k+1}) = 0)$**:**
In this case, we use the x^0-alignment of the strings in $S(\ell_j^3)$. The situation is displayed below.

$$b \triangleright \triangleright \boxed{X}\, x^{m0} C_j x^{m0}\ \boxed{x^{m0}} \triangleleft \triangleleft m_1 \triangleright \triangleright \boxed{Y}\boxed{y^{m0}} \triangleleft \triangleleft m_2 \triangleright \triangleright \boxed{Z}\boxed{z^{m0}} \triangleleft \triangleleft e$$

$$\downarrow$$

$$b \triangleright \triangleright \boxed{X}\ \underbrace{x^{m0} C_j \boxed{x^{m0}}}_{x^{m0} C_j x^{m0}} \triangleleft \triangleleft m_1 \triangleright \triangleright \boxed{Y}\boxed{y^{m0}} \triangleleft \triangleleft m_2 \triangleright \triangleright \boxed{Z}\boxed{z^{m0}} \triangleleft \triangleleft e$$

Here, we are able to achieve an overlap of five letters if all equations with two variables are satisfied, i.e. $X = x^{m0}$, $Z = z^{m0}$ and $Y = y^{m0}$.

The situation for equations of the form $x \oplus y \oplus z = 1$ can be analyzed analogously. In summary, we obtain the following statement.

Lemma 9.9.1
Given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 *problem with n wheels, m_2 equations with two variables, m_3 equations with three variables and an assignment ϕ to the variables of $\mathscr{L}$ leaving u equations unsatisfied, then, it is possible to construct a superstring s_ϕ for $\mathcal{S}_{\mathscr{L}}$ with length at most*

$$7 \cdot n + m_2 \cdot 5 + m_3 \cdot 22 + u.$$

As for the next step, we are going to define the assignment ψ_s, which is associated to a given superstring s for $\mathcal{S}_{\mathscr{L}}$.

9.10 Defining Assignments From Superstrings

Given a superstring s for $\mathcal{S}_{\mathscr{L}}$, we are going to define the associated assignment ψ_s to the variables in $\mathscr{L}$. In order to deduce the values assigned to the variables in $\mathscr{L}$ from s, we have to normalize the underlying superstring s. For this reason, we define rules that transform a superstring for $\mathcal{S}_{\mathscr{L}}$ into a normed superstring for $\mathcal{S}_{\mathscr{L}}$ without increasing the length.

Let us first give the definition of a normed superstring for $\mathcal{S}_{\mathscr{L}}$.

Definition 9.10.1 (Normed Superstring for $\mathcal{S}_{\mathscr{L}}$)
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem, $\mathcal{S}_{\mathscr{L}}$ the corresponding instance of the* SHORTEST SUPERSTRING *problem and s a superstring for $\mathcal{S}_{\mathscr{L}}$. We refer to s as a normed superstring for $\mathcal{S}_{\mathscr{L}}$ if for every equation ℓ in $\mathscr{L}$, the superstring s contains s_ℓ as a substring, where s_ℓ is a 0/1-alignment of the strings in $S(\ell)$.*

After having defined a normed superstring, we are going to state rules which transform a superstring for $\mathcal{S}_{\mathscr{L}}$ into a normed superstring for $\mathcal{S}_{\mathscr{L}}$ without increasing the length. All transformation can be performed in polynomial time. Once accomplished to generate a normed superstring, we are able to define the assignment ψ_s and analyze the number of overlapped letters in relation to the number of satisfied equations in $\mathscr{L}$ by ψ_s. Let us start with transformations of strings corresponding to cycle equations and wheel border equations.

Normalizing Strings for Cycle and Wheel Border Equations

Let $x_i \oplus x_{i+1} = 0$ be a cycle equation in $\mathscr{L}$. Furthermore, let $x_i^{m0}x_{i+1}^{r0}x_i^{l1}x_{i+1}^{m1}$ and $x_i^{l1}x_{i+1}^{m1}x_i^{m0}x_{i+1}^{r0}$ be its corresponding strings. We note that these strings can have an overlap of at most one letter from the left side as well as from the right side with other strings in $\mathcal{S}_{\mathscr{L}}$. Given a superstring s for $\mathcal{S}_{\mathscr{L}}$, we obtain at least the same number of overlapped letters if we use one of the 0/1-alignments in s. More precisely, we will use the 0/1-alignment that maximizes the overlap with the remaining strings after separating $x_i^{m0}x_{i+1}^{r0}x_i^{l1}x_{i+1}^{m1}$ and $x_i^{l1}x_{i+1}^{m1}x_i^{m0}x_{i+1}^{r0}$ from s. In order to build some intuition, we first consider the following example.

Example 9.10.1
Let S be a finite set of strings over the alphabet Σ such that no string is a substring of another string in S and $\{x_i^{m0}x_{i+1}^{r0}x_i^{l1}x_{i+1}^{m1},\ x_i^{l1}x_{i+1}^{m1}x_i^{m0}x_{i+1}^{r0}\} \subseteq S$. Let s be a superstring for S defined by

$$s \;=\; \mathrlap{\overbrace{bx_i^{m0}}^{bx_i^{m0}}\phantom{x_{i+1}^{r0}x_i^{l1}}\overbrace{x_{i+1}^{m1}mx_i^{l1}}^{x_{i+1}^{m1}mx_i^{l1}}\phantom{x_{i+1}^{m1}x_i^{m0}}\overbrace{x_{i+1}^{r0}e}^{x_{i+1}^{r0}e}}\underbrace{x_i^{m0}x_{i+1}^{r0}x_i^{l1}x_{i+1}^{m1}}_{x_i^{m0}x_{i+1}^{r0}x_i^{l1}x_{i+1}^{m1}}\underbrace{x_i^{l1}x_{i+1}^{m1}x_i^{m0}x_{i+1}^{r0}}_{x_i^{l1}x_{i+1}^{m1}x_i^{m0}x_{i+1}^{r0}}$$

where $b, m, e \in \Sigma^+$. By separating $x_i^{m0}x_{i+1}^{r0}x_i^{l1}x_{i+1}^{m1}$ and $x_i^{l1}x_{i+1}^{m1}x_i^{m0}x_{i+1}^{r0}$ from s, we obtain three remaining strings bx_i^{m0}, $x_{i+1}^{m1}mx_i^{l1}$ and $x_{i+1}^{r0}e$ such that

$$bx_i^{m0}x_{i+1}^{m1}mx_i^{l1}x_{i+1}^{r0}e$$

is a superstring for $S\backslash\{x_i^{m0}x_{i+1}^{r0}x_i^{l1}x_{i+1}^{m1},\ x_i^{l1}x_{i+1}^{m1}x_i^{m0}x_{i+1}^{r0}\}$. Then, we define the transformed superstring s' for S with at least the same number of overlapped letters by

$$s' \;=\; \mathrlap{\phantom{bx_i^{m0}x_{i+1}^{r0}}\overbrace{x_i^{l1}x_{i+1}^{m1}x_i^{m0}x_{i+1}^{r0}}^{x_i^{l1}x_{i+1}^{m1}x_i^{m0}x_{i+1}^{r0}}}b\underbrace{x_i^{m0}x_{i+1}^{r0}x_i^{l1}x_{i+1}^{m1}}_{x_i^{m0}x_{i+1}^{r0}x_i^{l1}x_{i+1}^{m1}}x_i^{m0}x_{i+1}^{r0}e\;x_{i+1}^{m1}mx_i^{l1}.$$

Given a wheel $\mathcal{W}_x$ in $\mathscr{L}$, we successively transform the underlying superstring s for $\mathcal{S}_{\mathscr{L}}$ according to the order of the cycle equations ℓ_i^x in $\mathcal{W}_x$. For each equation ℓ_i^x in the wheel $\mathcal{W}_x$, we apply the following rule that determines the particular 0/1-alignment of the strings in $S(\ell_i^x)$ that is used in s.

We denote by s^1 and s^0 the 1-alignment and the 0-alignment of the strings in $S(\ell_i^x)$, respectively. Furthermore, let s_1, s_2 and s_3 be the strings that result by separating the strings in $S(\ell_i^x)$ from s. Note that one of them could be the empty string Λ_0. Then, we say that the strings in $S(\ell_i^x)$ use the 1-alignment in s if the following condition holds.

$$\max_{\{t,j\}\subseteq[3]}\{|ov(s_t, s^1)| + |ov(s^1, s_j)|\} \;\geq\; \max_{\{t,j\}\subseteq[3]}\{|ov(s_t, s^0)| + |ov(s^0, s_j)|\}$$

Otherwise, we say that the strings in $S(\ell_i^x)$ use the 0-alignment in s.

Let us assume that the strings in $S(\ell_i^x)$ use the a-alignment in s, where $a \in \{0,1\}$. Furthermore, let $s_x, s_y \in \{s_1, s_2, s_3\}$ be strings with the following property.

$$|ov(s_x, s^a)| + |ov(s^a, s_y)| \quad \geq \quad \max_{\substack{\{t,j\} \subseteq [3] \\ b \in \{0,1\}}} \left\{ |ov(s_t, s^b)| + |ov(s^b, s_j)| \right\}$$

Let $k \in \{1,2,3\} \backslash \{x, y\}$. Then, we define the transformed superstring as follows.

$$s' \;=\; pref(s_x, s^a)\, pref(s^a, s_y)\; s_y\; s_k$$

Let us fix an arbitrary order of wheels in $\mathscr{L}$. For each wheel $\mathcal{W}_x$ in the fixed order and for every cycle equation ℓ_i^x in $\mathcal{W}_x$, we define the 0/1-alignment for the strings in $S(\ell_i^x)$ in s according to the rule given above and apply the corresponding transformation. Note that we obtain a superstring s' for $\mathcal{S}_{\mathscr{L}}$ with $|s'| \leq |s|$.

For wheel border equations, we use a similar argumentation. Let $\ell_1^x \equiv x_1 \oplus x_n = 0$ be the wheel border equation of $\mathcal{W}_x$. Furthermore, we let the strings in $S(\ell_1^x)$ be represented by

$$L_x C_x^l,\ C_x^l x_1^{m0} x_n^{l1} C_x^r,\ x_n^{l1} C_x^r C_x^l x_1^{m0},\ C_x^l x_1^{r1} x_n^{m0} C_x^r,\ x_n^{m0} C_x^r C_x^l x_1^{r1} \text{ and } C_x^r R_x.$$

Since the 0/1-alignments of the strings in $S(\ell_1^x)$ achieve an overlap of two letters for each pair, $\{C_x^l x_1^{m0} x_n^{l1} C_x^r,\ x_n^{l1} C_x^r C_x^l x_1^{m0}\}$ and $\{C_x^l x_1^{r1} x_n^{m0} C_x^r,\ x_n^{m0} C_x^r C_x^l x_1^{r1}\}$, we argue as before that these strings can be rearranged such that these pairs use a 0/1-alignment without increasing the length of the underlying superstring for $\mathcal{S}_{\mathscr{L}}$. Note that if both pairs are using the same 0/1-alignment, it is possible to overlap one of the mentioned pairs with $L_x C_x^l$ from the left side and the other one with $C_x^r R_x$ from the right. This construction checks whether the variables x_1 and x_n have the same assigned value, which is rewarded by another overlap of one letter.

For the aforementioned fixed order of the wheels in $\mathscr{L}$, we build the backbone of our new superstring consisting of the concatenation of the strings

$s^x s^y \dots s^z$, where the string s^x is associated to its wheel $\mathcal{W}^x$. Furthermore, s^x consists of the corresponding 0/1-alignments of the strings in $S(\ell_i^x)$ used in s for each cycle equation ℓ_i^x in $\mathcal{W}_x$ and the order of the strings is given by the order of appearance of cycle equations in $\mathcal{W}_x$. The string s^x starts with the letter L_x and ends with R_x.

Note that similar transformations can be applied to strings corresponding to matching equations and equations with three variables, but we are going to define the transformation for those strings in detail while analyzing the upper bound of overlapped letters.

Before we start our analysis, we define the assignment ψ_s based on the actual superstring s for $\mathcal{S}_{\mathscr{L}}$, which is not necessarily a normed superstring for $\mathcal{S}_{\mathcal{H}}$. By applying the transformations, which we are going to define, the assignment ψ_s will change in dependence to the actual considered superstring.

$$\psi_s(x_i) = \begin{cases} 1 & \text{if the strings in } S(\ell_i^x) \text{ use a 1-alignment in } s \\ 0 & \text{otherwise} \end{cases}$$

Due to the transformations for the strings corresponding to cycle and wheel border equations, the assignment ψ_s is well-defined.

Defining the Assignment for Checker Variables

Let $\mathcal{W}_x$ be a wheel in $\mathscr{L}$ and $M(\mathcal{W}_x)$ its associated perfect matching. Furthermore, let $x_i \oplus x_{i+1} = 0$, $x_{i-1} \oplus x_i = 0$, $x_{j-1} \oplus x_j = 0$, $x_j \oplus x_{j+1} = 0$ and $x_i \oplus x_j = 0$ be equations in $\mathscr{L}$, where $\{i, j\} \in M(\mathcal{W}_x)$ and $i < j$. Let s be a superstring for $\mathcal{S}_{\mathscr{L}}$ such that the strings corresponding to cycle and wheel border equations are using a 0/1-alignment in s. Based on the particular alignments of the strings in s corresponding to ℓ_i^x, ℓ_{i+1}^x, ℓ_j^x and ℓ_{j+1}^x, we are going to define the assignment to the variables x_i and x_j. Furthermore, we analyze the number of overlapped letters that can be achieved by 0/1-alignments and relate them to the number of satisfied equations in $\mathscr{L}$ by ψ_s. Before we start our analysis, we introduce a notation to specify the 0/1-alignments used by the strings corresponding to ℓ_i^x, ℓ_{i+1}^x, ℓ_j^x and ℓ_{j+1}^x in s.

Given a superstring s for $S_{\mathcal{H}}$ and $\{i, j\} \in M(\mathcal{W}_x)$, a *constellation* c is defined by $(X_iX_{i+1}, X_jX_{j+1})^s_{\{i,j\}}$ with $X_i, X_{i+1}, X_j, X_{j+1} \in \{0, 1\}$, where $X_k = b$ if and only if the strings in $S(\ell^x_k)$ use the b-alignment in s for $k \in \{i, i+1, j, j+1\}$. We call a constellation $(X_iX_{i+1}, X_jX_{j+1})^s_{\{i,j\}}$ *inconsistent* if there is a $k \in \{i, j\}$ such that $X_k \neq X_{k+1}$. Otherwise, we refer to c as *consistent.*

Based on a given superstring s and a corresponding constellation, we are going to define ψ_s.

Definition 9.10.2 (Assignment ψ_s to Checker Variables)
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem, $\mathcal{S}_{\mathscr{L}}$ its corresponding instance of the* SHORTEST SUPERSTRING *problem and s a superstring for $\mathcal{S}_{\mathscr{L}}$. Given the constellation $c = (X_iX_{i+1}, X_jX_{j+1})^s_{\{i,j\}}$, we define ψ_s for the variable x_i as follows.*

$$\psi_s(x_i) = \begin{cases} 1 - X_i & \textit{if } \; X_i \oplus X_j = 1 \textit{ and } X_i \neq X_{i+1} \\ X_i & \textit{otherwise} \end{cases}$$

For the variable x_j, we use the following assignment.

$$\psi_s(x_j) = \begin{cases} 1 - X_j & \textit{if } \; X_i \oplus X_j = 1, X_i = X_{i+1} \textit{ and } X_j \neq X_{j+1} \\ X_j & \textit{otherwise} \end{cases}$$

In the following, we are going to analyze the different constellations and discuss the associated cases of the definition of ψ_s. We start with the case when c is consistent and $X_i \oplus X_j = 1$.

Case ($X_i \oplus X_j = 1$ and c is consistent):
There are two constellations, which we have to analyze, namely $(11, 00)^s_{\{i,j\}}$ and $(00, 11)^s_{\{i,j\}}$. Starting with the former constellation, we obtain the scenario displayed below. Since we know that using the most profitable 0/1-alignment of the strings in $S(\ell^x_{\{i,j\}})$ does not increase the length of the superstring, we make use of the 1-alignment and transform the superstring s in the superstring s', which are both displayed below.

$$s = b \triangleright \triangleright \boxed{x_i^{r1}}\, x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}\, \boxed{x_i^{l1}} \triangleleft \triangleleft m \triangleright \triangleright \boxed{x_j^{r0}}\, x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}\, \boxed{x_j^{l0}} \triangleleft \triangleleft e$$

$$\downarrow$$

$$s' = b \triangleright \triangleright \overbrace{\boxed{x_i^{r1}}\, x_i^{l1} \rlap{$\underbrace{\phantom{x_j^{r0}x_j^{l0}x_i^{r1}\boxed{x_i^{l1}}}}_{x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}}$} x_j^{r0}x_j^{l0}}^{x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}} x_i^{r1}\boxed{x_i^{l1}} \triangleleft \triangleleft m \triangleright \triangleright \boxed{x_j^{r0}}\boxed{x_j^{l0}} \triangleleft \triangleleft e$$

Let us derive an upper bound on the number of overlapped letters. More precisely, we are interested in the number of overlapped letters being additional to the overlap of two letters due to the 0/1-alignment. In both cases, either by using the 1-alignment or the 0-alignment of the strings in $S(\ell^x_{\{i,j\}})$, we cannot obtain more than an overlap of two letters. It corresponds to the number of satisfied equations, $x_i \oplus x_{i+1} = 0$ and $x_j \oplus x_{j+1} = 0$, by ψ_s.

In case of the constellation $(00, 11)^s_{\{i,j\}}$, we separate the strings $x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}$ and $x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}$ from the superstring s. Then, we attach the aligned string $x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}$ at the end of the actual solution. The considered situation is displayed below.

$$b \triangleright \triangleright \boxed{x_i^{m0}}\, x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}\, \boxed{x_i^{m0}} \triangleleft \triangleleft m \triangleright \triangleright \boxed{x_j^{m1}}\, x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}\, \boxed{x_j^{m1}} \triangleleft \triangleleft e$$

$$\downarrow$$

$$b \overbrace{\triangleright \triangleright \boxed{x_i^{m0}}}^{\triangleright \triangleright \boxed{x_i^{m0}}} \triangleleft \triangleleft m \overbrace{\triangleright \triangleright \boxed{x_j^{m1}}}^{\triangleright \triangleright \boxed{x_j^{m1}}} \triangleleft \triangleleft e \quad \overbrace{x_i^{r1}x_i^{l1} \rlap{$\underbrace{\phantom{x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}}}_{x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}}$} x_j^{r0}x_j^{l0}}^{x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}} x_i^{r1}x_i^{l1}$$

In this scenario, we obtain an overlap of at most two letters. This corresponds to the number of satisfied equations, namely $x_i \oplus x_{i+1} = 0$ and $x_j \oplus x_{j+1} = 0$.

Case $(X_i \oplus X_j = 0)$:
Let us start with the constellation $(0X_{i+1}, 0X_{j+1})^s_{\{i,j\}}$. In this case, we set $\psi_s(x_i) = 0$ and $\psi_s(x_j) = 0$. Given the strings $\rhd\rhd\boxed{x_i^{m0}}$, $\boxed{X_{i+1}}\lhd\lhd$, $\rhd\rhd\boxed{x_j^{r0}}$ and $\boxed{X_{j+1}}\lhd\lhd$ with $X_{i+1} \in \{x_i^{m0}, x_i^{l1}\}$ and $X_{j+1} \in \{x_j^{m1}, x_j^{l0}\}$, we obtain the following scenario:

$$b \rhd\rhd\boxed{x_i^{m0}}x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}\boxed{X_{i+1}}\lhd\lhd m \rhd\rhd\overbrace{\boxed{x_j^{r0}}x_j^{l0}x_i^{r1}x_i^{l1}}^{x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}}\boxed{X_{j+1}}\lhd\lhd e$$

$$\downarrow$$

$$b \rhd\rhd\boxed{x_i^{m0}}\boxed{X_{i+1}}\lhd\lhd m \rhd\rhd\rlap{$\overbrace{\phantom{\boxed{x_j^{r0}}x_j^{l0}x_i^{r1}x_i^{l1}}}^{x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}}$}\boxed{x_j^{r0}}x_j^{l0}\underbrace{x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}}_{x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}}\boxed{X_{j+1}}\lhd\lhd e$$

The most advantageous 0/1-alignment in this case is the 0-alignment of the strings in $S(\ell^x_{\{i,j\}})$. If $\psi_s(x_i) = \psi_s(x_{i+1}) = 0$ and therefore, $X_{i+1} = x_i^{m0}$ holds, we obtain another overlap of one letter by aligning $\rhd\rhd\boxed{x_i^{m0}}$ with $\boxed{x_i^{m0}}\lhd\lhd$. A similar argument holds for $\psi_s(x_j) = \psi_s(x_{j+1}) = 0$. Notice that the equation $x_i \oplus x_j = 0$ is satisfied by ψ_s. In summary, we state that we obtain an overlap of one additional letter per satisfied equation. Hence, we obtain an overlap of three letters according to the satisfied equations $x_i \oplus x_{i+1} = 0$, $x_i \oplus x_j = 0$ and $x_j \oplus x_{j+1} = 0$.

Let us consider the constellation $(1X_{i+1}, 1X_{j+1})^s_{\{i,j\}}$. Hence, we are given the strings $\rhd\rhd\boxed{x_i^{r1}}$, $\boxed{X_{i+1}}\lhd\lhd$, $\rhd\rhd\boxed{x_j^{m1}}$ and $\boxed{X_{j+1}}\lhd\lhd$ with $X_{i+1} \in \{x_i^{m0}, x_i^{l1}\}$ and $X_{j+1} \in \{x_j^{m1}, x_j^{l0}\}$. We obtain the scenario displayed below.

$$b \triangleright \triangleright \boxed{x_i^{r1}}\, x_i^{r1} x_i^{l1} x_j^{r0} x_j^{l0} \boxed{X_{i+1}} \triangleleft \triangleleft m \triangleright \triangleright \boxed{x_j^{m1}}\, x_j^{r0} x_j^{l0} x_i^{r1} x_i^{l1} \boxed{X_{j+1}} \triangleleft \triangleleft e$$

$$\downarrow$$

$$b \triangleright \triangleright \overbrace{\boxed{x_i^{r1}}\, x_i^{l1} \rlap{$\underbrace{x_j^{r0} x_j^{l0} x_i^{r1} x_i^{l1}}_{x_j^{r0} x_j^{l0} x_i^{r1} x_i^{l1}}$}\phantom{x_j^{r0} x_j^{l0}}}^{x_i^{r1} x_i^{l1} x_j^{r0} x_j^{l0}}\phantom{x_i^{r1} x_i^{l1}} \boxed{X_{i+1}} \triangleleft \triangleleft m \triangleright \triangleright \boxed{x_j^{m1}} \boxed{X_{j+1}} \triangleleft \triangleleft e$$

In this case, we use the 1-alignment of the strings in $S(\ell^x_{\{i,j\}})$. If $\psi_s(x_i) = \psi_s(x_{i+1}) = 1$ holds, which means $X_{i+1} = x_i^{l1}$, we obtain another overlap of one letter by aligning

$$\triangleright \triangleright \overbrace{\boxed{x_i^{r1}}\, x_i^{l1} \rlap{$\underbrace{x_j^{r0} x_j^{l0} x_i^{r1} x_i^{l1}}_{x_j^{r0} x_j^{l0} x_i^{r1} x_i^{l1}}$}\phantom{x_j^{r0} x_j^{l0}}}^{x_i^{r1} x_i^{l1} x_j^{r0} x_j^{l0}}\phantom{x_i^{r1} x_i^{l1}} \quad \text{with} \quad \boxed{x_i^{l1}} \triangleleft \triangleleft .$$

In case of $\psi_s(x_j) = \psi_s(x_{j+1}) = 1$, we may apply a similar argument. Notice that the equation $x_i \oplus x_j = 0$ is satisfied by ψ_s. In summary, we state that we obtain an overlap of one additional letter per satisfied equation. Hence, we obtain an overlap of three letters according to the satisfied equations $x_i \oplus x_{i+1} = 0$, $x_i \oplus x_j = 0$ and $x_j \oplus x_{j+1} = 0$.

Case ($X_i \oplus X_j = 1$ and $X_i \neq X_{i+1}$):
Let us begin with the constellation $(10, 0X_{j+1})^s_{\{i,j\}}$. We consider the scenario depicted below, in which we are given the strings $\triangleright \triangleright \boxed{x_i^{r1}}$, $\boxed{x_i^{m0}} \triangleleft \triangleleft$, $\triangleright \triangleright \boxed{x_j^{l0}}$ and $\boxed{X_{j+1}} \triangleleft \triangleleft$ with $X_{j+1} \in \{x_j^{l0}, x_j^{m1}\}$.

$$b \rhd \rhd \boxed{x_i^{r1}} x_i^{r1} x_i^{l1} x_j^{r0} x_j^{l0} \boxed{x_i^{m0}} \lhd \lhd m \rhd \rhd \boxed{x_j^{r0}} x_j^{r0} x_j^{l0} x_i^{r1} x_i^{l1} \boxed{X_{j+1}} \lhd \lhd e$$

$$\downarrow$$

$$\overbrace{b \rhd \rhd \boxed{x_i^{m0}}}^{\rhd \rhd \boxed{x_i^{m0}}} \lhd \lhd m \rhd \rhd \overbrace{\boxed{x_j^{r0}} x_j^{l0} \mathrlap{\underbrace{\phantom{x_i^{r1} x_i^{l1} x_j^{r0} x_j^{l0}}}_{x_i^{r1} x_i^{l1} x_j^{r0} x_j^{l0}}} x_i^{r1} x_i^{l1}}^{x_j^{r0} x_j^{l0} x_i^{r1} x_i^{l1}} x_j^{r0} x_j^{l0} \boxed{X_{j+1}} \lhd \lhd e$$

Instead of using the 1-alignment of the strings in $S(\ell_i^x)$, we rather switch to the 0-alignment, i.e. we obtain the string $\rhd \rhd \boxed{x_i^{m0}}$ and define $\psi(x_i) = 0$. It results directly in gaining two additional satisfied equations and an overlap of one additional letter. As a matter of fact, we might lose an overlap of one letter, because the string $\rhd \rhd \boxed{x_1^{m}}$ could have been aligned from the right side with another string. Furthermore, the equation $x_{i-1} \oplus x_i = 0$ could be unsatisfied. But all in all, we obtain at least $2 - 1$ additional satisfied equations by switching the value without increasing the length the superstring. Notice that we may achieve an additional overlap of one letter if $X_{j+1} = x_j^{l0}$ holds, which means that ψ_s satisfies the equation $x_j \oplus x_{j+1} = 0$.

The next constellation, which we are going to analyze, is $(01, 1X_{j+1})_{\{i,j\}}^s$. Hence, we are given the strings $\rhd \rhd \boxed{x_i^{m0}}$, $\boxed{x_i^{l1}} \lhd \lhd$, $\rhd \rhd \boxed{x_j^{m1}}$ and $\boxed{X_{j+1}} \lhd \lhd$, with $X_{j+1} \in \{x_j^{l0}, x_j^{m1}\}$. The situation is displayed below.

$$b \rhd \rhd \boxed{x_i^{m0}} x_i^{r1} x_i^{l1} x_j^{r0} x_j^{l0} \boxed{x_i^{l1}} \lhd \lhd m \rhd \rhd \boxed{x_j^{m1}} x_j^{r0} x_j^{l0} x_i^{r1} x_i^{l1} \boxed{X_{j+1}} \lhd \lhd e$$

$$\downarrow$$

$$b \rhd \rhd \overbrace{\boxed{x_i^{r1}} x_i^{l1} \mathrlap{\underbrace{\phantom{x_j^{r0} x_j^{l0} x_i^{r1} \boxed{x_i^{l1}}}}_{x_j^{r0} x_j^{l0} x_i^{r1} x_i^{l1}}} x_j^{r0} x_j^{l0}}^{x_i^{r1} x_i^{l1} x_j^{r0} x_j^{l0}} x_i^{r1} \boxed{x_i^{l1}} \lhd \lhd m \rhd \rhd \boxed{x_j^{m1}} \boxed{X_{j+1}} \lhd \lhd e$$

We obtain a similar situation, in which we switch $\triangleright\triangleright\boxed{x_i^{m0}}$ to $\triangleright\triangleright\boxed{x_i^{r1}}$. Accordingly, we define $\psi_s(x_i) = 1$. We obtain at least one additional satisfied equation by switching the value without increasing the length of the superstring. Notice that we may achieve an additional overlap of one letter if $X_{j+1} = x_j^{m1}$ holds. It corresponds to the satisfied equation $x_j \oplus x_{j+1} = 0$.

Case ($X_i \oplus X_j = 1$, $X_j \neq X_{j+1}$ and $X_i = X_{i+1}$):
Starting our analysis with the constellation $(00, 10)^s_{\{i,j\}}$, we obtain the following scenario.

$$b \triangleright \triangleright \boxed{x_i^{m0}} x_i^{r1} x_i^{l1} x_j^{r0} x_j^{l0} \boxed{x_i^{m0}} \triangleleft \triangleleft m \triangleright \triangleright \boxed{x_j^{m1}} x_j^{r0} x_j^{l0} x_i^{r1} x_i^{l1} \boxed{x_j^{l0}} \triangleleft \triangleleft e$$

$$\downarrow$$

$$\overbrace{b \triangleright \triangleright \boxed{x_i^{m0}}}^{\triangleright\triangleright\boxed{x_i^{m0}}} \triangleleft \triangleleft m \triangleright \triangleright \overbrace{\boxed{x_j^{r0}}\, x_j^{l0} \rlap{$\underbrace{\phantom{x_i^{r1} x_i^{l1} x_j^{r0} \boxed{x_j^{l0}}}}_{x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}}$} x_i^{r1} x_i^{l1}}^{x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}} x_j^{r0} \boxed{x_j^{l0}} \triangleleft \triangleleft e$$

In this case, we argue that we switch the string $\triangleright\triangleright\boxed{x_j^{m1}}$ to $\triangleright\triangleright\boxed{x_j^{r0}}$. This means that we set $\psi_s(x_j) = 0$. This transformation yields an overlap of at least the same number of letters since we could lose an overlap of one letter from the left side. On the other hand, we align the string

$$\triangleright \triangleright \boxed{x_j^{r0}} \quad \text{with} \quad \overbrace{x_j^{r0} x_j^{l0} \rlap{$\underbrace{\phantom{x_i^{r1} x_i^{l1} x_j^{r0} \boxed{x_j^{l0}}}}_{x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}}$} x_i^{r1} x_i^{l1}}^{x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}} x_j^{r0} \boxed{x_j^{l0}} \triangleleft \triangleleft$$

from the right side by one letter. Notice that we gain at least one additional satisfied equation.

The last constellation, we are going to analyze, is $(11, 01)^s_{\{i,j\}}$. The corresponding situation is displayed below.

$$b \rhd \rhd \boxed{x_i^{r1}}\, x_i^{r1} x_i^{l1} x_j^{r0} x_j^{l0} \boxed{x_i^{l1}} \lhd \lhd m \rhd \rhd \boxed{x_j^{r0}}\, x_j^{r0} x_j^{l0} x_i^{r1} x_i^{l1} \boxed{x_j^{m1}} \lhd \lhd e$$

$$\downarrow$$

$$b \rhd \overbrace{\rhd \boxed{x_i^{r1}}\, x_i^{l1} \rlap{\underbrace{x_j^{r0} x_j^{l0} x_i^{r1} \boxed{x_i^{l1}}}_{x_j^{r0}x_j^{l0}x_i^{r1}x_i^{l1}}}\phantom{x_j^{r0} x_j^{l0}}}^{x_i^{r1}x_i^{l1}x_j^{r0}x_j^{l0}} \phantom{x_i^{r1} \boxed{x_i^{l1}}} \lhd \lhd m \rhd \overbrace{\rhd \boxed{x_j^{m1}}}^{\rhd\rhd\boxed{x_j^{m1}}} \lhd \lhd e$$

In this case, we switch the string $\rhd \rhd \boxed{x_j^{r0}}$ to $\rhd \rhd \boxed{x_j^{m1}}$. Similarly to the former case, this transformation does not increase the length of the superstring. By defining $\psi_s(x_j) = 1$, we achieve at least one more satisfied equation.

As for the next step, we are going to define the assignment ψ_s for contact variables. After that, we will analyze the relation between the number of satisfied equation by ψ_s and the overlap of the associated strings.

Defining the Assignment for Contact Variables

Let $\ell_j^3 \equiv x \oplus y \oplus z = 0$ be an equation with exactly three variables in $\mathscr{L}$. Given a 0/1-alignment of the strings corresponding to the equations $x_{j_1-1} \oplus x = 0$, $x \oplus x_{j_1+1} = 0$, $y_{j_2-1} \oplus y = 0$, $y \oplus y_{j_2+1} = 0$, $z_{j_3-1} \oplus z = 0$, and $z \oplus z_{j_3+1} = 0$, we are going to define an assignment based on the underlying 0/1-alignments. As before, we introduce a notation in order to specify the 0/1-alignments used in the underlying superstring.

For a given superstring s for $\mathcal{S}_{\mathscr{L}}$ and equation $\ell_j^3 \equiv x \oplus y \oplus z = 0$, we define a *constellation* $c = (X_1X_2, Y_1Y_2, Z_1Z_2)_j^s$ with $X_1, X_2, Y_1, Y_2, Z_1, Z_2 \in \{0,1\}$, where $X_1 = b_1$, $X_2 = b_2$, $Y_1 = b_3$, $Y_2 = b_4$, $Z_1 = b_5$ and $Z_2 = b_6$ if and only if the strings corresponding to $x_{j_1-1} \oplus x = 0$, $x \oplus x_{j_1+1} = 0$, $y_{j_2-1} \oplus y = 0$,

$y \oplus y_{j_2+1} = 0$, $z_{j_3-1} \oplus z = 0$ and $z \oplus z_{j_3+1} = 0$ are using a b_1, b_2, b_3, b_4, b_5 and b_6-alignment in s, respectively. We call a constellation inconsistent if there is an $A \in \{X, Y, Z\}$ with $A_1 \neq A_2$. Otherwise, we refer to c as consistent.

Based on a constellation for a given superstring s and an equation ℓ_j^3 with three variables, we are going to define the assignment ψ_s for the variables in ℓ_j^3.

Definition 9.10.3 (Assignment ψ_s to Contact Variables)
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem, $\mathcal{S}_{\mathscr{L}}$ its corresponding instance of the* SHORTEST SUPERSTRING *problem, s a superstring for $\mathcal{S}_{\mathscr{L}}$ and $\ell_j^3 \equiv x \oplus y \oplus z = 0$ an equation with three variables in $\mathscr{L}$. For the associated constellation $c = (X_1X_2, Y_1Y_2, Z_1Z_2)_j^s$, we define ψ_s for the variable x as follows.*

$$\psi_s(x) = \begin{cases} 1 - X_1 & \text{if } X_1 \oplus Y_1 \oplus Z_1 = 1 \text{ and } X_1 \neq X_2 \\ X_1 & \text{else} \end{cases}$$

For the variable y, we use the following assignment.

$$\psi_s(y) = \begin{cases} 1 - Y_1 & \text{if } X_1 \oplus Y_1 \oplus Z_1 = 1, X_1 = X_2 \text{ and } Y_1 \neq Y_2 \\ Y_1 & \text{else} \end{cases}$$

For the variable z, we define ψ_s as follows.

$$\psi_s(z) = \begin{cases} 1 - Z_1 & \text{if } X_1 \oplus Y_1 \oplus Z_1 = 1, X_1 = X_2, Y_1 = Y_2 \text{ and } Z_1 \neq Z_2 \\ Z_1 & \text{else} \end{cases}$$

In the following, we will analyze the relation of satisfied equations by ψ_s and the number of overlapped letters in the given superstring. In particular, we consider the following three cases and define the corresponding transformations.

- $X_1 \oplus Y_1 \oplus Z_1 = 1$ and c is consistent.
- $X_1 \oplus Y_1 \oplus Z_1 = 0$ and c is inconsistent.

- $X_1 \oplus Y_1 \oplus Z_1 = 1$ and c is inconsistent.

Let us begin with the first case.

Case ($X_1 \oplus Y_1 \oplus Z_1 = 1$ and c is consistent):
In this case, we start with the constellation $(11, 11, 11)^s_j$. We display the considered situation below.

$$b \rhd \rhd \boxed{x^{r1}}\boxed{x^{l1}} \lhd \lhd m_1 \rhd \rhd \boxed{y^{r1}}\, y^{r1}A_jy^{l1}\boxed{y^{l1}} \lhd \lhd m_2 \rhd \rhd \boxed{z^{r1}}\, z^{r1}B_jz^{l1}\boxed{z^{l1}} \lhd \lhd e$$

$$\downarrow$$

$$b \rhd \rhd \boxed{x^{r1}}\boxed{x^{l1}} \lhd \lhd m_1 \rhd \rhd \underbrace{\boxed{y^{r1}}A_j\boxed{y^{l1}}}_{y^{r1}Ay^{l1}} \lhd \lhd m_2 \rhd \rhd \underbrace{\boxed{z^{r1}}B_j\boxed{z^{l1}}}_{z^{r1}Bz^{l1}} \lhd \lhd e$$

According to the definition of ψ_s, we have $\psi_s(x) = \psi_s(y) = \psi_s(z) = 1$. Notice that the equation $x \oplus y \oplus z = 0$ is unsatisfied. On the other hand, the assignment ψ_s satisfies the equations $x \oplus x_{j_1+1} = 0$, $y \oplus y_{j_2+1} = 0$ and $z \oplus z_{j_3+1} = 0$.
We note that a string corresponding to $S^A(\ell^3_j)$ or $S^B(\ell^3_j)$ using a 0/1-alignment can have an overlap of at most one letter from the right side as well as from the left side. Therefore, one possibility, maximizing the number of overlapped letters given the constellation $(11, 11, 11)^s_j$, is to align the string $y^{r1}A_jy^{l1}$ with $\rhd \rhd \boxed{y^{r1}}$ and $\boxed{y^{l1}} \lhd \lhd$ each by one letter. We may argue similarly for the string $z^{r1}B_jz^{l1}$. Consequently, we conclude that the number of overlapped letters is bounded from above by four.

In case of $X_1 + Y_1 + Z_1 = 1$, we analyze exemplary the constellation $(00, 00, 11)^s_j$. We set $\psi_s(z) = 1$, $\psi_s(x) = 0$ and $\psi_s(y) = 0$. This situation is displayed below.

$$b \triangleright \triangleright \boxed{x^{m0}} \boxed{x^{m0}} \triangleleft \triangleleft m' \triangleright \triangleright \boxed{y^{m0}} y^{r1} A_j y^{l1} \boxed{y^{m0}} \triangleleft \triangleleft m \triangleright \triangleright \boxed{z^{r1}} z^{r1} B_j z^{l1} \boxed{z^{l1}} \triangleleft \triangleleft e$$

$$\downarrow$$

$$b \underbrace{\triangleright \triangleright \boxed{x^{m0}}}_{\triangleright \triangleright \boxed{x^{m0}}} \triangleleft \triangleleft m' \underbrace{\triangleright \triangleright \boxed{y^{m0}}}_{\triangleright \triangleright \boxed{y^{m0}}} \triangleleft \triangleleft m \triangleright \triangleright \underbrace{\boxed{z^{r1}} B_j \boxed{z^{l1}}}_{z^{r1} B_j z^{l1}} \triangleleft \triangleleft e \; y^{r1} A_j y^{l1}$$

Due to the z^1-alignment of the strings in $S^B(\ell_j^3)$, we obtain an overlap of two letters. Additionally, we align the string $\triangleright \triangleright \boxed{x^{m0}}$ from the left with $\boxed{x^{m0}} \triangleleft \triangleleft$. The same holds for $\triangleright \triangleright \boxed{y^{m0}}$ and $\boxed{y^{m0}} \triangleleft \triangleleft$. Notice that it is not more advantageous to align the string $x^{m0} B_j C_j$ with $\triangleright \triangleright \boxed{x^{m0}}$ since we lose the overlap of one letter with $\boxed{x^{m0}} \triangleleft \triangleleft$. Hence, we are able to get an overlap of at most four letters, which corresponds to the satisfied equations $x \oplus x_{j_1+1} = 0$, $y \oplus y_{j_2+1} = 0$ and $z \oplus z_{j_3+1} = 0$.

Case ($X_1 \oplus Y_1 \oplus Z_1 = 0$ and c is inconsistent):
First, we concentrate on constellations with the property $X_1 + Y_1 + Z_1 = 2$. Exemplary, we analyze the constellation $(0X_2, 1Y_2, 1Z_2)_j^s$ displayed below.

$$b \triangleright \triangleright \boxed{x^{m0}} \boxed{X_2} \triangleleft \triangleleft m_1 \triangleright \triangleright \boxed{y^{r1}} y^{r1} A_j y^{l1} \boxed{Y_2} \triangleleft \triangleleft m_2 \triangleright \triangleright \boxed{z^{r1}} z^{r1} B_j z^{l1} \boxed{Z_2} \triangleleft \triangleleft e$$

$$\downarrow$$

$$b \triangleright \triangleright \boxed{x^{m0}} \boxed{X_2} \triangleleft \triangleleft m_1 \triangleright \triangleright \underbrace{\boxed{y^{r1}} A_j y^{l1}}_{y^{r1} A_j y^{l1}} \boxed{Y_2} \triangleleft \triangleleft m_2 \triangleright \triangleright \underbrace{\boxed{z^{r1}} B_j \, z^{l1}}_{z^{r1} B_j z^{l1}} \boxed{Z_2} \triangleleft \triangleleft e$$

The strings $\triangleright \triangleright \boxed{y^{r1}}$ and $\triangleright \triangleright \boxed{z^{r1}}$ can be used to align from the right side with $z^{r1} B_j z^{l1}$ and $y^{r1} A_j y^{l1}$, respectively. It yields an overlap of two letters. If the corresponding equations with two variables are satisfied, which means $X_2 = x^{m0}$, $Y_2 = y^{l1}$ and $Z_2 = z^{l1}$, we gain an additional overlap of one letter per satisfied equation. Notice that using the x^0-alignment of $S(\ell_j^3)$ does not yield more overlapped letters. In summary, it is possible to attain an overlap of at most five letters, which corresponds to the constellation $(00, 11, 11)_j^s$. An analogous argumentation holds for the constellations $(1X_2, 1Y_2, 0Z_2)_j^s$ and $(1X_2, 0Y_2, 1Z_2)_j^s$.

Next, we discuss constellations with the property $X_1 + Y_1 + Z_1 = 0$. For this reason, we consider the constellation $(0X_2, 0Y_2, 0Z_2)_j^s$.

$$b \rhd \rhd \boxed{x^{m0}} \boxed{X_2} \lhd \lhd m_1 \rhd \rhd \boxed{y^{m0}} y^{r1} B_j y^{r1} \boxed{Y_2} \lhd \lhd m_2 \rhd \rhd \boxed{z^{m0}} y^{r1} B_j y^{r1} \boxed{Z_2} \lhd \lhd e$$

$$\downarrow$$

$$b \rhd \rhd \underbrace{\boxed{x^{m0}} A_j \rlap{\overbrace{\phantom{C_j B_j x^{m0}}}^{C_j B_j x^{m0}}} C_j}_{x^{m0} A_j C_j} B_j x^{m0} \boxed{X_2} \lhd \lhd m_1 \rhd \rhd \boxed{y^{m0}} \boxed{Y_2} \lhd \lhd m_2 \rhd \rhd \boxed{z^{m0}} \boxed{Z_2} \lhd \lhd e$$

Recall that $x^{m0} C_j x^{m0}$ denotes the x^0-alignment of $S(\ell_j^3)$. This string can be aligned from the left with $\rhd \rhd x^{m0}$. If $X_2 = x^{m0}$ holds, we achieve another overlap of one letter. Furthermore, the string $\rhd \rhd \boxed{y^{m0}}$ can be aligned from the right with $\boxed{Y_2} \lhd \lhd$ if and only if $Y_2 = y^{m0}$ holds. A similar argumentation can be applied to the strings $\rhd \rhd \boxed{z^{m0}}$ and $\boxed{Z_2} \lhd \lhd$. Finally, we note that we cannot benefit by aligning the string $\boxed{y^{l1}} \lhd \lhd$ with $y^{r1} A_j y^{l1}$. Consequently, we see that using the string $x^{m0} C_j x^{m0}$ is generally more profitable. All in all, we gain an additional overlap of one letter for satisfying $x \oplus y \oplus z = 0$ and another overlap of one letter if the equation with two variables corresponding to the considered variable is satisfied.

Case ($X_1 \oplus Y_1 \oplus Z_1 = 1$ and c is inconsistent):

Let us start with constellations satisfying $X_1 + Y_1 + Z_1 = 3$. Exemplary, we analyze the constellation $(10, 1Y_2, 1Z_2)_j^s$. According to the definition of ψ_s, we set $\psi_s(x) = 1 - X_1$, $\psi_s(y) = 1$ and $\psi_s(z) = 1$. Notice that ψ_s satisfies the equation $x \oplus y \oplus z = 0$. By switching the value $\psi_s(x)$ from X_1 to $1 - X_1$, the equation $x_{j_1-1} \oplus x = 0$ could become unsatisfied. Furthermore, we could lose an overlap of one letter by flipping the 1-alignment of the strings corresponding to $x_{j_1-1} \oplus x = 0$ to the 0-alignment. On the other hand, we gain an overlap of one letter by aligning the string $\triangleright \triangleright \boxed{x^{m0}}$ from the right side with $\boxed{x^{m0}} \triangleleft \triangleleft$. This transformation yields at least one more satisfied equation. In addition, the strings $y^{r1}A_jy^{l1}$ and $z^{r1}B_jz^{l1}$ can be aligned by one letter with $\triangleright \triangleright \boxed{y^{r1}}$ and $\triangleright \triangleright \boxed{z^{r1}}$, respectively. If $Z_2 = z^{l1}$ and $Y_2 = y^{l1}$ holds, we achieve another overlap of one letter in each case (see below).

$$\begin{gathered}
b \triangleright \triangleright \boxed{x^{r1}}\,\boxed{x^{m0}} \triangleleft \triangleleft m_1 \triangleright \triangleright \boxed{y^{r1}} y^{r1}A_jy^{l1} \boxed{Y_2} \triangleleft \triangleleft m_2 \triangleright \triangleright \boxed{z^{r1}} z^{r1}B_jz^{l1} \boxed{Z_2} \triangleleft \triangleleft e \\
\downarrow \\
\underbrace{b \triangleright \triangleright \boxed{x^{m0}}}_{\triangleright \triangleright \boxed{x^{m0}}} \triangleleft \triangleleft \ \triangleright \triangleright \underbrace{\boxed{y^{r1}} A_jy^{l1}}_{y^{r1}A_jy^{l1}} \boxed{Y_2} \triangleleft \triangleleft \ \triangleright \triangleright \underbrace{\boxed{z^{r1}} B_jz^{l1}}_{z^{r1}B_jz^{l1}} \boxed{Z_2} \triangleleft \triangleleft
\end{gathered}$$

The other constellations satisfying $X_1 + Y_1 + Z_1 = 3$ can be analyzed analogously.

The remaining constellations $(X_1X_2, Y_1Y_2, Z_1Z_2)^s_j$ to be discussed satisfy $X_1 + Y_1 + Z_1 = 1$ and are inconsistent. Exemplary, we analyze the constellation $(01, 0Y_2, 1Z_2)^s_j$. For $(01, 0Y_2, 1Z_2)^s_j$, we set $\psi_s(x) = 1 - X_1$, $\psi_s(y) = Y_1$ and $\psi_s(z) = Z_1$. The scenario is displayed below.

$$\begin{gathered}
b \triangleright \triangleright \boxed{x^{m0}} x^{r1}A_jx^{l1} \boxed{x^{l1}} \triangleleft \triangleleft m_1 \triangleright \triangleright \boxed{y^{m0}}\,\boxed{Y_2} \triangleleft \triangleleft m_2 \triangleright \triangleright \underbrace{\boxed{z^{r1}} B_jz^{l1}}_{z^{r1}B_jz^{l1}} \boxed{Z_2} \triangleleft \triangleleft e \\
\downarrow \\
b \triangleright \triangleright \underbrace{\boxed{x^{r1}} A_j \boxed{x^{l1}}}_{x^{r1}A_jx^{l1}} \triangleleft \triangleleft m_1 \triangleright \triangleright \boxed{y^{m0}}\,\boxed{Y_2} \triangleleft \triangleleft m_2 \triangleright \triangleright \underbrace{\boxed{z^{r1}} B_jz^{l1}}_{z^{r1}B_jz^{l1}} \boxed{Z_2} \triangleleft \triangleleft e
\end{gathered}$$

By flipping the 0-alignment of the strings corresponding to $x_{j_1-1} \oplus x = 0$ to the 1-alignment, we can overlap $x^{r1}A_jx^{l1}$ from the left side with $\triangleright \triangleright \boxed{x^{r1}}$ and with $\boxed{x^{l1}} \triangleleft \triangleleft$ from the right side. This transformation achieves an overlap of at most one more letter. Moreover, we obtain at least one more satisfied equation. If $Z_2 = z^{l1}$ and $Y_2 = y^{m0}$ holds, it yields an overlap of three additional letters, which corresponds to the constellation $(11, 00, 11)^s_j$.

Finally, we note that the strings corresponding to equations of the form $x \oplus y \oplus z = 1$ can be discussed analogously. In summary, we obtain the following statement.

Lemma 9.10.1
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem with n wheels, m_2 equations with two variables and m_3 equations with three variables. Given a superstring s for $\mathcal{S}_{\mathscr{L}}$ with length $|s| = 7 \cdot n + 5 \cdot m_2 + 22 \cdot m_3 + u$, it is possible to transform s in polynomial time into a superstring t without increasing the length such that the associated assignment ψ_t leaves at most u equations in $\mathscr{L}$ unsatisfied.*

9.10.1 The Proof of Theorem 9.6.1

Given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem with n wheels, m_2 equations with two variables and m_3 equations with exactly three variables with the properties described in Theorem 4.9.1, we construct in polynomial time an instance $\mathcal{S}_{\mathscr{L}}$ of the SHORTEST SUPERSTRING problem with the properties described in section 9.8.

Let ϕ be an assignment to the variables of $\mathscr{L}$ leaving at most u equations in $\mathscr{L}$ unsatisfied. According to Lemma 9.9.1, it is possible to construct in polynomial time a superstring s_ϕ for $\mathcal{S}_{\mathscr{L}}$ with length at most

$$|s_\phi| \;\le\; 7 \cdot n + 5 \cdot m_2 + 22 \cdot m_3 + u$$

since the length of the superstring increases by at most one letter for every unsatisfied equation of the assignment. Regarding the compression measure, we obtain the following.

$$\begin{aligned} comp(\mathcal{S}_{\mathscr{L}}, s_\phi) &\geq \sum_{s\in\mathcal{S}_{\mathscr{L}}} |s| \;-\; (7\cdot n + 5\cdot m_2 + 22\cdot m_3 + u) \\ &= (4+8)n + 8\cdot m_2 + 36\cdot m_3 \;-\; (7\cdot n + 5\cdot m_2 + 22\cdot m_3 + u) \\ &= 5n + 3m_2 + 14m_3 - u \end{aligned}$$

On the other hand, let s be a superstring for $S_{\mathscr{L}}$ with length

$$|s| \;=\; 5m_2 + 22m_3 + u + 7n,$$

it is possible to construct in polynomial time a normed superstring s' without increasing the length by applying the transformations defined in section 9.10. According to Lemma 9.10.1, it enables us to define an assignment ψ_s to the variables of $\mathscr{L}$ that leaves at most u equations in $\mathscr{L}$ unsatisfied. A similar argumentation leads to the conclusion that given a superstring s for $S_{\mathscr{L}}$ with compression

$$comp(\mathcal{S}_{\mathscr{L}}, s_\phi) = 5n + 3m_2 + 14m_3 - u,$$

it is possible to construct in polynomial time an assignment to the variables in $\mathscr{L}$ that leaves at most u equations unsatisfied. ■

In the next section, we are going to describe smaller gadgets for equations with three variables implying an improved explicit lower bound and give the proof of Theorem 9.5.1.

9.11 An Improved Reduction

Given an equation with three variables $g_c^3 \equiv x \oplus y \oplus z = 0$, we introduce the sets $S^\alpha(g_j^3)$ and $S^\beta(g_j^3)$ including the following strings.

$$\begin{array}{llll} x^{r1\alpha}x^{l1}y^{r1}y^{l1}, & y^{r1}y^{l1}x^{m0}C_j, & x^{m0}C_jx^{r1\alpha}x^{l1} & \in S^\alpha(g_j^3) \\ x^{r1\beta}x^{l1}z^{r1}z^{l1}, & z^{r1}z^{l1}C_jx^{m0}, & C_jx^{m0}x^{r1\beta}x^{l1} & \in S^\beta(g_j^3) \end{array}$$

In addition, we introduce new strings for the equation $x_{i-1} \oplus x = 0$. On the other hand, the strings corresponding to $x \oplus x_{i+1} = 0$, $y_{i-1} \oplus y = 0$, $y \oplus y_{i+1} = 0$,

$z_{i-1} \oplus z = 0$ and $z \oplus z_{i+1} = 0$ remain the same. Let us define the strings for $x_{i-1} \oplus x = 0$:

$$x_{i-1}^{l1} x^{r1\beta} x_{i-1}^{l1} x^{r1\alpha} \quad x_{i-1}^{l1} x^{r1\alpha} x_{i-1}^{m0} x^{m0} \quad x_{i-1}^{m0} x^{m0} x_{i-1}^{l1} x^{r1\beta}$$

These three strings can be aligned each by two letters in a cyclic fashion. Accordingly, we obtain three combinations that can be used to overlap with other strings by one letter from the left side as well as from the right side. Note that we have only two combinations if we consider only the left most position of the combined strings. For example, the combination

$$x_{i-1}^{l1} x^{r1\beta} x_{i-1}^{l1} x^{r1\alpha} x_{i-1}^{m0} x^{m0} x_{i-1}^{l1} x^{r1\beta}$$

can be used to overlap from the right side with strings in $S^{\beta}(g_j^3)$, whereas

$$x_{i-1}^{l1} x^{r1\alpha} x_{i-1}^{m0} x^{m0} x_{i-1}^{l1} x^{r1\beta} x_{i-1}^{l1} x^{r1\alpha}$$

can be aligned with strings contained in $S^{\alpha}(g_j^3)$. Therefore, we may apply the same arguments as in the proof of Theorem 9.5.1. The strings corresponding to equations of the form $x \oplus y \oplus z = 1$ can be constructed analogously.

9.11.1 The Proof of Theorem 9.5.1

Given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem with n wheels, m_2 equations with two variables and m_3 equations with exactly three variables with the properties described in Theorem 4.9.1, we construct in polynomial time an instance $\mathcal{S}_{\mathscr{L}}$ of the SHORTEST SUPERSTRING problem. Let ϕ be an assignment to the variables of $\mathscr{L}$ which leaves at most u equations unsatisfied. Then, it is possible to construct a superstring s_ϕ with length

$$|s_\phi| \;\le\; 7 \cdot n + 5 \cdot m_2 + 16 \cdot m_3 + u,$$

since the length of the superstring increases by at most one letter for every unsatisfied equation of the assignment. Regarding the compression measure, we obtain the following.

$$\begin{aligned} comp(\mathcal{S}_{\mathscr{L}}, s_\phi) &\ge \sum_{s \in \mathcal{S}_{\mathcal{H}}} |s| - (7 \cdot n + 5 \cdot m_2 + 16 \cdot m_3 + u) \\ &= (4+8)n + 8 \cdot m_2 + 28 \cdot m_3 - (7 \cdot n + 5 \cdot m_2 + 16 \cdot m_3 + u) \\ &= 5n + 3m_2 + 12m_3 - u \end{aligned}$$

On the other hand, given an superstring s for $S_{\mathscr{L}}$ with length

$$|s| = 5m_2 + 16m_3 + u + 7n,$$

we can construct in polynomial time a normed superstring s' without increasing the length of it. The corresponding assignment $\psi_{s'}$ to the variables of $\mathscr{L}$ leaves at most u equations in $\mathscr{L}$ unsatisfied. A similar argumentation leads to the conclusion that given a superstring s for $S_{\mathscr{L}}$ with compression

$$comp(\mathcal{S}_{\mathscr{L}}, s_{\phi}) = 5n + 3m_2 + 12m_3 - u,$$

we construct in polynomial time an assignment to the variables in $\mathscr{L}$ such that at most u equations are unsatisfied. ■

9.12 Bibliographic Notes

The presented material in this chapter is based on the paper [KS11], in which the proofs of Theorem 9.5.1 and 9.6.1 appeared.

CHAPTER 10

Traveling Salesman Problems

In this chapter, we investigate the approximation hardness of symmetric and asymmetric TSP problems with bounded metrics. By extending our reduction method from Chapter 9, we give improved explicit approximation lower bounds for the (1, 2)-ATSP problem, (1, 2)-TSP problem, (1, 4)-ATSP problem, (1, 4)-TSP problem and the MAX-(0, 1)-ATSP. In particular, we prove the best known approximation lower bound for TSP with bounded metrics for the metric bound equal to 4.

We study the approximation complexity of the (1, 2)-TSP problem and the GRAPHIC-TSP problem restricted to subcubic and cubic instances, and obtain new inapproximability bounds for those problems.

By constructing a new bounded degree wheel amplifier and exploiting the special properties of a well-suited bounded occurrence CSP, we obtain the best up to now inapproximability thresholds for the general METRIC and ASYMMETRIC TSP problem.

10.1 Introduction

The Metric Traveling Salesman (TSP) problem is the following problem: Given a metric space (V, d) and the objective is to construct a shortest tour visiting each element in V exactly once.

The TSP problem in metric spaces is one of the most fundamental **NP**-hard optimization problems. The decision version of this problem was shown early to be **NP**-complete by Karp [K75]. Christofides [C76] gave an efficient algorithm approximating the TSP problem with an approximation ratio 3/2, that is, an algorithm that produces in polynomial time a tour with length being at most a factor 3/2 from the optimum.

As for lower bounds, a reduction due to Papadimitriou and Yannakakis [PY93] and the PCP Theorem [AS98, ALM⁺98] together imply that there exists some constant, not greater than $1 + 10^{-6}$, such that it is **NP**-hard to approximate the TSP problem with distances either one or two. For discussion of bounded metrics TSP, see also [T00]. This hardness result was firstly improved by Engebretsen [E03], who proved that it is **NP**-hard to approximate the TSP problem restricted to distances one and two to within any constant less than 5381/5380. Böckenhauer and Seibert [BS00] studied the TSP problem with distances one, two and three. They obtained an approximation lower bound of 3813/3812 for this restricted version of the problem. After that, Papadimitriou and Vempala [PV06] proved that approximating the general problem to within any constant approximation ratio less than 220/219 is **NP**-hard. Only recently, this lower bound was improved by Lampis [L12]. In particular, he gave an approximation lower bound for the problem of 185/184.

In this chapter, we prove that it is **NP**-hard to approximate the TSP problem to within any constant approximation ratio less than 123/122.

The Asymmetric Traveling Salesperson (ATSP) Problem

In the ATSP problem, we are given an asymmetric metric space (V, d), that is, d is not necessarily symmetric, and we would like to construct a shortest tour visiting every vertex exactly once.

The best known efficient algorithm for the ATSP problem approximates the solution within $O(\log n / \log\log n)$, where n is the number of vertices in the metric space (cf. [AGM+10]). On the other hand, the currently best known inapproximability threshold for the ATSP problem is 117/116 (cf. [PV06]).

In this chapter, we prove that that the ATSP problem is **NP**-hard to approximate to within any constant approximation ratio less than 75/74.

TSP Problems with Bounded Metrics

It is conceivable that the special cases of the ATSP problem with bounded metrics are easier to approximate than the cases when the distance between two points grows with the size of the instance. Clearly, the $(1, \mathsf{B})$-ATSP problem, in which the distance function is taking values in the set $\{1, \ldots, \mathsf{B}\}$, can be approximated within B by just picking any tour as the solution.

When we restrict the problem to distances 1 and 2, it can be approximated within 5/4 due to Bläser [B04]. Furthermore, it is **NP**-hard to approximate the problem within any constant approximation ratio less than 321/320 [EK06].

For the case $\mathsf{B} = 8$, Engebretsen and Karpinski [EK06] constructed a reduction yielding the approximation lower bound of 135/134 for the problem.

In this chapter, we prove that it is **NP**-hard to approximate the $(1,2)$-ATSP problem and the $(1,4)$-ATSP problem to within any constant approximation ratio less than 207/206 and 141/140, respectively.

The restricted version of the TSP problem, in which the distance function takes values in $\{1,\ldots,\mathsf{B}\}$, is referred to as the $(1,\mathsf{B})$-TSP problem.

The $(1,2)$-TSP problem can be approximated in polynomial time with an approximation ratio 8/7 due to Berman and Karpinski [BK06]. On the other hand, Engebretsen and Karpinski [EK06] gave an approximation lower bound for the $(1,\mathsf{B})$-TSP problem of 741/740 for $\mathsf{B} = 2$ and 389/388 for $\mathsf{B} = 8$.

In this chapter, we prove that the $(1,2)$-TSP problem and the $(1,4)$-TSP problem are **NP**-hard to approximate to within any constant approximation ratio less than 535/534 and 337/336, respectively.

An instance of the $(1,2)$-TSP problem is called *cubic* and *subcubic* if the graph induced by the edges of weight 1 is cubic (3-regular) and subcubic (maximum degree 3), respectively. As claimed in [CKK02], the inapproximability thresholds for the $(1,2)$-TSP problem restricted to cubic and subcubic instances are 1141/1140 and 673/672 , respectively.

In this chapter, we prove new inapproximability bounds for cubic and subcubic instances of the $(1,2)$-TSP problem of 1141/1140 and 673/672, respectively.

The Graphic TSP Problem on Subcubic and Cubic Graphs

The restricted version of the TSP problem in the shortest path metric completion of graphs is referred to as the GRAPHIC-TSP problem.

The problem is known to be **NP**-hard in exact setting even when we restrict the input graph to be cubic, as the Hamiltonian cycle problem is **NP**-hard for 3-regular graphs (cf. [GJT76]) and can be reduced to the GRAPHIC-TSP problem. The $(1,2)$-TSP problem can be viewed as a special case of the GRAPHIC-TSP problem. To see this, we simply augment the subgraph induced by all weight 1 edges in an instance of the (1, 2)-TSP problem by a new vertex z and add all edges connecting the original vertices with that vertex z. Thus, the explicit approximation lower bound of 535/534 for the (1, 2)-TSP problem is also the inapproximability bound for the GRAPHIC-TSP problem. On the algorithmic front, there was a remarkable progress on the GRAPHIC-TSP problem ([OSS11] ,[MS11], [M12]) leading to the approximation ratio 7/5, cf. Sebö and Vygen [SV12].

The GRAPHIC-TSP problem on cubic graphs as well as subcubic graphs played a crucial role in some recent developments on the GRAPHIC-TSP problem (cf. [GLS05], [BSSS11a], [BSSS11b], [MS11], [M12]). Both problems are of special interest because of its connection to the famous 4/3 conjecture on the integrality gap of the metric TSP problem (cf. [BSSS11a], [BSSS11b]). It is also known that the approximation ratio 3/2 of Christofides' algorithm [C76] for the general metric TSP is tight for the GRAPHIC-TSP problem on cubic graphs. Recently, the first efficient approximation algorithms with approximation ratio 4/3 for the above problem on cubic and subcubic graphs were designed in [BSSS11b] and [MS11]. Correa, Larré and Soto [CLS12] managed to break the bound of 4/3 and gave an efficient algorithm for the problem with approximation ratio $(4/3 - \varepsilon_0)$, where ε_0 is a small constant $(0 < \varepsilon_0 < 10^{-4})$.

In this chapter, we shed some light on the inapproximability status of the GRAPHIC-TSP problem and prove explicit approximation hardness bounds of 1153/1152 for the cubic and 685/684 for the subcubic case.

The MAX-ATSP Problem with Weights 1 and 2

The MAX-(0, 1)-ATSP problem is the restricted version of the MAX-ATSP problem, in which the weight function w takes values in the set $\{0, 1\}$.

Vishwanathan [V92] constructed an approximation preserving reduction proving that an approximation algorithm for the MAX-(0, 1)-ATSP problem with approximation ratio α transforms into an approximation algorithm for the (1, 2)-ATSP problem with approximation ratio $(2 - 1/\alpha)$. Due to this reduction, all negative results concerning the approximation of the $(1,2)$-ATSP problem imply hardness results for the MAX-$(0,1)$-ATSP problem. Since the $(1,2)$-ATSP problem is **APX**-hard [PY93], there is little hope for polynomial time approximation algorithms with arbitrary good precision for the MAX-$(0,1)$-ATSP problem. Due to the explicit approximation lower bound for the $(1,2)$-ATSP problem obtained by Engebretsen and Karpinski [EK06], it is **NP**-hard to approximate the MAX-$(0,1)$-ATSP problem to within any constant approximation ratio less than 320/319. The best up to now known approximation algorithm for the restricted version of this problem is due to Bläser [B04] and achieves an approximation ratio 4/3.

In this chapter, we prove that approximating the MAX-$(0,1)$-ATSP problem to within any constant approximation ratio less than 206/205 is **NP**-hard.

10.2 The Proof Methods and Summary of Results

The results of this chapter depend on several new reductions from a bounded occurrence CSP problem called the MAX-HYBRID-LIN2 problem (cf. Definition 4.9.1). We extend our reduction method from Chapter 9 and define parity gadgets for TSP problems with bounded metrics. The crucial point of the reductions is that we make essential use of the underlying structure of the equations in the MAX-HYBRID-LIN2 problem, which are induced by a 3-regular wheel amplifier graph. This could be a more widely useful method for improving the approximation lower bounds of other problems.

In order to give improved inapproximability results for the ATSP problem and the TSP problem, we introduce a new 3-regular amplifier graph called *bi-wheel* amplifier. This bi-wheel amplifier graph entails an even more advantageous structure of the linear equations in the MAX-HYBRID-LIN2 problem, which enables us to define parity gadgets simulating two variables of the CSP instance simultaneously. Furthermore, we prove that it is only necessary to construct gadgets for roughly one third of the constraints of the CSP instance, while the remaining constraints are simulated without additional cost using the consistency properties of our gadgets. This crucial idea leads to very economical reductions with less vertices in the instances produced by the reduction to the ATSP problem and the TSP problem. We believe that this approach may be useful in improving the efficiency of approximation preserving reductions for other problems.

In Table 10.1, we summarize our approximation lower bounds as compared with previous inapproximability results.

(1, B)-ATSP problem	B = 2	B = 4	B = 8	unbounded
Previously known results	321/320 [EK06]	321/320 [EK06]	135/134 [EK06]	117/116 [PV06]
Our results	207/206	141/140		75/74
(1, B)-TSP problem	B = 2	B = 4	B = 8	unbounded
Previously known results	741/740 [EK06]	741/740 [EK06]	389/388 [EK06]	185/184 [L12]
Our results	535/534	337/336	337/336	123/122
MAX-(0, 1)-ATSP problem				
Previously known results	320/319 [EK06]			
Our results	206/205			

Table 10.1: Comparison of our approximation lower bounds with previously known inapproximability results.

In order to prove the new inapproximability bounds for the (1, 2)-TSP problem restricted to subcubic and cubic instances, we design new cubic gadgets simulating the linear equations with three variables of the MAX-HYBRID-LIN2 problem. This construction will be extended, in a special way, to the cubic and subcubic instances of the GRAPHIC-TSP problem.

In Table 10.2, we summarize our approximation lower bounds as compared with previous inapproximability results.

$(1,2)$-TSP problem	Cubic instances	Subcubic instances
Previously known results	1291/1290 [CKK02]	787/786 [CKK02]
Our results	1141/1140	673/672

Graphic-TSP problem	Cubic instances	Subcubic instances
Previously known results	– –	– –
Our results	1153/1152	685/684

Table 10.2: Known explicit lower bounds and the new results.

10.3 Outline of this Chapter

This chapter is organized as follows. In Section 10.5, we review some selected results related to the topics covered in this chapter. In Section 10.6, we formulate our main results. We give improved inapproximability results for the $(1,2)$-ATSP problem in Section 10.7, the $(1,4)$-ATSP problem in Section 10.8, the $(1,2)$-TSP problem in Section 10.9, the $(1,4)$-TSP problem in Section 10.10, the $(1,2)$-TSP problem restricted to subcubic instances in Section 10.11, the $(1,2)$-TSP problem restricted to cubic instances in Section 10.12 and the GRAPHIC-TSP problem restricted to subcubic and cubic graphs in Section 10.13. In Section 10.14 and 10.15, we study the hardness of approximation of the general metric and asymmetric TSP problem, respectively.

10.4 Preliminaries

In this section, we fix the notation and provide some definitions.

Given an asymmetric metric space (V,d) with $V = \{v_1,\ldots,v_n\}$ and $d : V \times V \to \mathbb{Q}_{\geq 0}$, a tour σ in (V,d) is a tour in the associated complete directed graph $\big(V, \{(v_i,v_j) \in V \times V \mid i \neq j\}\big)$ (cf. Section 9.4). The *length* or *cost* of a tour σ in (V,d) is defined as follows.

$$\text{length of } \sigma \text{ in } (V,d) = \sum_{a\in\sigma} d(a)$$

By means of this notation, we define the ATSP problem as follows.

Definition 10.4.1 (ATSP problem)
Instances: *A asymmetric metric space* (V,d)
Solutions: *A tour* σ *in* (V,d)
Task: *Minimize the length of* σ *in* (V,d)

For every integer $\mathsf{B} \geq 2$, the $(1,\mathsf{B})$-ATSP problem is the ATSP problem restricted to instances (V,d) with $d : \{(v_i,v_j) \in V \times V \mid i \neq j\} \to [\mathsf{B}]$.

Given a complete graph $\mathcal{G}$, a tour σ in $\mathcal{G}$ is a subset of $E(\mathcal{G})$, which induces a cycle in $\mathcal{G}$ visiting every vertex of $\mathcal{G}$ only once. A *Hamiltonian path* in $\mathcal{G}$ is a simple path containing all vertices of $\mathcal{G}$. For a given metric space (V,d) with $V = \{v_1,\ldots,v_n\}$ and $d : \binom{V}{2} \to \mathbb{Q}_{>0}$, a tour σ in (V,d) is a tour in the associated complete graph $\big(V, \binom{V}{2}\big)$. The length or cost of a tour σ in (V,d) is defined as follows.

$$\text{length of } \sigma \text{ in } (V,d) = \sum_{e\in\sigma} d(e)$$

Analogously, the TSP problem can be formulated as follows.

Definition 10.4.2 (TSP problem)
Instances: *A metric space* (V,d)
Solutions: *A tour* σ *in* (V,d)
Task: *Minimize the length of* σ *in* (V,d)

For every integer $\mathsf{B} \geq 2$, the $(1,\mathsf{B})$-TSP problem is the TSP problem restricted to instances (V,d) with $d : \binom{V}{2} \to [\mathsf{B}]$.

In order to specify, an instance (V,d) of the $(1,2)$-ATSP problem, it suffices to identify the arcs $a \in V \times V$ with $d(a) = 1$. Accordingly, we decode an instance $(\{v_1,\ldots,v_n\},d)$ by a directed graph $\mathcal{G}_d$, where for every

$a \in \{(v_i, v_j) \in V \times V \mid i \neq j\}$, we have

$$d(a) = \begin{cases} 1 & \text{if } a \in A(\mathcal{G}_d) \\ 2 & \text{else.} \end{cases}$$

Analogously, in the symmetric case, an instance of the $(1,2)$-TSP problem is completely specified by a graph.

In the following, for every $c \in \{1,2\}$, we refer to an arc a and an edge e with $d(a) = d(e) = c$ as a *c-arc* and a *c-edge*, respectively.
We call an instance of the $(1,2)$-TSP problem cubic and subcubic if the graph induced by the all weight 1 edges is cubic and subcubic, respectively.

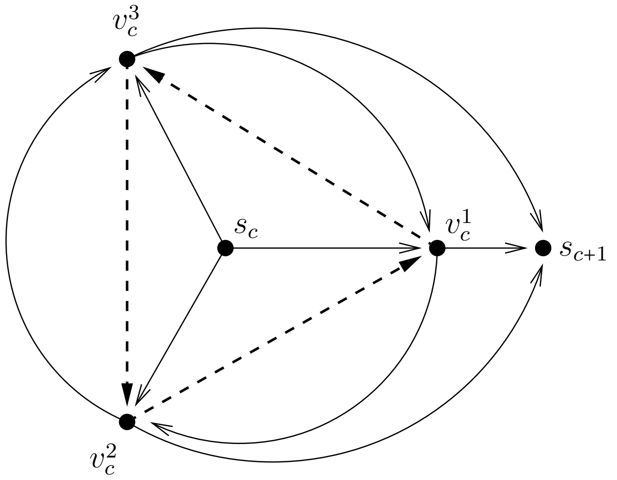


Figure 10.1: Graph $\mathcal{G}^{3A}$ simulating equations of the form $x \oplus y \oplus z = 0$.

10.5 Related Work

Engebretsen and Karpinski [EK06] constructed approximation preserving reductions from the MAX-HYBRID-LIN2 problem to the $(1,2)$-ATSP problem and to the $(1,2)$-TSP problem implying explicit approximation lower bounds for both problems. They introduced graphs (gadgets), which simulate variables, equations with two variables and equations with three variables. For equations of the form $x \oplus y \oplus z = 0$, they used a gadget that was constructed in [PV06] displayed in Figure 10.1. We rely on this gadget and use it in our reduction. In particular, we will exploit the following statement that was proved by Papadimitriou an Vempala [PV06].

Lemma 10.5.1 ([PV06])
There is a simple path from s_c to s_{c+1} in $\mathcal{G}^{3A}$ (Figure 10.1) containing all vertices of $\mathcal{G}^{3A}$ if and only if an even number of dashed arcs is traversed.

In [EK06], a similar gadget was constructed to prove explicit approximation lower bounds for the $(1,2)$-TSP problem. The graph corresponding to an equation of the from $x \oplus y \oplus z = 0$ is displayed in Figure 10.2.

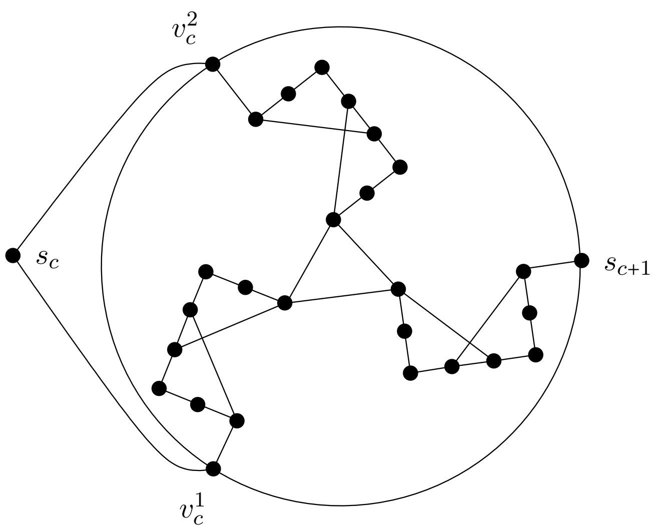


Figure 10.2: Gadgets used in [EK06] to prove approximation hardness of the $(1,2)$–TSP problem.

In the reduction of the $(1,2)$-TSP problem, the following statement was proved by Engebretsen and Karpinski [EK06] concerning the graph corresponding to equations of the form $x \oplus y \oplus z = 0$.

Lemma 10.5.2 ([EK06])
There is a simple path from s_c to s_{c+1} in $\mathcal{G}^{3S}$ (Figure 10.2) containing the vertices $v \in \{v_c^1, v_c^2\}$ if and only if an even number of parity graphs (Figure

10.50) is traversed.

By using the hardness results due to Berman and Karpinski [BK99] for the MAX-HYBRID-LIN2 problem, the reductions due to Engebretsen and Karpinski [EK06] yield the following inapproximability results.

Theorem 10.5.1 ([EK06])
The $(1,2)$-ATSP *problem and the* $(1,2)$-TSP *problem are* **NP**-*hard to approximate to within any constant approximation ratio less than* $321/320$ *and* $741/740$*, respectively.*

Inapproximability Results for the MAX-ATSP problem

By replacing all arcs with weight 2 of an instance of the $(1,2)$-ATSP problem by arcs of weight 0, we obtain an instance of the MAX-$(0,1)$-ATSP problem. Vishwanathan [V92] proved that this reduction relates the $(1,2)$-ATSP problem to the MAX-ATSP problem in the following sense.

Lemma 10.5.3 ([V92])
A polynomial time approximation algorithm with approximation ratio α *for the* MAX-$(0,1)$-ATSP *problem implies a polynomial time approximation algorithm for the* $(1,2)$-ATSP *problem with approximation ratio* $(2-1/\alpha)$*.*

This reduction transforms every hardness result addressing the $(1,2)$-ATSP problem into a hardness result for the MAX-$(0,1)$-ATSP problem. In particular, Theorem 10.5.1 implies the best known explicit approximation lower bound for the MAX-$(0,1)$-ATSP problem stated below.

Corollary 10.5.1
It is **NP**-*hard to approximate the* MAX-$(0,1)$-ATSP *problem to within any constant approximation ratio less than* $320/319$*.*

10.6 Our Contributions

We now formulate our main results.

Theorem 10.6.1

Suppose we are given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 *problem with n wheels, m_2 equations with two variables and m_3 equations with exactly three variables with the properties described in Theorem 4.9.1.*

(*i*) *Then, it is possible to construct in polynomial time an instance $\mathcal{G}_{\mathscr{L}}$ of the* (1, 2)-ATSP *problem with the following properties:*

- *If there exists an assignment ϕ to the variables of $\mathscr{L}$ which leaves at most τ equations unsatisfied, then, there exist a tour in $\mathcal{G}_{\mathscr{L}}$ with length at most $3m_2 + 13m_3 + 3(n+1) - 1 + \tau$.*
- *From every tour σ in $\mathcal{G}_{\mathscr{L}}$ with length $3m_2+13m_3+3(n+1)-1+\tau$, it is possible to construct in polynomial time an assignment ψ_σ to the variables of $\mathscr{L}$ that leaves at most τ equations in $\mathscr{L}$ unsatisfied.*

(*ii*) *Furthermore, it is possible to construct in polynomial time an instance $(V_{\mathscr{L}}, d_{\mathscr{L}})$ of the* (1, 4)-ATSP *problem with the following properties:*

- *If there exists an assignment ϕ to the variables of $\mathscr{L}$ which leaves at most τ equations unsatisfied, then, there exist a tour in $(V_{\mathscr{L}}, d_{\mathscr{L}})$ with length at most $4m_2 + 20m_3 + 6(n+1) + 2\tau - 2$.*
- *From every tour σ in $(V_{\mathscr{L}}, d_{\mathscr{L}})$ with length $4m_2+20m_3+6(n+1)+2\tau-2$, it is possible to construct in polynomial time an assignment ψ_σ to the variables of $\mathscr{L}$ that leaves at most τ equations in $\mathscr{L}$ unsatisfied.*

The former theorem can be used to derive an explicit approximation lower bound for the (1, 2)-ATSP problem by reducing instances of the MAX-HYBRID-LIN2 problem of the form described in Theorem 4.9.1 to the (1, 2)-ATSP problem.

Corollary 10.6.1

It is **NP***-hard to approximate the* (1, 2)-ATSP *problem to within any constant approximation ratio less than* 207/206.

Proof of Corollary 10.6.1.

Let $\mathscr{L}^3$ be an instance of the MAX-E3LIN2 problem. For a fixed $\varepsilon > 0$, we choose constants $\delta \in (0, 1/2)$ and $k \in \mathbb{N}$ such that

$$\frac{207 - \delta}{206 + \delta + \dfrac{7}{k}} \geq \frac{207}{206} - \varepsilon$$

holds. Then, we generate k copies of $\mathscr{L}^3$ and the corresponding instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem. Given $\mathscr{L}$, we construct the corresponding instance $\mathcal{G}_{\mathscr{L}}$ of the (1,2)-ATSP problem with properties described in Theorem 10.6.1. We conclude according to Theorem 4.9.1 that there exist a tour in $\mathcal{G}_{\mathscr{L}}$ with length at most

$$\begin{aligned} 3 \cdot 60\nu \cdot k + 13 \cdot 2\nu \cdot k + \delta\nu \cdot k + n + 1 &\leq \left(206 + \delta + \frac{n+1}{\nu \cdot k}\right)\nu \cdot k \\ &\leq \left(206 + \delta + \frac{6+1}{k}\right)\nu \cdot k \end{aligned}$$

or the length of a tour in $\mathcal{G}_{\mathscr{L}}$ is bounded from below by

$$\begin{aligned} 3 \cdot 60\nu \cdot k + 13 \cdot 2\nu \cdot k + (1 - \delta)\nu \cdot k + n + 1 &\geq (206 + (1 - \delta))\nu \cdot k \\ &\geq (207 - \delta)\nu \cdot k. \end{aligned}$$

From Theorem 4.9.1, we know that the two cases above are **NP**-hard to distinguish. Hence, for every $\epsilon > 0$, it is **NP**-hard to find a solution to the (1,2)-ATSP problem with approximation ratio better than

$$\frac{207 - \delta}{206 + \delta + \dfrac{7}{k}} \geq \frac{207}{206} - \varepsilon$$

and the proof follows. ■

Analogously, we derive the following statement by combining Theorem 4.9.1 and Theorem 10.6.1.

Corollary 10.6.2
It is **NP**-*hard to approximate the* (1,4)-ATSP *problem to within any constant approximation ratio less than* 141/140.

For the symmetric version of the problems, we construct reductions from the MAX-HYBRID-LIN2 problem with similar properties.

Theorem 10.6.2
Suppose we are given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 *problem with n wheels, m_2 equations with two variables and m_3 equations with exactly three variables with the properties described in Theorem 4.9.1.*

(*i*) *Then, it is possible to construct in polynomial time an instance $\mathcal{G}_{\mathscr{L}}$ of the* $(1,2)$-TSP *problem with the following properties:*

- *If there exists an assignment ϕ to the variables of $\mathscr{L}$ which leaves at most τ equations unsatisfied, then, there exist a tour in $\mathcal{G}_{\mathscr{L}}$ with length at most $8m_2 + 27m_3 + 3n + 2 + \tau$.*
- *From every tour σ in $\mathcal{G}_{\mathcal{H}}$ with length $8m_2 + 27m_3 + 3n + 2 + \tau$, it is possible to construct in polynomial time an assignment ψ_σ to the variables of $\mathscr{L}$ that leaves at most τ equations in $\mathscr{L}$ unsatisfied.*

(*ii*) *Furthermore, it is possible to construct in polynomial time an instance $(V_{\mathscr{L}}, d_{\mathscr{L}})$ of the* $(1,4)$-TSP *problem with the following properties:*

- *If there exists an assignment ϕ to the variables of $\mathscr{L}$ which leaves at most τ equations unsatisfied, then, there exist a tour in $(V_{\mathcal{H}}, d_{\mathcal{H}})$ with length at most $10m_2 + 36m_3 + 6(n + 1) + 2 + 2\tau$.*
- *From every tour σ in $(V_{\mathscr{L}}, d_{\mathscr{L}})$ with length $10m_2+36m_3+6(n+1)+2+2\tau$, it is possible to construct in polynomial time an assignment ψ_σ to the variables of $\mathscr{L}$ that leaves at most τ equations in $\mathscr{L}$ unsatisfied.*

Analogously, we combine the former theorem with the explicit approximation lower bound for the MAX-HYBRID-LIN2 problem described in Theorem 4.9.1 yielding the following approximation hardness result.

Corollary 10.6.3
It is **NP**-*hard to approximate the* $(1,2)$-TSP *problem and the* $(1,4)$-TSP *problem to within any factor better than* $535/534$ *and* $337/336$, *respectively.*

By Lemma 10.5.3 and Corollary 10.6.1, we obtain the following explicit approximation lower bound for the MAX-$(0,1)$-ATSP problem.

Corollary 10.6.4
It is **NP**-*hard to approximate the* MAX-$(0,1)$-ATSP *problem to within any constant factor less than* 206/205.

For the $(1,2)$-ATSP problem restricted to subcubic instances, we prove the following explicit inapproximability result.

Theorem 10.6.3
The $(1,2)$-TSP *problem restricted to subcubic instances is* **NP**-*hard to approximate to within any factor less than* 673/672.

For cubic instances of the $(1,2)$-ATSP problem, we obtain the following inapproximability threshold.

Theorem 10.6.4
The $(1,2)$-TSP *problem restricted to subcubic instances is* **NP**-*hard to approximate to within any factor less than* 1141/1140.

For subcubic and cubic instances of the Graphic TSP, we prove the following.

Theorem 10.6.5
The GRAPHIC-TSP *problem restricted to subcubic and cubic graphs is* **NP**-*hard to approximate to within any factor less than* 685/684 *and* 1153/1152, *respectively.*

For the general version of the metric TSP problem, we give the following inapproximability result.

Theorem 10.6.6
It is **NP**-*hard to approximate the* TSP *problem to within any constant approximation ratio less than* 123/122.

For the general version of the ATSP problem, we obtain the following explicit approximation lower bound.

Theorem 10.6.7
It is **NP***-hard to approximate the* ATSP *problem to within any constant approximation ratio less than* $75/74$.

10.7 The $(1,2)$-ATSP Problem

Before we proceed to give the proof of Theorem 10.6.1 (i), we describe the reduction from a high-level view and try to build some intuition.

10.7.1 High-Level View of the Reduction

As mentioned above, we prove our hardness results by a reduction from the MAX-HYBRID-LIN2 problem (cf. Definition 4.9.1). Suppose we are given an instance $\mathscr{L}^3$ of the MAX-E3LIN2 problem and $\mathscr{L}$ the corresponding instance of the MAX-HYBRID-LIN2 problem. Recall that every variable x^l in the original instance $\mathscr{L}^3$ introduces an associated wheel $\mathcal{W}_l$ in the instance $\mathscr{L}$ as illustrated in Figure 4.1.

The idea of our reduction is to make use of the special structure of the wheels in $\mathscr{L}$. Every wheel $\mathcal{W}_l$ in $\mathscr{L}$ corresponds to a graph $\mathcal{G}_l$ in the instance $\mathcal{G}_{\mathscr{L}}$ of the $(1,2)$-ATSP problem. Moreover, $\mathcal{G}_l$ is a subgraph of $\mathcal{G}_{\mathscr{L}}$, which forms almost a cycle. An assignment to the variable x^l will have a natural interpretation in this reduction. The parity of x^l, that is, the value $b \in \{0,1\}$ of the variable x^l, corresponds to the direction of movement in $\mathcal{G}_l$ of the underlying tour. The wheel graphs of $\mathcal{G}_{\mathscr{L}}$ are connected and build together the *inner loop* of $\mathcal{G}_{\mathscr{L}}$. Every variable x_i^l in a wheel $\mathcal{W}_l$ possesses an associated parity graph $\mathcal{P}_i^l$ (Figure 10.4) in $\mathcal{G}_l$ as a subgraph. The two natural ways to traverse a parity graph will be called 0/1-traversals and correspond to the parity of the variable x_i^l. Some of the parity graphs in $\mathcal{G}_l$ are also contained in graphs $\mathcal{G}_c^{3A}$ (Figure 10.1 and Figure 10.6 for a more detailed view) corresponding to equations with three variables of the form $\ell_c^3 \equiv x \oplus y \oplus z = 0$. (By negating a variable in an equation $x \oplus y \oplus z = 1$, we obtain the equation $\bar{x} \oplus y \oplus z = 0$, which is satisfied by an assignment to x, y and z if and only if the former equation is satisfied. Accordingly, we only need to consider equations of the from $\bar{x} \oplus y \oplus z = 0$ or $x \oplus y \oplus z = 0$)

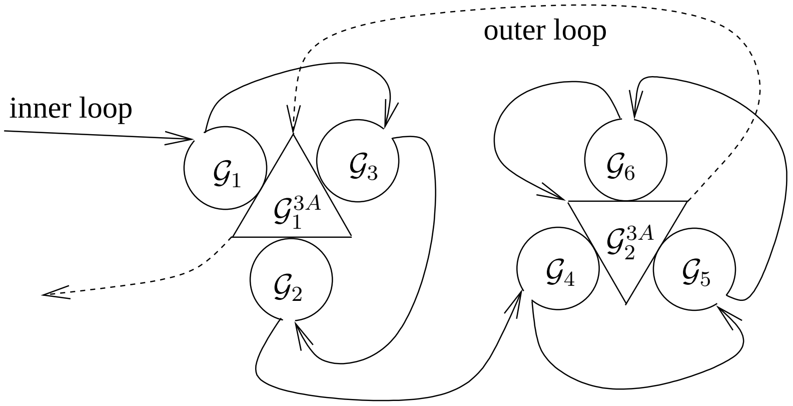


Figure 10.3: An illustration of the instance $\mathcal{G}_{\mathscr{L}}$ and a tour in $\mathcal{G}_{\mathscr{L}}$.

The graphs corresponding to equations with three variables are connected and build the *outer loop* of $\mathcal{G}_{\mathscr{L}}$. The whole construction is illustrated in Figure 10.3. The outer loop of the tour checks whether the 0/1-traversals of the parity graphs correspond to a satisfying assignment of the equations with three variables. If an underlying equation is not satisfied by the assignment defined via a 0/1-traversal of the associated parity graph, it will be punished by using a costly 2-arc.

10.7.2 Constructing $\mathcal{G}_{\mathscr{L}}$ from $\mathscr{L}$

Given a instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem, we are going to construct the corresponding instance $\mathcal{G}_{\mathscr{L}}$ of the (1,2)-ATSP problem.

For every type of equation in $\mathscr{L}$, we will introduce a specific graph or a specific way to connect the so far constructed subgraphs. In particular, we will distinguish between graphs corresponding to cycle equations, matching equations, wheel border equations and equations with three variables. First of all, we introduce graphs corresponding to the variables in $\mathscr{L}$.

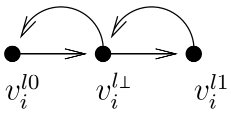


Figure 10.4: Parity graph $\mathcal{P}_i^l$ corresponding to the variable x_i^l in $\mathcal{W}_l$.

Variable Graphs

Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 problem and $\mathcal{W}_l$ a wheel in $\mathscr{L}$. For every variable x_i^l in the wheel $\mathcal{W}_l$, we introduce the parity graph $\mathcal{P}_i^l$ consisting of the vertices v_i^{l1}, $v_i^{l\perp}$ and v_i^{l0} displayed in Figure 10.4.

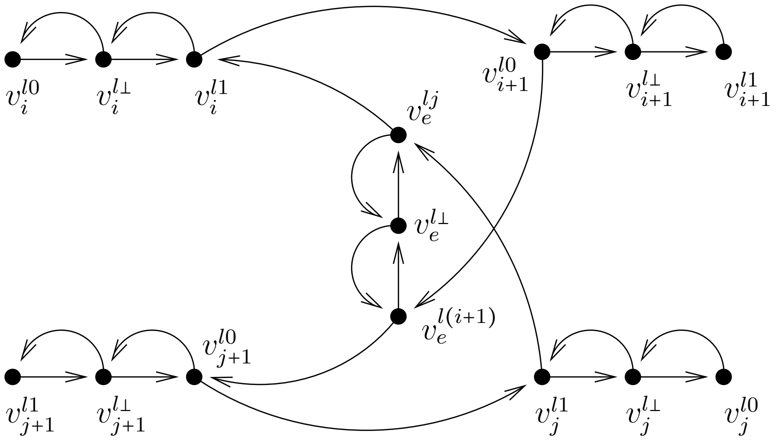


Figure 10.5: Connecting the parity graph $\mathcal{P}_e^l$.

Matching and Circle Equations

Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 problem, $\mathcal{W}_l$ a wheel in $\mathscr{L}$ and $M(\mathcal{W}_l)$ the associated perfect matching. Furthermore, let $x_i^l \oplus x_j^l = 0$ with $e = \{i,j\} \in M(\mathcal{W}_l)$ and $i < j$ be a matching equation. Due to the construction of $\mathscr{L}$, the cycle equations $x_i^l \oplus x_{i+1}^l = 0$ and $x_j^l \oplus x_{j+1}^l = 0$ are both contained in $\mathcal{W}_l$. Then, we introduce the associated parity graph $\mathcal{P}_e^l$ consisting of the vertices v_e^{lj}, $v_e^{l\perp}$ and $v_e^{l(i+1)}$. In addition to it, we connect the parity graphs $\mathcal{P}_i^l$, $\mathcal{P}_{i+1}^l$, $\mathcal{P}_j^l$, $\mathcal{P}_{j+1}^l$ and $\mathcal{P}_e^l$ as displayed in Figure 10.5.

Graphs Corresponding to Equations with Three Variables

Let $\ell_c^3 \equiv x_i^l \oplus x_j^s \oplus x_t^k = 0$ be an equation with three variables in $\mathscr{L}$. Then, we introduce the graph $\mathcal{G}_c^{3A}$ (Figure 10.1) corresponding to the equation ℓ_c^3. The graph $\mathcal{G}_c^{3A}$ includes the vertices s_c, v_c^1, v_c^2, v_c^3 and s_{c+1}.

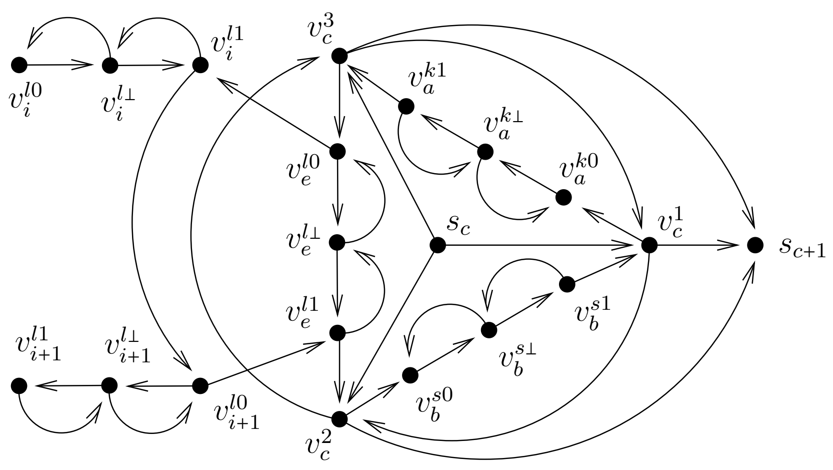


Figure 10.6: The graph $\mathcal{G}_c^{3A}$ corresponding to $\ell_c^3 \equiv x_i^l \oplus x_j^s \oplus x_k^u = 0$, which is connected to the graph corresponding to $x_i^l \oplus x_{i+1}^l = 0$.

Furthermore, it contains the parity graphs $\mathcal{P}_e^l$, $\mathcal{P}_b^s$ and $\mathcal{P}_a^k$ as subgraphs, where $e = \{i, i+1\}$, $b = \{j, j+1\}$ and $a = \{t, t+1\}$. Exemplary, we display $\mathcal{G}_c^{3A}$ with its connections to the graph corresponding to the cycle equation $x_i^l \oplus x_{i+1}^l = 0$ in Figure 10.6.

In case of $\ell_c^3 \equiv \bar{x}_i^l \oplus x_j^s \oplus x_k^u = 0$, we connect the parity graphs with arcs (v_i^{l1}, v_e^{l1}), (v_{i+1}^{l0}, v_i^{l1}) and (v_e^{l0}, v_{i+1}^{l0}).

Graphs Corresponding to Wheel Border Equations

Let $\mathcal{W}_l$ and $\mathcal{W}_{l+1}$ be wheels in $\mathscr{L}$. In addition, let $x_1^l \oplus x_n^l = 0$ be the wheel border equation of $\mathcal{W}_l$. Recall that x_n^l also occurs in an equation ℓ_c^3 with three variables. Let us assume that ℓ_c^3 is of the form $\bar{x}_n^l \oplus y \oplus z = 0$. Then, we introduce the vertex b_l and connect it to v_1^{l0} and v_n^{l1}. Let b_{l+1} be the vertex corresponding to the wheel $\mathcal{W}_{l+1}$. We draw an arc from v_1^{l0} to b_{l+1}.

Finally, we connect the vertex $v^{l0}_{\{n,1\}}$ to b_{l+1}. This construction is illustrated in Figure 10.7, where we displayed only a part of the corresponding graph $\mathcal{G}^{3A}_c$.

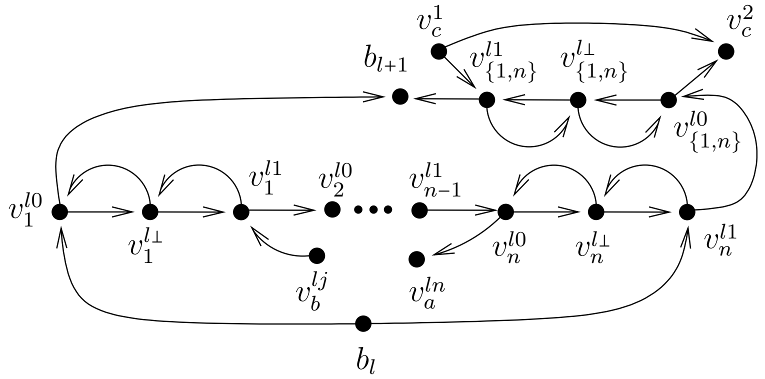


Figure 10.7: The graph corresponding to $x^l_1 \oplus x^l_n = 0$ in the case $\bar{x}^l_n \oplus y \oplus z = 0$

For equations ℓ^3_c of the form $x^l_n \oplus y \oplus z = 0$, we use a similar construction, in which the parity graph $\mathcal{P}^l_{\{1,n\}}$ is connected to b_l and v^{l1}_n by means of $\left(b_l, v^{l0}_{\{1,n\}}\right)$ and $\left(v^{l1}_{\{1,n\}}, v^{l1}_n\right)$. Furthermore, we connect v^{l1}_n with b_{l+1}. Let $\mathcal{W}_n$ be the last wheel in $\mathscr{L}$. Then, we set $b_{n+1} = s_1$ as s_1 is the starting vertex of the graph $\mathcal{G}^{3A}_1$ corresponding to the equation ℓ^3_1. This is the whole description of the graph $\mathcal{G}_{\mathscr{L}}$.

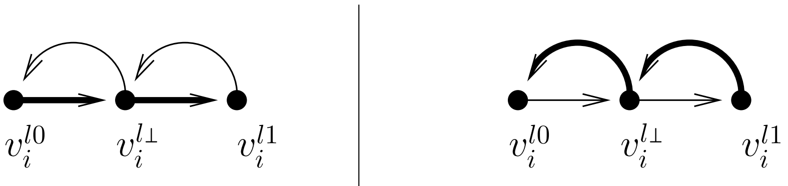
10.7.3 Constructing a Tour from an Assignment

Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 problem and $\mathcal{G}_{\mathscr{L}}$ the corresponding instance of the $(1,2)$-ATSP problem as defined in Section 10.7.2. Given an assignment ϕ to the variables of $\mathscr{L}$, we are going to construct a tour in $\mathcal{G}_{\mathscr{L}}$ in dependence of ϕ. In addition to it, we analyze the relation between the length of the tour and the number of satisfied equations by ϕ.

Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 problem consisting of the wheels $\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_n$ and equations with three variables ℓ^3_c, where $c \in [m_3]$. The tour in $\mathcal{G}_{\mathscr{L}}$ starts at the vertex b_1. From a high-level view, it traverses all graphs corresponding to equations associated with the wheel $\mathcal{W}_1$ ending with the vertex b_2. Successively, it passes all graphs for each wheel in $\mathscr{L}$ until it reaches the vertex $b_{n+1} = s_1$ as s_1 is the starting vertex of the graph $\mathcal{G}_1^{3A}$.

At this point, the tour begins to traverse the remaining graphs $\mathcal{G}_c^{3A}$ which are simulating equations with three variables in $\mathscr{L}$. By now, some of the parity graphs appearing in graphs $\mathcal{G}_c^{3A}$ already have been traversed in the *inner loop* of the tour. The *outer loop* checks whether for each graph $\mathcal{G}_c^{3A}$, an even number of parity graphs has been traversed in the inner loop. In every situation, in which ϕ leaves the underlying equation unsatisfied, the tour needs to use a 2-arc. For each wheel $\mathcal{W}_l$, we have to use 2-arcs in order to obtain a Hamiltonian path from b_l to b_{l+1} traversing all graphs associated with $\mathcal{W}_l$ in some order except in the case when all variables in the wheel have the same parity.

In order to analyze the length of the tour in relation to the number of satisfied equation, we are going to examine the part passing the graphs corresponding to the underlying equation and account the local length to those parts of the tour. Let us begin to describe the tour passing through parity graphs associated to variables in $\mathscr{L}$.



1-traversal of $\mathcal{P}_i^l$ given $\phi(x_i^l) = 1$.

0-traversal of $\mathcal{P}_i^l$ given $\phi(x_i^l) = 0$.

Figure 10.8: Traversal of the graph $\mathcal{P}_i^l$ given the assignment ϕ. The traversed arcs are represented by thick arrows.

Traversing Variable Graphs

Let x_i^l be a variable in $\mathscr{L}$. Then, the tour traverses the parity graph $\mathcal{P}_i^l$ by using the path $v_i^{l[1-\phi(x_i^l)]} \to v_i^{l\perp} \to v_i^{l\phi(x_i^l)}$. In the following, we call this part of

the tour a 0/1-*traversal* of the parity graph. In Figure 10.8, we displayed the corresponding traversals of the graph $\mathcal{P}_i^l$ given the assignment $\phi(x_i^l)$, where the traversed arcs are illustrated by thick arrows.
In both cases, we associate the local length 2 with this part of the tour.

Traversing Graphs Corresponding to Matching Equations

Let $\mathcal{W}_l$ be a wheel in $\mathscr{L}$ and $x_i^l \oplus x_j^l = 0$ with $e = \{i, j\} \in M(\mathcal{W}_l)$ a matching equation. We assume that $i < j$ holds. Given $x_i^l \oplus x_{i+1}^l = 0$, $x_i^l \oplus x_j^l = 0$, $x_j^l \oplus x_{j+1}^l = 0$ and the assignment ϕ, we are going to construct a tour through the corresponding parity graphs in dependence of ϕ. We begin with the case $\phi(x_i^l) \oplus \phi(x_{i+1}^l) = 0$, $\phi(x_i^l) \oplus \phi(x_j^l) = 0$ and $\phi(x_j^l) \oplus \phi(x_{j+1}^l) = 0$.

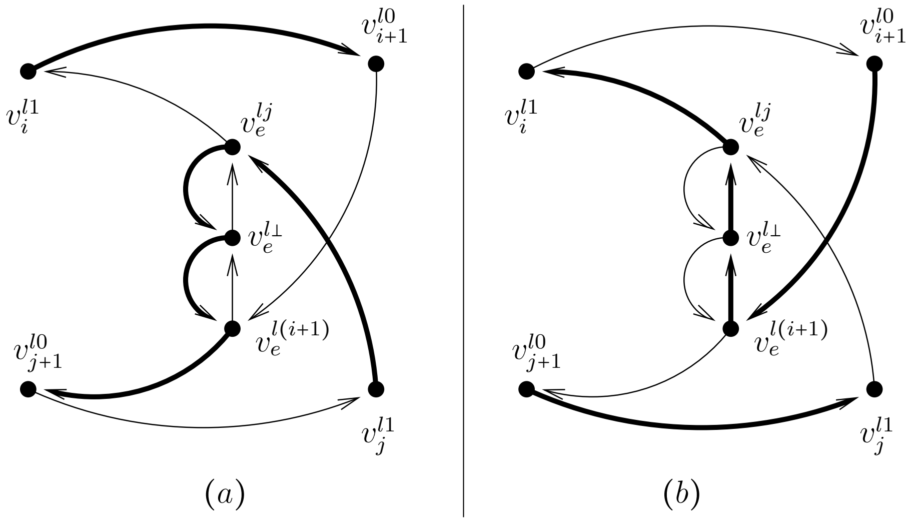


Figure 10.9: The scenario in the 1. Case.

1. Case ($\phi(x_i^l) \oplus \phi(x_{i+1}^l) = 0$, $\phi(x_i^l) \oplus \phi(x_j^l) = 0$ and $\phi(x_j^l) \oplus \phi(x_{j+1}^l) = 0$): In this case, we traverse the corresponding parity graphs as displayed in Figure 10.9. In Figure 10.9 (a), we have $\phi(x_i^l) = \phi(x_{i+1}^l) = \phi(x_j^l) = \phi(x_{j+1}^l) = 1$, whereas in Figure 10.9 (b), we have $\phi(x_i^l) = \phi(x_{i+1}^l) = \phi(x_j^l) = \phi(x_{j+1}^l) = 0$. In both cases, this part of the tour has local length 5.

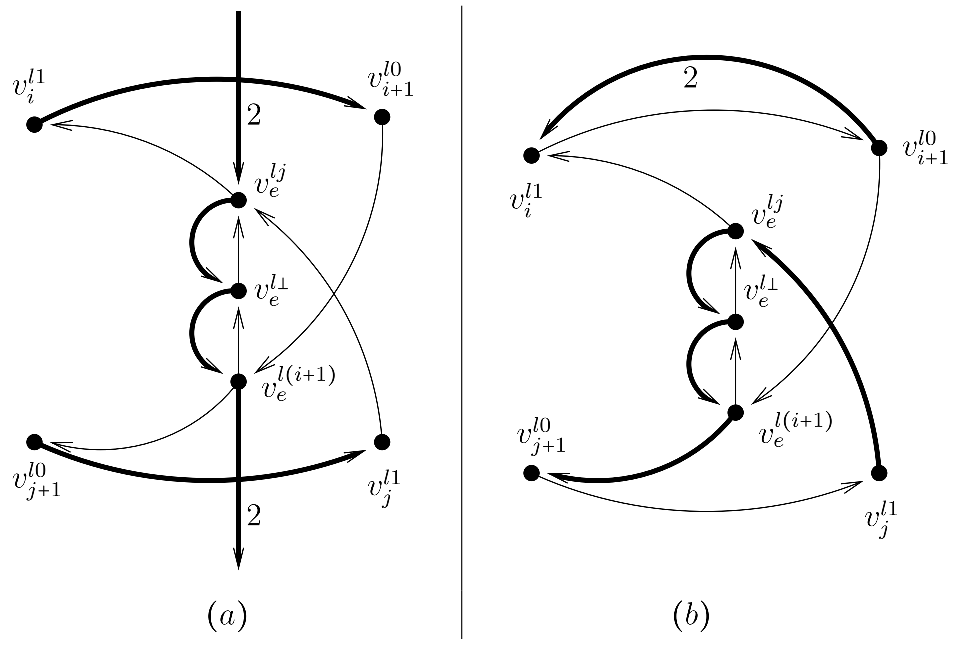


Figure 10.10: The scenario in the 2. Case.

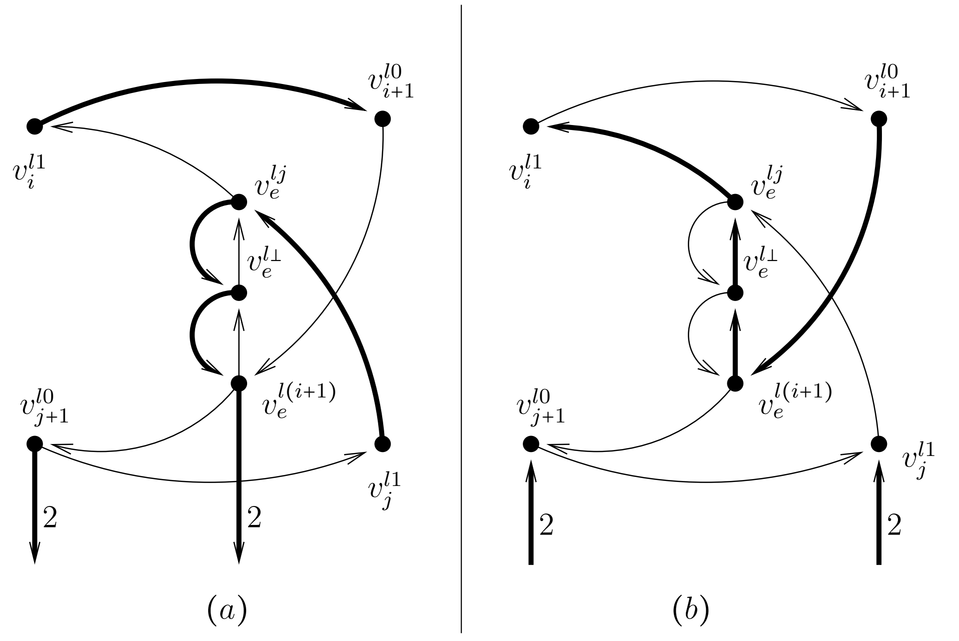


Figure 10.11: The scenario in the 3. Case.

2. Case ($\phi(x_i^l) \oplus \phi(x_{i+1}^l) = 0$, $\phi(x_i^l) \oplus \phi(x_j^l) = 1$ and $\phi(x_j^l) \oplus \phi(x_{j+1}^l) = 0$): The tour is displayed in Figure 10.10 (a) and (b).

In the case $\phi(x_i^l) = \phi(x_{i+1}^l) = 1$ and $\phi(x_j^l) = \phi(x_{j+1}^l) = 0$ displayed in Figure 10.10 (a), we are forced to enter and leave the parity graph $\mathcal{P}_e^l$ via 2-arcs. So far, we associate the local length 6 with this part of the tour.

In Figure 10.10 (b), we have $\phi(x_i^l) = \phi(x_{i+1}^l) = 0$ and $\phi(x_j^l) = \phi(x_{j+1}^l) = 1$. This part of the tour contains one 2-arc yielding the local length 6.

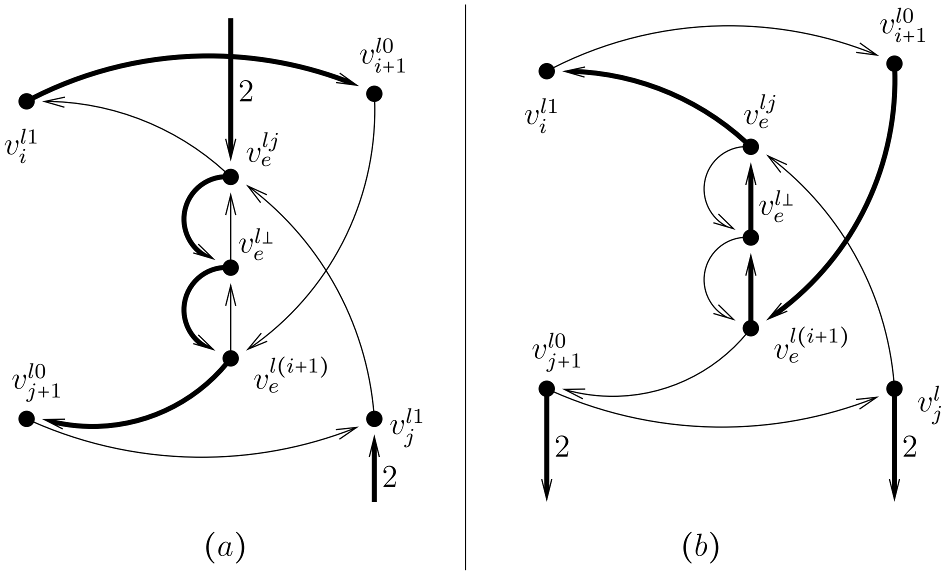


Figure 10.12: The scenario in the 4. Case.

3. Case ($\phi(x_i^l) \oplus \phi(x_{i+1}^l) = 0$, $\phi(x_i^l) \oplus \phi(x_j^l) = 0$ and $\phi(x_j^l) \oplus \phi(x_{j+1}^l) = 1$): In dependence of ϕ, we traverse the parity graphs as displayed in Figure 10.11. The situation, in which $\phi(x_i^l) = \phi(x_{i+1}^l) = 1$ and $\phi(x_j^l) = 1 \neq \phi(x_{j+1}^l)$ holds, is depicted in Figure 10.11 (a). On the other hand, if we have $\phi(x_i^l) = \phi(x_{i+1}^l) = 0$ and $\phi(x_j^l) = 0 \neq \phi(x_{j+1}^l)$, the tour is pictured in Figure 10.11 (b). In both cases, we associate the local length 6.

4. Case ($\phi(x_i) \oplus \phi(x_{i+1}) = 0$, $\phi(x_i) \oplus \phi(x_j) = 1$ and $\phi(x_j) \oplus \phi(x_{j+1}) = 1$): The tour is displayed in Figure 10.12. In Figure 10.12 (a), we are given $\phi(x_i^l) = \phi(x_{i+1}^l) = 1$ and $\phi(x_j^l) \neq \phi(x_{j+1}^l) = 1$, whereas in Figure 10.12 (b), we have $\phi(x_i^l) = \phi(x_{i+1}^l) = 0$ and $\phi(x_j^l) \neq \phi(x_{j+1}^l) = 0$. In both cases, we associate the local length 6 with this part of the tour.

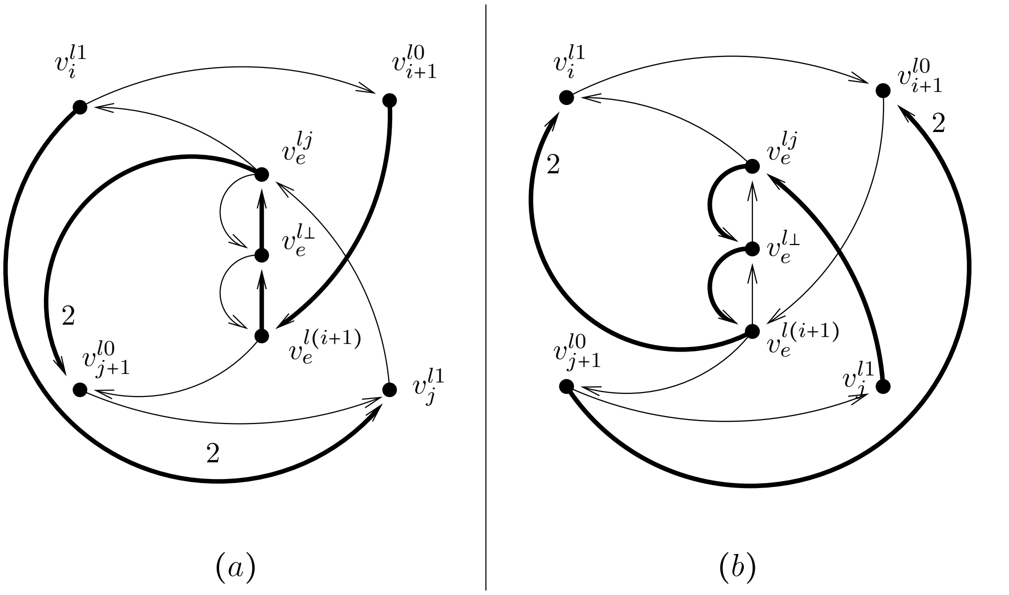


Figure 10.13: The scenario in the 5.Case.

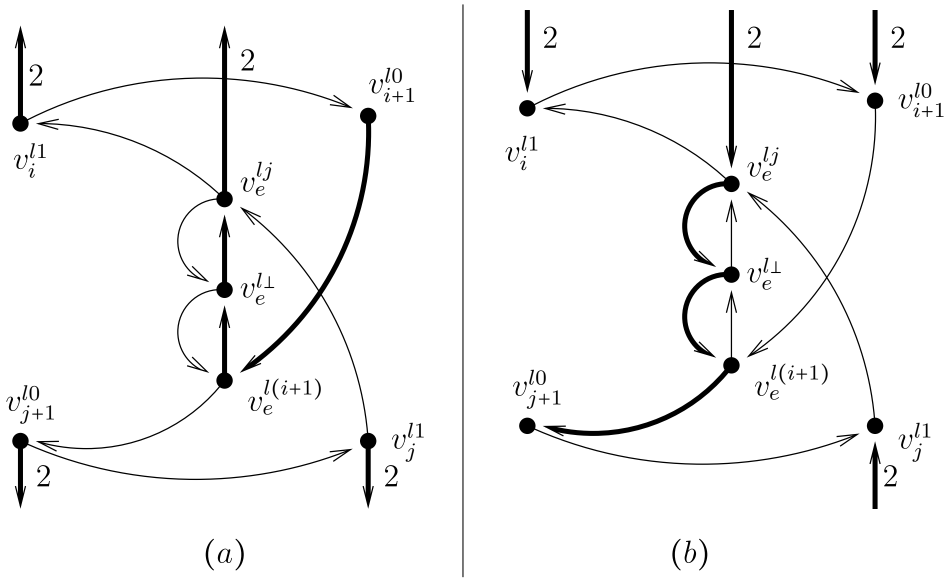


Figure 10.14: The scenario in the 6. Case.

5. Case ($\phi(x_i^l) \oplus \phi(x_{i+1}^l) = 1$, $\phi(x_i^l) \oplus \phi(x_j^l) = 1$ and $\phi(x_j^l) \oplus \phi(x_{j+1}^l) = 1$): In this case, we traverse the corresponding parity graphs as depicted in Figure 10.13.

In Figure 10.13 (a), we notice that $\phi(x_i^l) \neq \phi(x_{i+1}^l) = 0$ and $\phi(x_j^l) \neq \phi(x_{j+1}^l) = 1$, whereas in (b), we have $\phi(x_i^l) \neq \phi(x_{i+1}^l) = 1$ and $\phi(x_j^l) \neq \phi(x_{j+1}^l) = 0$. This part of the tour has local length 7.

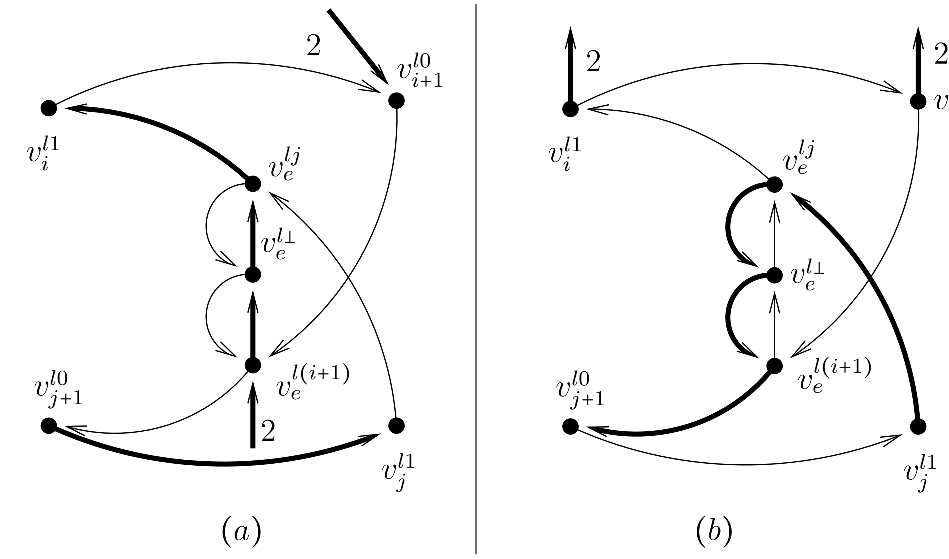


Figure 10.15: The scenario in the 7. Case.

6. Case ($\phi(x_i^l) \oplus \phi(x_{i+1}^l) = 1$, $\phi(x_i^l) \oplus \phi(x_j^l) = 0$ and $\phi(x_j^l) \oplus \phi(x_{j+1}^l) = 1$): In this case, we traverse the corresponding parity graphs as depicted in Figure 10.14. In Figure 10.14 (a), we have $\phi(x_i^l) \neq \phi(x_{i+1}^l) = 0$ and $\phi(x_j^l) \neq \phi(x_{j+1}^l) = 0$, whereas in (b), we have $\phi(x_i^l) \neq \phi(x_{i+1}^l) = 1$ and $\phi(x_j^l) \neq \phi(x_{j+1}^l) = 1$. This part of the tour has local length 7.

7. Case ($\phi(x_i) \oplus \phi(x_{i+1}) = 1$, $\phi(x_i) \oplus \phi(x_j) = 0$ and $\phi(x_j) \oplus \phi(x_{j+1}) = 0$): In this case, we traverse the corresponding parity graphs as displayed in Figure 10.15. In Figure 10.15 (a), we have $\phi(x_i^l) \neq \phi(x_{i+1}^l) = 1$ and $\phi(x_j^l) = \phi(x_{j+1}^l) = 0$, whereas in (b), we have $\phi(x_i^l) \neq \phi(x_{i+1}^l) = 0$ and $\phi(x_j^l) = \phi(x_{j+1}^l) = 1$. This part of the tour has local length 6.

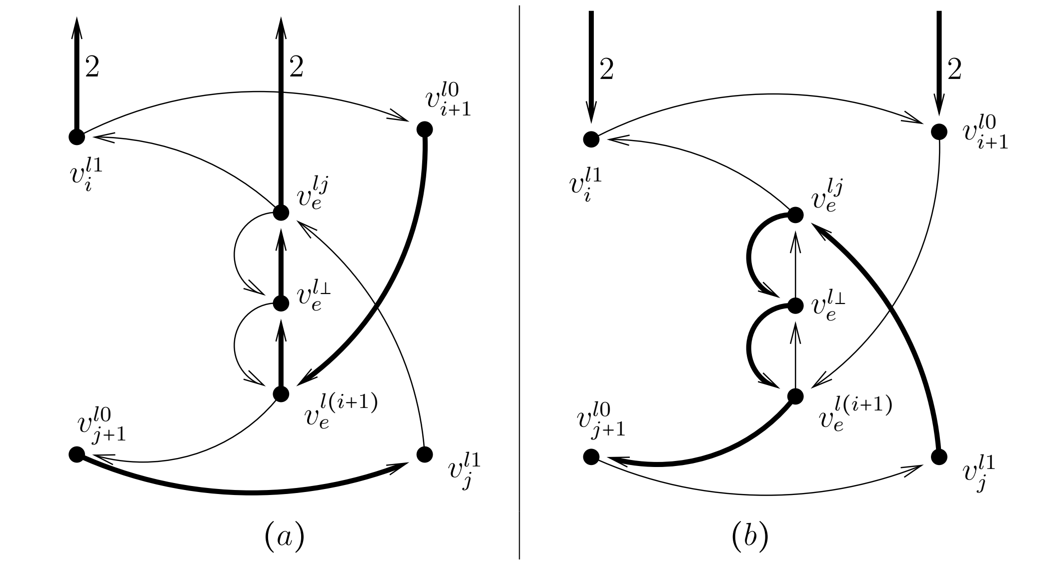


Figure 10.16: The scenario in the 8. Case.

8.Case ($\phi(x_i^l) \oplus \phi(x_{i+1}^l) = 1$, $\phi(x_i^l) \oplus \phi(x_j^l) = 1$ and $\phi(x_j^l) \oplus \phi(x_{j+1}^l) = 0$): In the last case, we traverse the corresponding parity graphs as depicted in Figure 10.16. In Figure 10.16 (a), we notice that $\phi(x_i^l) \neq \phi(x_{i+1}^l) = 0$ and $\phi(x_j^l) = \phi(x_{j+1}^l) = 0$, whereas in (b), we have $\phi(x_i^l) \neq \phi(x_{i+1}^l) = 1$ and $\phi(x_j^l) = \phi(x_{j+1}^l) = 1$. This part of the tour has local length 6.

As for the next step, we are going to analyze the length of the tour in graphs corresponding to equations with three variables.

Traversing Graphs for Equations with Three Variables

Let $\ell_c^3 \equiv x_i^l \oplus x_j^s \oplus x_t^k = 0$ be an equation with three variables in $\mathscr{L}$. Furthermore, let $x_i^l \oplus x_{i+1}^l = 0$, $x_j^s \oplus x_{j+1}^s = 0$ and $x_t^k \oplus x_{t+1}^k = 0$ be cycle equations in $\mathscr{L}$. For notational simplicity, we introduce $e = \{i, i+1\}$, $a = \{t, t+1\}$ and $b = \{j, j+1\}$. In Figure 10.17, we display the construction involving the graphs $\mathcal{G}_c^{3A}$, $\mathcal{P}_a^k$, $\mathcal{P}_b^s$, $\mathcal{P}_e^l$, $\mathcal{P}_i^l$ and $\mathcal{P}_{i+1}^l$. Exemplary, we depicted the connections of the graphs $\mathcal{G}_c^{3A}$, $\mathcal{P}_e^l$, $\mathcal{P}_i^l$ and $\mathcal{P}_{i+1}^l$ in this figure. We are going to construct the tour traversing the corresponding graphs and analyze the relation of the local length and the number of satisfied participating equations.

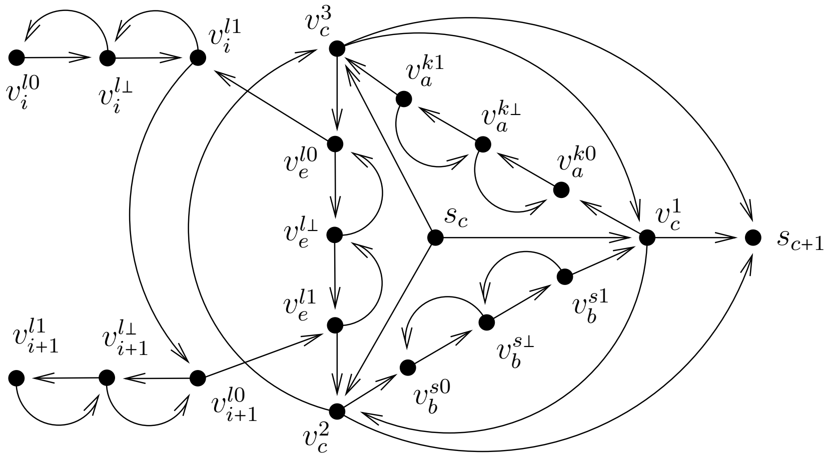


Figure 10.17: The graph $\mathcal{G}_c^{3A}$ with its connections to $\mathcal{P}_i^l$ and $\mathcal{P}_{i+1}^l$.

Recall from Lemma 10.5.1 that there is a simple path from s_c to s_{c+1} containing the vertices v_c^1, v_c^2 and v_c^3 in $\mathcal{G}_c^{3A}$ if and only if an even number of parity graphs $\mathcal{P} \in \{\mathcal{P}_a^k, \mathcal{P}_b^s, \mathcal{P}_e^l\}$ is traversed.

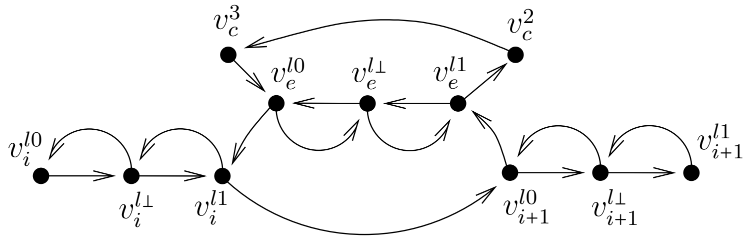


Figure 10.18: A part of the graphs corresponding to $x_i^l \oplus x_{i+1}^l = 0$ and $x_i^l \oplus x_j^s \oplus x_t^k = 0$.

The outer loop traverses the graph $\mathcal{G}_c^{3A}$ starting at s_c and ending in s_{c+1} while passing the vertices v_c^1, v_c^2 and v_c^3 in some order. If the inner loop of the tour contains an odd number of parity graphs $\mathcal{P} \in \{\mathcal{P}_a^k, \mathcal{P}_b^s, \mathcal{P}_e^l\}$, it is possible to construct a simple path from s_c to s_{c+1} containing the vertices that are not included in the inner loop of the tour. In particular, it passes an even number of remaining parity graphs, and we associate the local length $3 \cdot 3 + 4$ with this part. In the other case, we have to use a 2-arc yielding the local length 14.

Let us analyze the part of the tour traversing graphs corresponding to $x_i^l \oplus x_{i+1}^l = 0$. For this reason, we will examine the situation displayed in Figure 10.18. Let us begin with the case $\big(\phi(x_i^l) \oplus \phi(x_{i+1}^l) = 0\big)$.

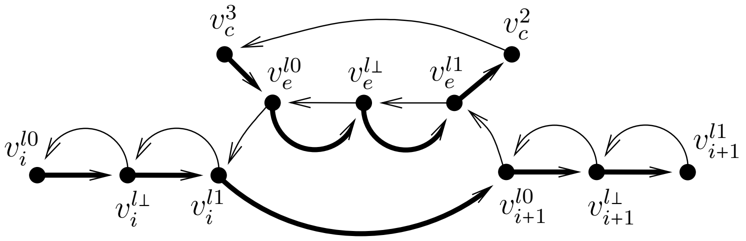


Figure 10.19: Case $\big(\phi(x_i^l) = \phi(x_{i+1}^l) = 1\big)$.

1. Case $(\phi(x_i^l) \oplus \phi(x_{i+1}^l) = 0)$:
If $\phi(x_i^l) = \phi(x_{i+1}^l) = 1$ holds, the tour uses the arc (v_i^{l1}, v_{i+1}^{l0}). Afterwards, the parity graph $\mathcal{P}_e^l$ will be traversed when the tour leads through the graph $\mathcal{G}_c^{3A}$. More precisely, it will use the path $v_c^3 \to v_e^{l0} \to v_e^{l\perp} \to v_e^{l1} \to v_c^2$. In Figure 10.19, we illustrated this part of the tour.
In the other case $(\phi(x_i^l) = \phi(x_{i+1}^l) = 0)$, we use the path $v_{i+1}^{l0} \to v_e^{l1} \to v_e^{l\perp} \to v_e^{l0} \to v_i^{l1}$. Afterwards, the tour contains the arc (v_c^2, v_c^3).

In both cases, we associate the local length 1 with this part of the tour.

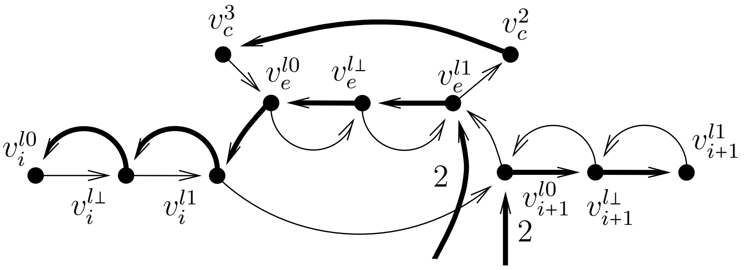


Figure 10.20: Case $(\phi(x_i^l) \neq \phi(x_{i+1}^l) = 1)$.

2. Case $(\phi(x^l_{i+1}) \oplus \phi(x^l_{i+1}) = 1)$**:**
Assuming $\phi(x^l_i) \neq \phi(x^l_{i+1}) = 1$, the tour uses a 2-arc entering v^{l1}_e and the path $v^{l1}_e \to v^{l\perp}_e \to v^{l0}_e \to v^{l1}_i$. Furthermore, we need another 2-arc in order to reach v^{l0}_{i+1}. The situation is depicted in Figure 10.20.
In the other case $(\phi(x^l_i) \neq \phi(x^l_{i+1}) = 0)$, we use 2-arcs leaving v^{l0}_{i+1} and v^{l1}_i. Afterwards, the tour uses the path $v^3_c \to v^{l0}_e \to v^{l\perp}_e \to v^{l1}_e \to v^2_c$ while traversing the graph $\mathcal{G}^{3A}_c$.

In both cases, we associate the local length 2 with this part of the tour.

In the next section, we are going to analyze the length of the tour in graphs corresponding to wheel border equations.

Traversing Graphs Corresponding to Wheel Border Equations

Let $\mathcal{W}_l$ be a wheel in $\mathscr{L}$ and $x^l_1 \oplus x^l_n = 0$ its wheel border equation. Recall that the variable x^l_n is also included in an equation with three variables. We assume that the equation is of the form $\bar{x}^l_n \oplus y \oplus z = 0$. We are going to describe the part of the tour passing through the graphs depicted in Figure 10.21 in dependence of the assigned values to the variables x^l_1 and x^l_n. Let us start with the case $\big(\phi(x^l_1) \oplus \phi(x^l_n) = 0\big)$.

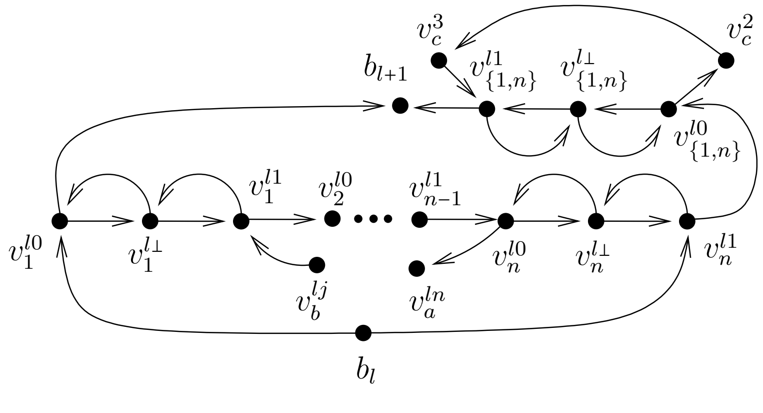


Figure 10.21: Traversing graphs corresponding to wheel border equations.

1. (Case $\phi(x^l_1) \oplus \phi(x^l_n) = 0$)**:**
The starting point of the tour passing through the graph corresponding to

$x_1^l \oplus x_n^l = 0$ is the vertex b_l. Given the values $\phi(x_1^l) = \phi(x_n^l)$, we use in each case the $\phi(x_1^l)$-traversal of the parity graphs $\mathcal{P}_1^l$ and $\mathcal{P}_n^l$ ending in b_{l+1}^l. Note that in the case $\phi(x_1^l) = \phi(x_n^l) = 0$, we use the 1-traversal of the parity graph $\mathcal{P}_{\{1,n\}}^l$. Exemplary, we display the situation $\phi(x_1^l) = \phi(x_n^l) = 1$ in Figure 10.22.

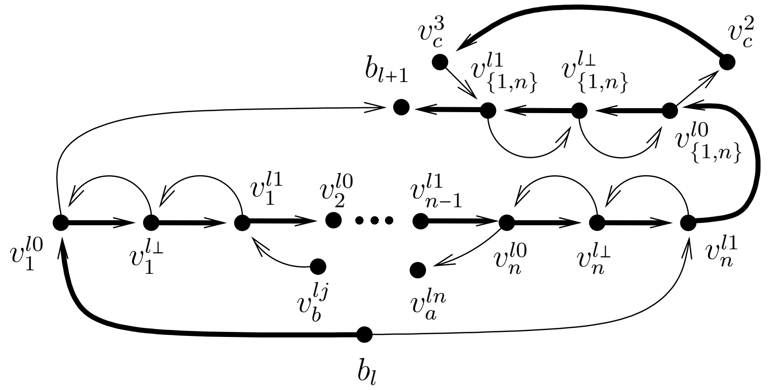


Figure 10.22: Case $(\phi(x_1^l) = \phi(x_n^l) = 1)$.

In both cases, we associated the local length 2 with this part of the tour.

2.Case $(\phi(x_1^l) \oplus \phi(x_n^l) = 1)$:

Given the assignment $\phi(x_1^l) \neq \phi(x_n^l) = 0$, we traverse the arc (b_l, v_1^{l0}). Due to the construction, we have to use 2-arcs to enter b_{l+1} and v_n^{l1} as displayed in Figure 10.23.

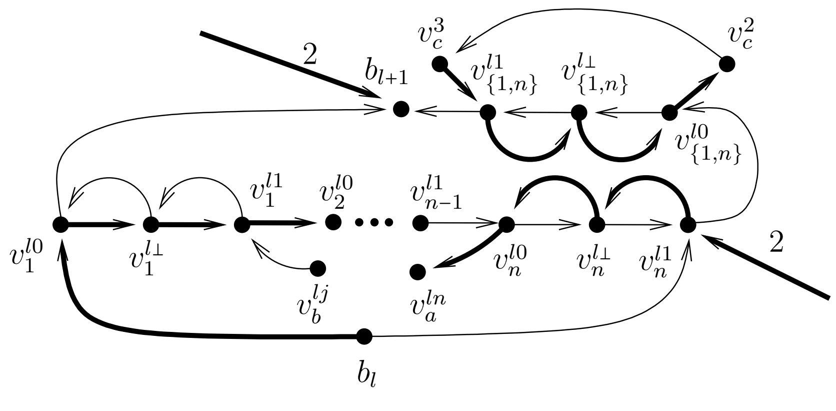


Figure 10.23: Case $(\phi(x_1^l) \neq \phi(x_n^l) = 0)$.

In the other case, we have to use a 2-arc in order to leave the vertices b_l and v_1^{l0}. In addition, the tour contains the path $v_n^{l1} \to v_{\{1,n\}}^{l0} \to v_{\{1,n\}}^{l\perp} \to v_{\{1,n\}}^{l1} \to b_{l+1}$.

Hence, in both cases, we associate the local length 3 with this part of the tour.

In summary, our analysis yields the following statement.

Lemma 10.7.1
Let $\mathscr{L}$ be an instance of th MAX-HYBRID-LIN2 *problem with n wheels, m_2 equations with two variables, m_3 equations with three variables and ϕ an assignment to the variables of $\mathscr{L}$ leaving u equations in $\mathscr{L}$ unsatisfied. Then, there is a tour in $\mathcal{G}_{\mathscr{L}}$ with length at most $n + 1 + 3m_2 + 13m_3 + u$.*

10.7.4 Constructing the Assignment from a Tour

Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 problem, $\mathcal{G}_{\mathscr{L}}$ the associated instance of the (1,2)-ATSP problem and σ a tour in $\mathcal{G}_{\mathscr{L}}$. We are going to define the corresponding assignment ψ_σ to the variables in $\mathscr{L}$. In addition to it, we establish a connection between the length of σ and the number of satisfied equations by ψ_σ. In order to define an assignment, we first introduce the notion of consistent tours in $\mathcal{G}_{\mathscr{L}}$.

Definition 10.7.1 (Consistent Tour)
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem and $\mathcal{G}_{\mathscr{L}}$ the associated instance of the* (1,2)-ATSP *problem. A tour in $\mathcal{G}_{\mathscr{L}}$ is called consistent if it uses only $0/1$-traversals of all parity graphs that are in contained in $\mathcal{G}_{\mathscr{L}}$.*

Due to the following lemma, we may assume that the underlying tour is consistent.

Lemma 10.7.2
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem and $\mathcal{G}_{\mathscr{L}}$ the associated instance of the* (1,2)-ATSP *problem. Any tour σ in $\mathcal{G}_{\mathscr{L}}$ can be transformed in polynomial time into a consistent tour π in $\mathcal{G}_{\mathscr{L}}$ without increasing*

the length.

Proof of Lemma 10.7.2.
For every parity graph contained in $\mathcal{G}_{\mathscr{L}}$, it can be seen by considering all possibilities exhaustively that any tour in $\mathcal{G}_{\mathscr{L}}$, that is not using the corresponding 0/1-traversals, can be modified into a tour with at most the same number of 2-arcs. The less obvious cases are shown in Figure 10.24 and Figure 10.25 ■

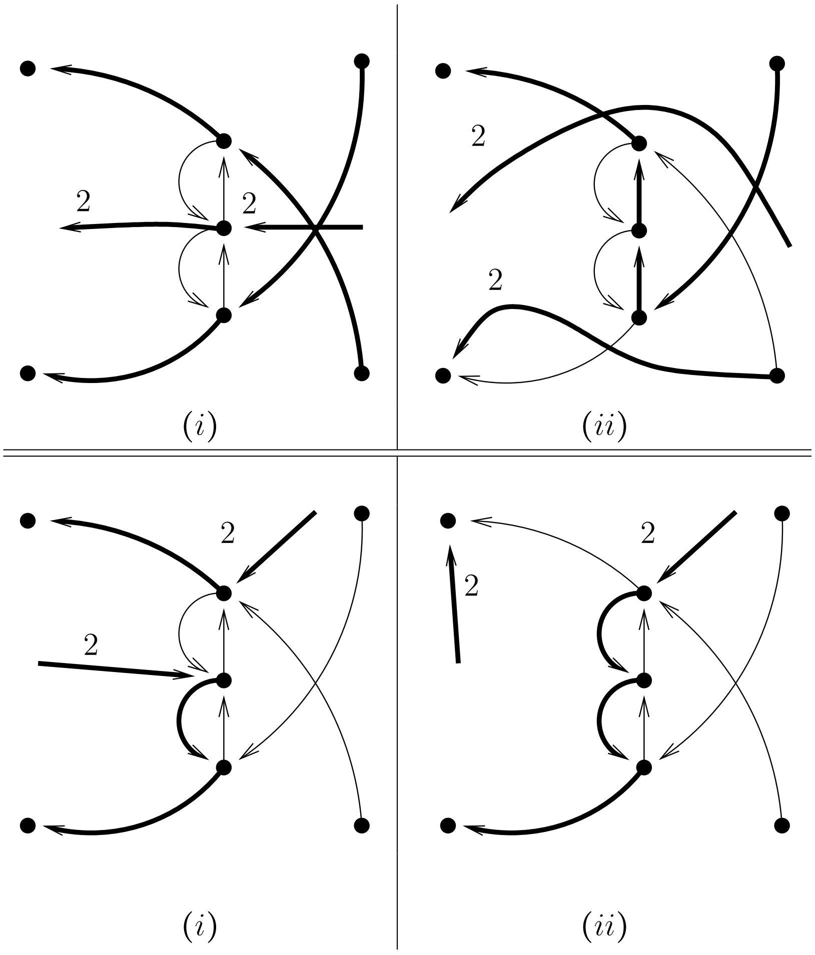


Figure 10.24: Situations before (i) and after the transformation (ii).

In the following, we assume that the underlying tour σ is consistent. Let us now define the corresponding assignment ψ_σ given σ.

Definition 10.7.2 (Assignment ψ_σ)
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem, $\mathcal{G}_{\mathscr{L}}$ the associated instance of the* (1,2)-ATSP *problem. Given a consistent tour σ in $\mathcal{G}_{\mathscr{L}}$, the corresponding assignment ψ_σ is defined as follows.*

$$\psi_\sigma(x_i^l) = \begin{cases} 1 & \textit{if } \sigma \textit{ uses a } 1\textit{-traversal of } \mathcal{P}_i^l \\ 0 & \textit{otherwise} \end{cases}$$

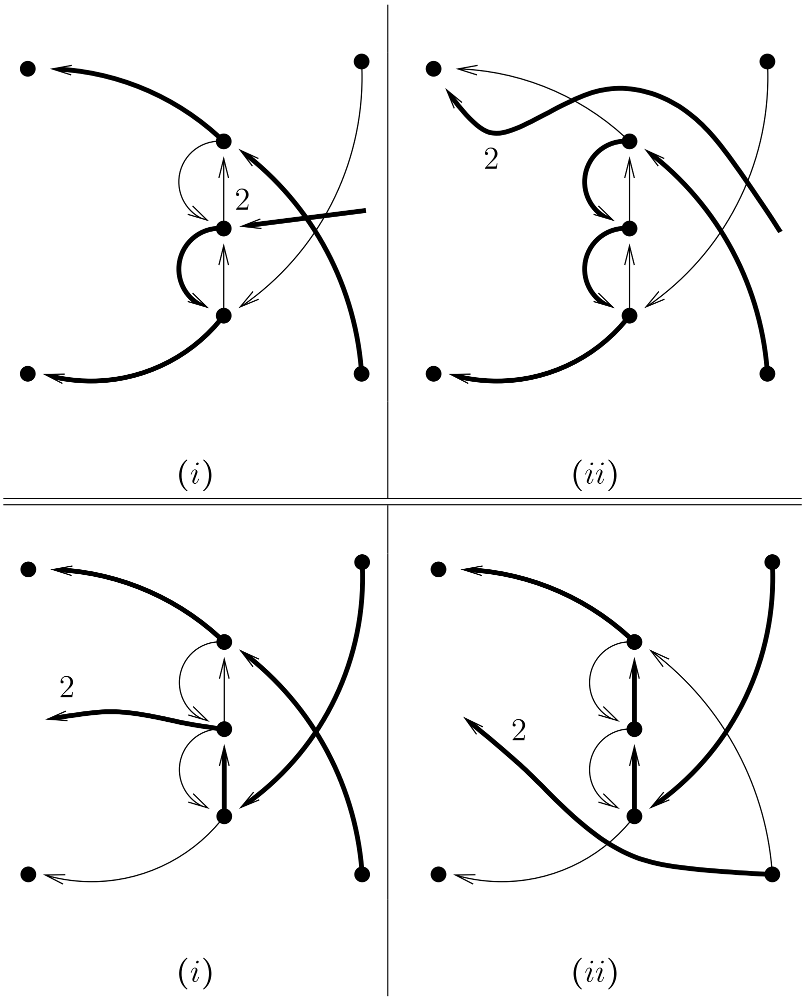


Figure 10.25: Situations before (i) and after the transformation (ii).

We are going to analyze the local length of σ in dependence of the number of corresponding satisfied equations by ψ_σ. In some cases, we will have to modify the underlying tour improving in this way on the number of satisfied equations by the corresponding assignment ψ_σ. Let us start with the analysis.

Transforming σ in Graphs for Matching Equations

Given the equations $x_i^l \oplus x_{i+1}^l = 0$, $x_i^l \oplus x_j^l = 0$, $x_j^l \oplus x_{j+1}^l = 0$ and a tour σ, we are going to construct an assignment in dependence of σ. In particular, we analyze the relation between the length of the tour and the number of satisfied equations by ψ_σ.

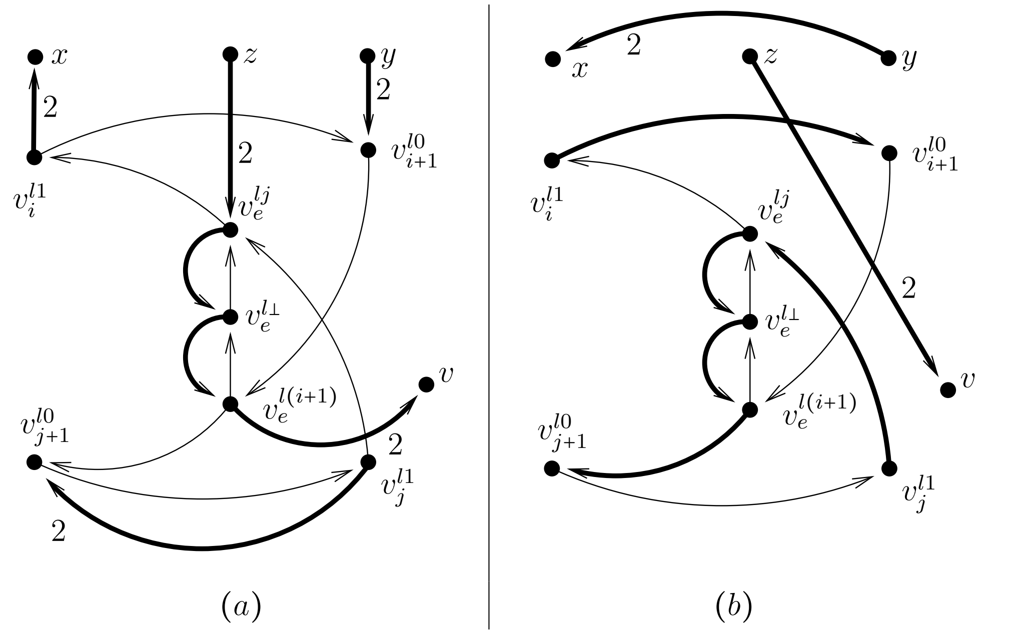


Figure 10.26: The scenario in the 1. Case.

1.Case ($\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_{i+1}^l) = 0$, $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_j^l) = 0$ & $\psi_\sigma(x_j^l) \oplus \psi_\sigma(x_{j+1}^l) = 0$): Given $\psi_\sigma(x_i^l) = \psi_\sigma(x_j^l) = \psi_\sigma(x_j^l) = \psi_\sigma(x_{j+1}^l) = 1$, it is possible to transform the underlying tour such that no 2-arcs enter or leave the vertices v_i^{l1}, v_{i+1}^{l0}, v_{j+1}^{l0}, v_e^{lj}, $v_e^{l(i+1)}$ and v_j^{l1}. Exemplary, we display in Figure 10.26 such a transformation, where Figure 10.26 (a) and Figure 10.26 (b) illustrate the underlying tour σ and the transformed tour σ', respectively. The case

$\psi_\sigma(x_i^l) = \psi_\sigma(x_{i+1}^l) = \psi_\sigma(x_j^l) = \psi_\sigma(x_{j+1}^l) = 0$ can be discussed analogously. In both cases, we obtain the local length 5 for this part of σ while ψ_σ satisfies all 3 equations.

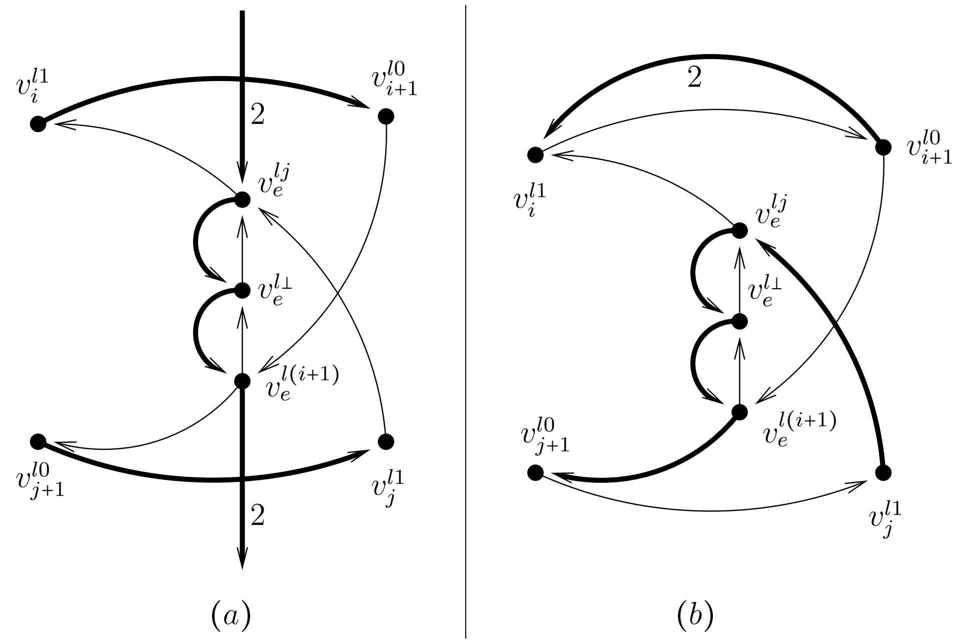


Figure 10.27: The scenario in the 2. Case.

2. Case ($\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_{i+1}^l) = 0$, $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_j^l) = 1$ & $\psi_\sigma(x_j^l) \oplus \psi_\sigma(x_{j+1}^l) = 0$): Assuming $\psi_\sigma(x_i^l) = \psi_\sigma(x_{i+1}^l) = 1$ and $\psi_\sigma(x_j^l) = \psi_\sigma(x_{j+1}^l) = 0$, we are able to transform the tour such that it uses the arcs (v_i^{l1}, v_{i+1}^{l0}) and (v_{j+1}^{l0}, v_j^{l1}). Due to the construction and our assumption, the tour cannot traverse the arcs (v_j^{l1}, v_e^{lj}), (v_e^{lj}, v_i^{l1}), $(v_e^{l(i+1)}, v_{j+1}^{l0})$ and $(v_{i+1}^{l0}, v_e^{l(i+1)})$. Consequently, we have to use 2-arcs entering and leaving the parity graph $\mathcal{P}_e^l$. The situation is displayed in Figure 10.27 (a). We associate only the cost of one 2-arc yielding the local length 6, which corresponds to the fact that ψ_σ leaves the equation $x_i^l \oplus x_j^l = 0$ unsatisfied.

Note that a similar situation holds in the case when $\psi_\sigma(x_i^l) = \psi_\sigma(x_{i+1}^l) = 0$ and $\psi_\sigma(x_j^l) = \psi_\sigma(x_{j+1}^l) = 1$ (cf. Figure 10.27 (b)).

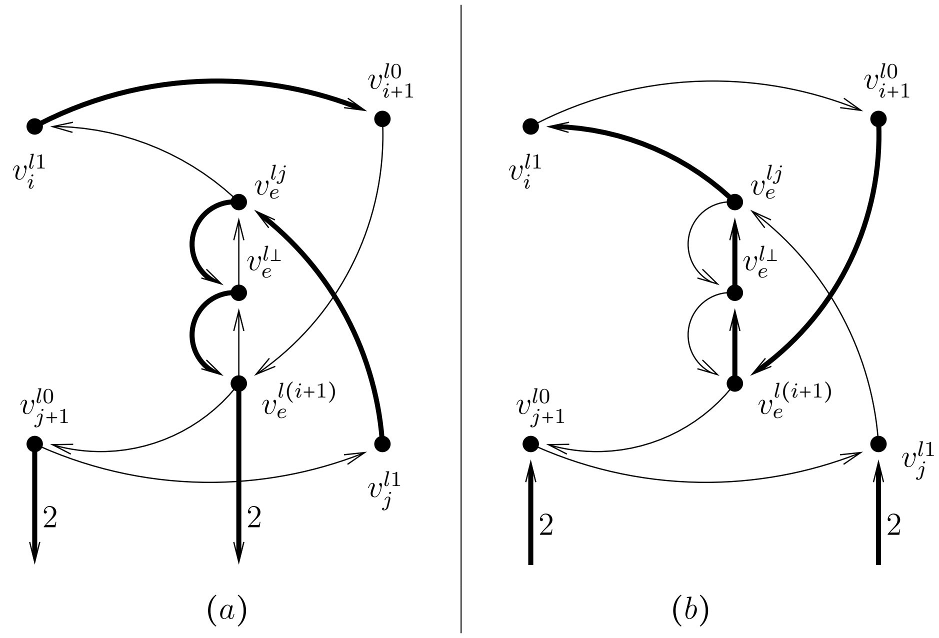


Figure 10.28: The scenario in the 3. Case.

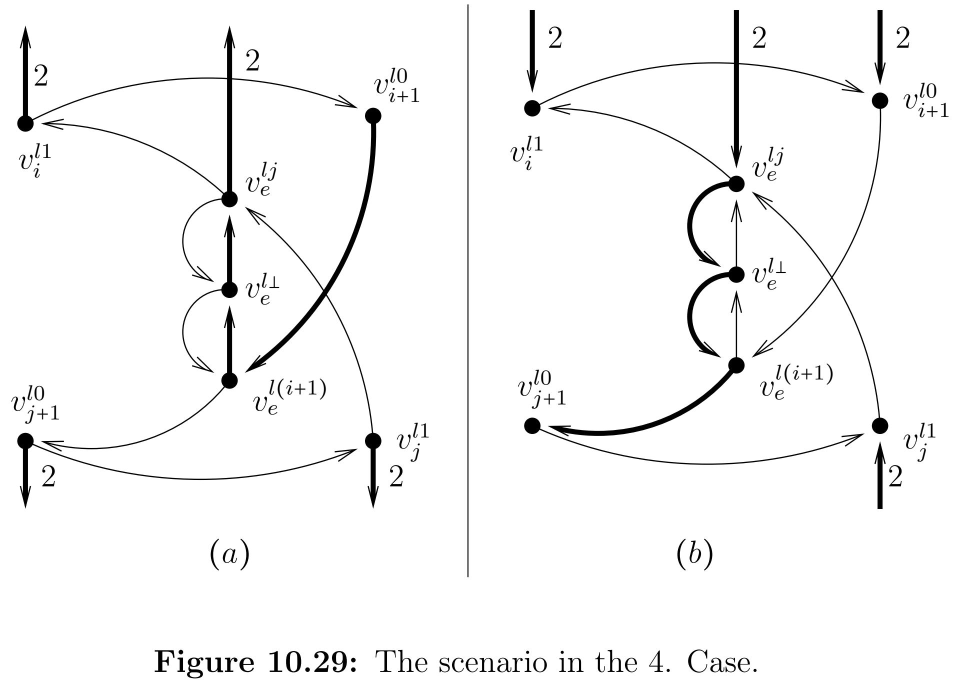


Figure 10.29: The scenario in the 4. Case.

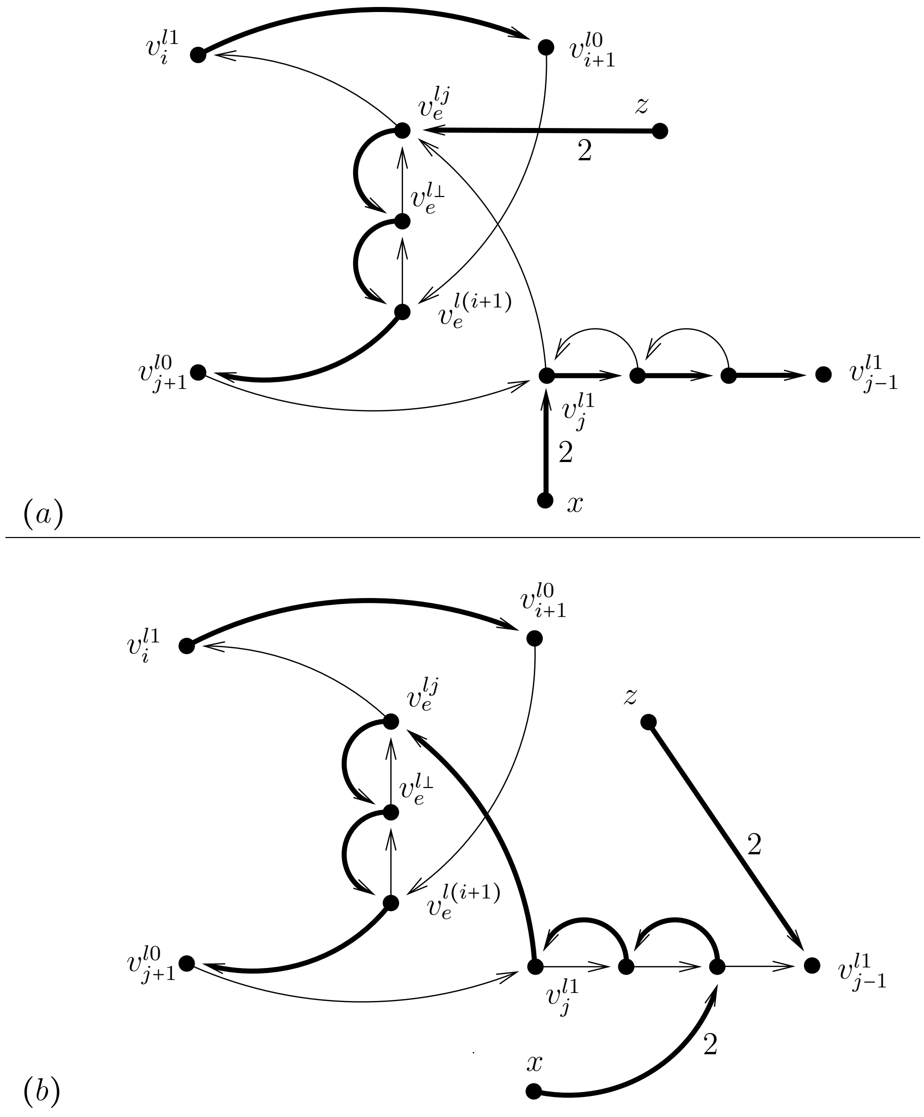


Figure 10.30: The scenario in the 5. Case.

3.Case ($\psi_\sigma(x_i^l)\oplus\psi_\sigma(x_{i+1}^l) = 0$, $\psi_\sigma(x_i^l)\oplus\psi_\sigma(x_j^l) = 0$ & $\psi_\sigma(x_j^l)\oplus\psi_\sigma(x_{j+1}^l) = 1$): Let us start with the case $\psi_\sigma(x_i^l) = \psi_\sigma(x_{i+1}^l) = 1$ and $\psi_\sigma(x_j^l) \neq \psi_\sigma(x_{j+1}^l) = 0$. The situation is displayed in Figure 10.28 (a). Due to the construction, we are able to transform σ such that it uses the arc (v_i^{l1}, v_{i+1}^{l0}). Note that the tour cannot traverse the arcs $(v_e^{l(i+1)}, v_{j+1}^{l0})$ and (v_{j+1}^{l0}, v_j^{l1}). Hence, we are forced to use two 2-arcs increasing the cost by 2. All in all, we obtain the local length 6.

The case $\psi_\sigma(x_i^l) = \psi_\sigma(x_{i+1}^l) = 0$ and $\psi_\sigma(x_j^l) \neq \psi_\sigma(x_{j+1}^l) = 1$ can be analyzed analogously (cf. Figure 10.28 (b)). Similar arguments can be applied when $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_{i+1}^l) = 1$, $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_j^l) = 0$ and $\psi_\sigma(x_j^l) \oplus \psi_\sigma(x_{j+1}^l) = 0$.

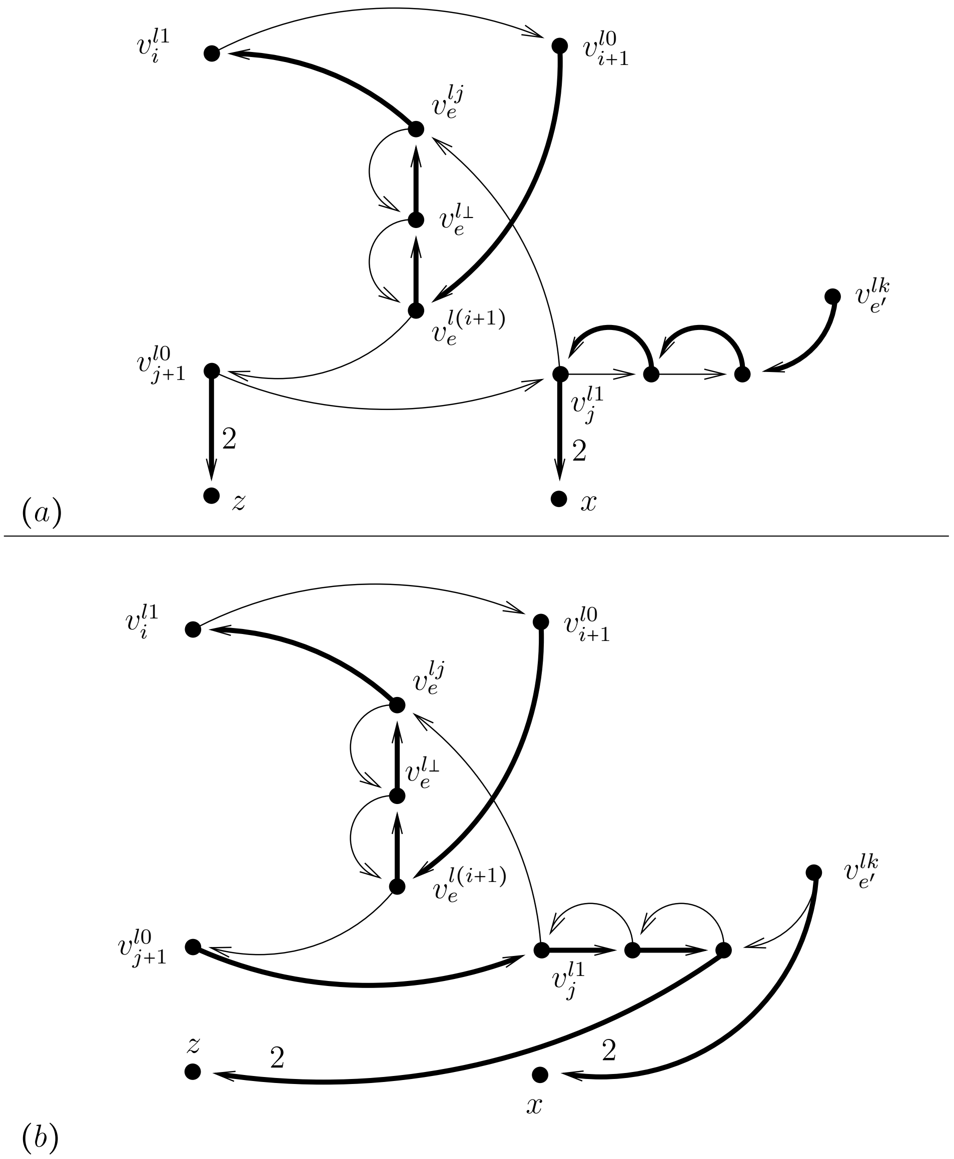


Figure 10.31: 5.Case ($\psi_\sigma(x_i^l) = \psi_\sigma(x_{i+1}^l) = 0$ & $\psi_\sigma(x_j^l) \neq \psi_\sigma(x_{j+1}^l) = 0$).

4.Case $(\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_{i+1}^l) = 1$**,** $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_j^l) = 0$ **&** $\psi_\sigma(x_j^l) \oplus \psi_\sigma(x_{j+1}^l) = 1)$**:** Given $\psi_\sigma(x_i^l) \neq \psi_\sigma(x_{i+1}^l) = 0$ and $\psi_\sigma(x_j^l) \neq \psi_\sigma(x_{j+1}^l) = 0$, we are able to transform the tour such that it uses the arc $(v_{i+1}^{l0}, v_e^{l(i+1)})$. This situation is depicted in Figure 10.29 (a). Notice that we are forced to use four 2-arcs in order to connect all vertices. Consequently, it yields the local length 7.

The case, in which $\psi_\sigma(x_i^l) \neq \psi_\sigma(x_{i+1}^l) = 0$ and $\psi_\sigma(x_j^l) \neq \psi_\sigma(x_{j+1}^l) = 0$ holds, is displayed in Figure 10.29 (b) and can be discussed analogously.

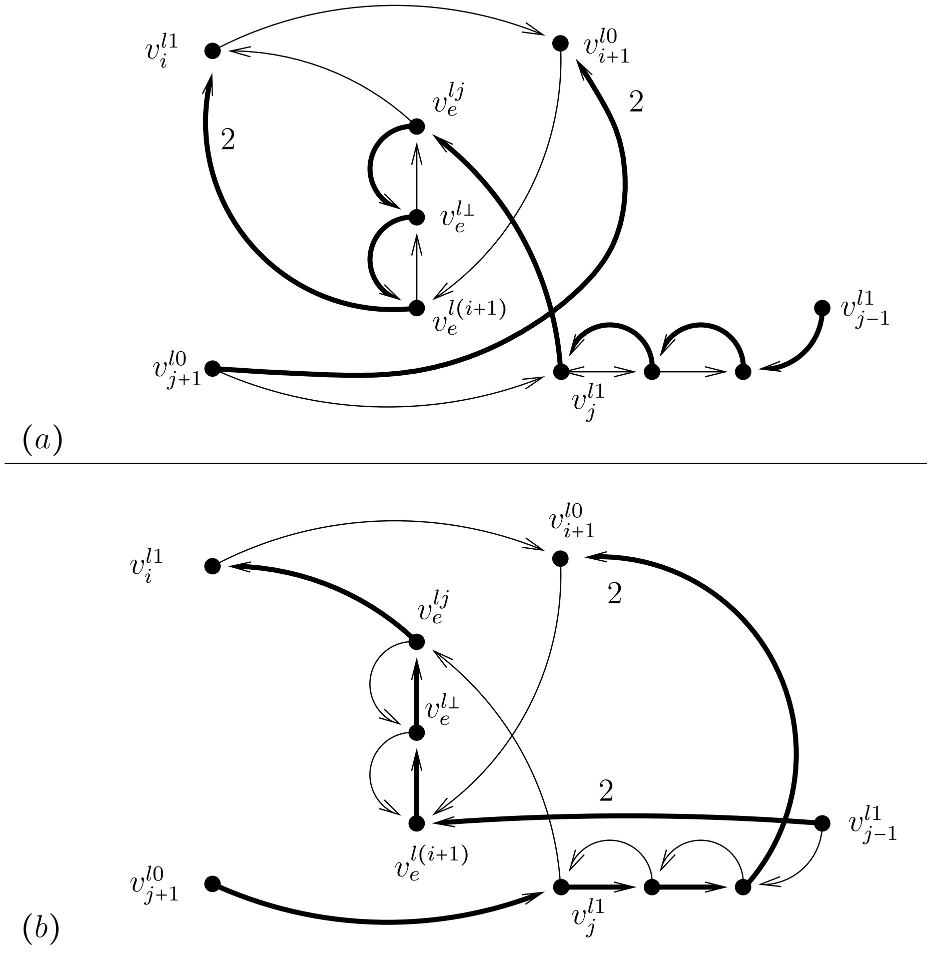


Figure 10.32: 6.Case $(\psi_\sigma(x_i^l) \neq \psi_\sigma(x_{i+1}^l) = 1$ and $\psi_\sigma(x_j^l) \neq \psi_\sigma(x_{j+1}^l) = 0)$.

5.Case $(\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_{i+1}^l) = 0$**,** $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_j^l) = 1$ **&** $\psi_\sigma(x_j^l) \oplus \psi_\sigma(x_{j+1}^l) = 1)$**:** Let the tour σ be characterized by $\psi_\sigma(x_i^l) = \psi_\sigma(x_{i+1}^l) = 1$ and $\psi_\sigma(x_j^l) \neq$

$\psi_\sigma(x^l_{j+1}) = 1$. Then, we transform σ in such a way that we are able to use the arc (v^{l1}_i, v^{l0}_{i+1}). The corresponding situation is illustrated in Figure 10.30 (a). In order to change the value of $\psi_\sigma(x^l_j)$, we transform the tour by traversing the parity graph $\mathcal{P}^l_j$ in the other direction and obtain $\psi_\sigma(x^l_j) = 1$. This transformation induces a tour with at most the same cost. On the other hand, the corresponding assignment ψ_σ satisfies at least $2-1$ more equations since $x^l_j \oplus x^l_{j-1} = 0$ could be unsatisfied. In this case, we associate the local cost 6 with σ.

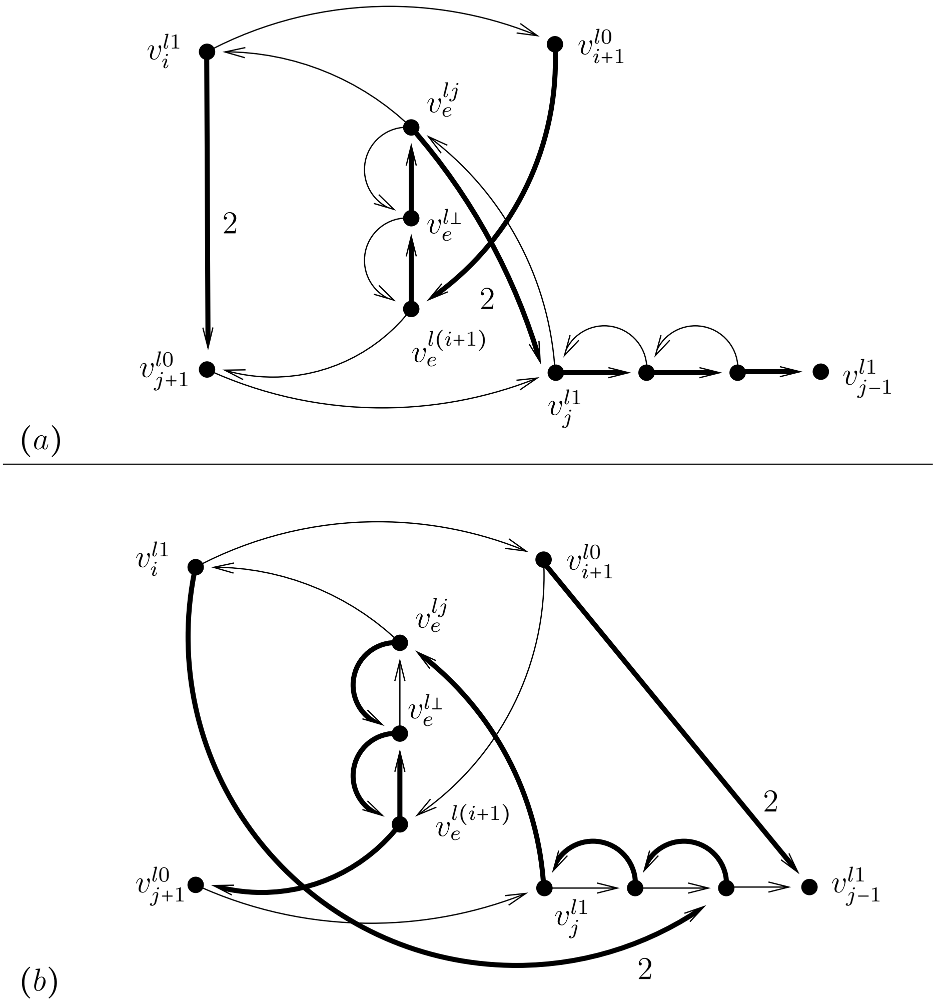


Figure 10.33: 6.Case $(\psi_\sigma(x^l_i) \neq \psi_\sigma(x^l_{i+1}) = 0$ and $\psi_\sigma(x^l_j) \neq \psi_\sigma(x^l_{j+1}) = 1)$.

In the other case, in which $\psi_\sigma(x_i^l) = \psi_\sigma(x_{i+1}^l) = 0$ and $\psi_\sigma(x_j^l) \neq \psi_\sigma(x_{j+1}^l) = 0$ holds, we may argue similarly. The transformation is depicted in Figure 10.31 $(a) - (b)$.

6.Case ($\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_{i+1}^l) = 1$, $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_j^l) = 1$ & $\psi_\sigma(x_j^l) \oplus \psi_\sigma(x_{j+1}^l) = 1$): Given a tour σ with $\psi_\sigma(x_i^l) \neq \psi_\sigma(x_{i+1}^l) = 1$ and $\psi_\sigma(x_j^l) \neq \psi_\sigma(x_{j+1}^l) = 0$, we transform σ such that it traverses the parity graph $\mathcal{P}_j^l$ in the opposite direction meaning $\psi_\sigma(x_j^l) = 0$ (cf. Figure 10.32). This transformation enables us to use the arc (v_{j+1}^{l0}, v_j^{l1}). Furthermore, it yields at least one more satisfied equation in $\mathscr{L}$. In order to connect the remaining vertices, we are forced to use at least two 2-arcs. In summary, we associate the local length 7 with this situation in conformity with the at most 2 unsatisfied equations by ψ_σ.

If we are given a tour σ with $\psi_\sigma(x_i^l) \neq \psi_\sigma(x_{i+1}^l) = 0$ and $\psi_\sigma(x_j^l) \neq \psi_\sigma(x_{j+1}^l) = 1$, we obtain the situation displayed in Figure 10.33 (a). By applying local transformations without increasing the length of the underlying tour, we obtain the scenario displayed in Figure 10.33 (b). We argue that the associated local length of the tour is 7. The case, in which $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_{i+1}^l) = 1$, $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_j^l) = 1$ and $\psi_\sigma(x_j^l) \oplus \psi_\sigma(x_{j+1}^l) = 0$ holds, can be discussed analogously.

Transforming σ in Graphs for Equations With Three Variables

Let $\ell_c^3 \equiv x_i^l \oplus x_j^s \oplus x_k^r = 0$ be an equation with three variables, $\mathcal{W}_l$ a wheel and $x_i^l \oplus x_{i+1}^l = 0$ a cycle equation in $\mathscr{L}$. For notational simplicity, we set $e = \{i, i+1\}$. We are going to analyze the relation between the number of satisfied equations by ψ_σ and the local length of σ in the graphs $\mathcal{P}_i^l$, $\mathcal{P}_{i+1}^l$, $\mathcal{P}_e^l$ and $\mathcal{G}_c^{3A}$. First, we transform the tour traversing the graphs $\mathcal{P}_i^l$, $\mathcal{P}_{i+1}^l$ and $\mathcal{P}_e^l$ such that it uses the $\psi_\sigma(x_i^l)$-traversal of $\mathcal{P}_e^l$. Afterwards, due to the construction of $\mathcal{G}_c^{3A}$ and Lemma 10.5.1, the outer loop of the tour can be transformed such that it has local length $3 \cdot 3 + 4$ if it passes an even number of parity graphs $\mathcal{P} \in \{\mathcal{P}_e^l, \mathcal{P}_a^k, \mathcal{P}_b^s\}$ by using a simple path through $\mathcal{G}_c^{3A}$. Otherwise, it yields a local length of $13 + 1$. Let us start with the case

$\psi_\sigma(x_i^l) = 1$ and $\psi_\sigma(x_{i+1}^l) = 1$.

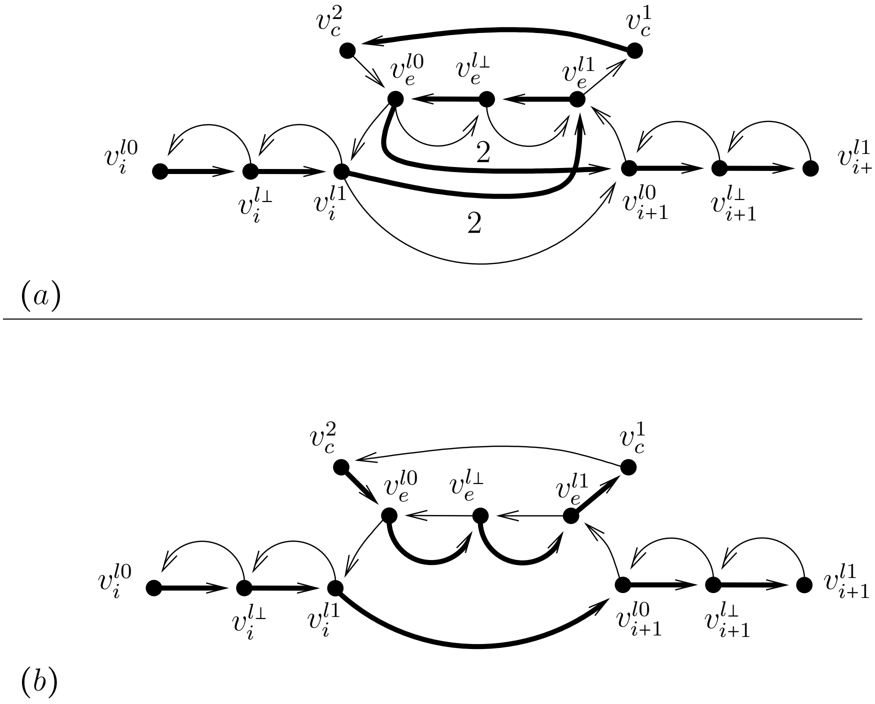


Figure 10.34: 1.Case ($\psi_\sigma(x_i^l) = 1$ and $\psi_\sigma(x_{i+1}^l) = 1$).

1. Case ($\psi_\sigma(x_i^l) = 1$ and $\psi_\sigma(x_{i+1}^l) = 1$):
In Figure 10.34 (a) and (b), we display the tour passing through $\mathcal{P}_i^l$, $\mathcal{P}_{i+1}^l$ and $\mathcal{P}_e^l$ with $\psi_\sigma(x_i^l) = 1$ and $\psi_\sigma(x_{i+1}^l) = 1$ before and after the transformation, respectively. It is possible to transform the tour σ without increasing the length such that it traverses the arc (v_i^{l1}, v_{i+1}^{l0}). In the outer loop, the tour may use at least one of the arcs (v_c^2, v_e^{l0}) and (v_e^{l1}, v_c^1) depending on the parity check in $\mathcal{G}_c^{3A}$. We associate the local length 1 with this part of the tour.

2. Case ($\psi_\sigma(x_i^l) = 0$ and $\psi_\sigma(x_{i+1}^l) = 0$):
In Figure 10.35, we display the underlying scenario with $\psi_\sigma(x_i^l) = 0$ and $\psi_\sigma(x_{i+1}^l) = 0$. The transformed tour uses the 0-traversal of the parity graph $\mathcal{P}_e^l$. The vertices v_c^2 and v_c^1 are connected via a 2-arc. We assign the local

length 1 to this part of the tour.

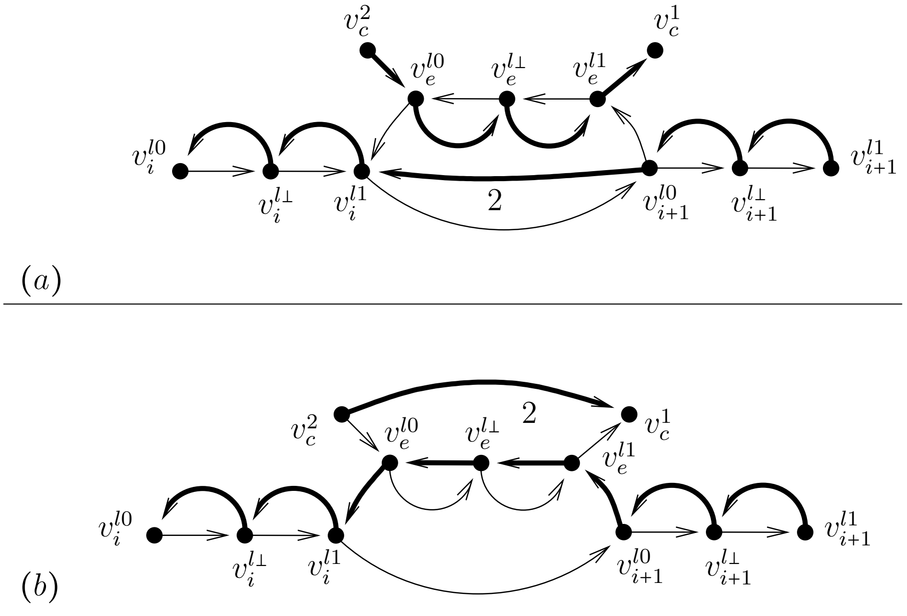


Figure 10.35: 2.Case ($\psi_\sigma(x_i^l) = 0$ and $\psi_\sigma(x_{i+1}^l) = 0$).

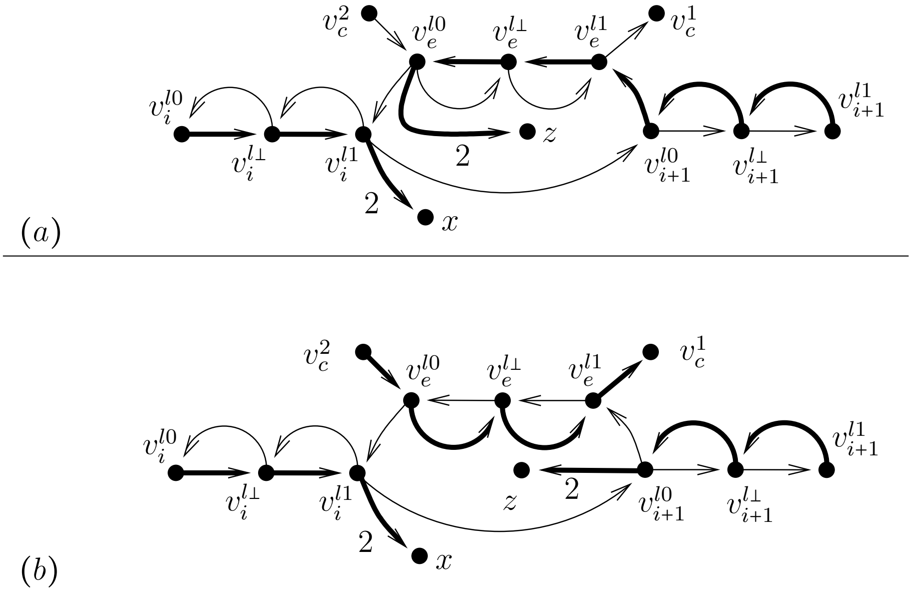


Figure 10.36: The scenario in the 3. Case with $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_j^s) \oplus \psi_\sigma(x_k^r) = 0$.

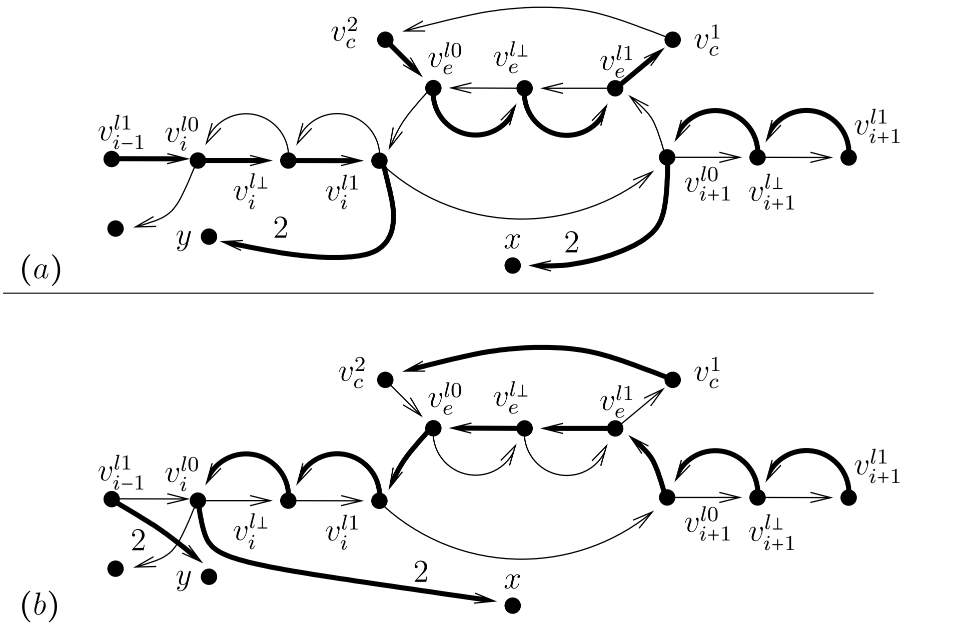


Figure 10.37: The scenario in the 3. Case with $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_j^s) \oplus \psi_\sigma(x_k^r) = 1$.

3.Case ($\psi_\sigma(x_i^l) = 1$ and $\psi_\sigma(x_{i+1}^l) = 0$):
Let us assume that $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_j^s) \oplus \psi_\sigma(x_k^r) = 0$ holds. Hence, it is possible to transform the tour such that it uses the path $v_c^2 \to v_e^{l0} \to v_e^{l\perp} \to v_e^{l1} \to v_c^1$ and thus, the 0-traversal of the parity graph $\mathcal{P}_e^l$ as displayed in Figure 10.36.

In the other case, namely $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_j^s) \oplus \psi_\sigma(x_k^r) = 1$, we will change the value of $\psi_\sigma(x_i^l)$ achieving in this way at least $2 - 1$ more satisfied equation. Let us examine the scenario in Figure 10.37. The tour uses the 0-traversal of the parity graph $\mathcal{P}_e^l$, which enables σ to pass the parity check in $\mathcal{G}_c^{3A}$.

In both cases, we associate the local length 2 in conformity with the at most one unsatisfied equation by ψ_σ.

4.Case ($\psi_\sigma(x_i^l) = 0$ and $\psi_\sigma(x_{i+1}^l) = 1$):
Assuming $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_j^s) \oplus \psi_\sigma(x_k^r) = 0$ and the scenario depicted in Figure 10.38 (a), the tour will be modified such that the parity graphs $\mathcal{P}_i^l$ and $\mathcal{P}_e^l$ are traversed in the same direction. Since we have $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_j^s) \oplus \psi_\sigma(x_k^r) = 0$, we are able to uncouple the parity graph $\mathcal{P}_e^l$

from the tour through $\mathcal{G}_c^{3A}$ without increasing its length. We display the transformed tour in Figure 10.38 (b).

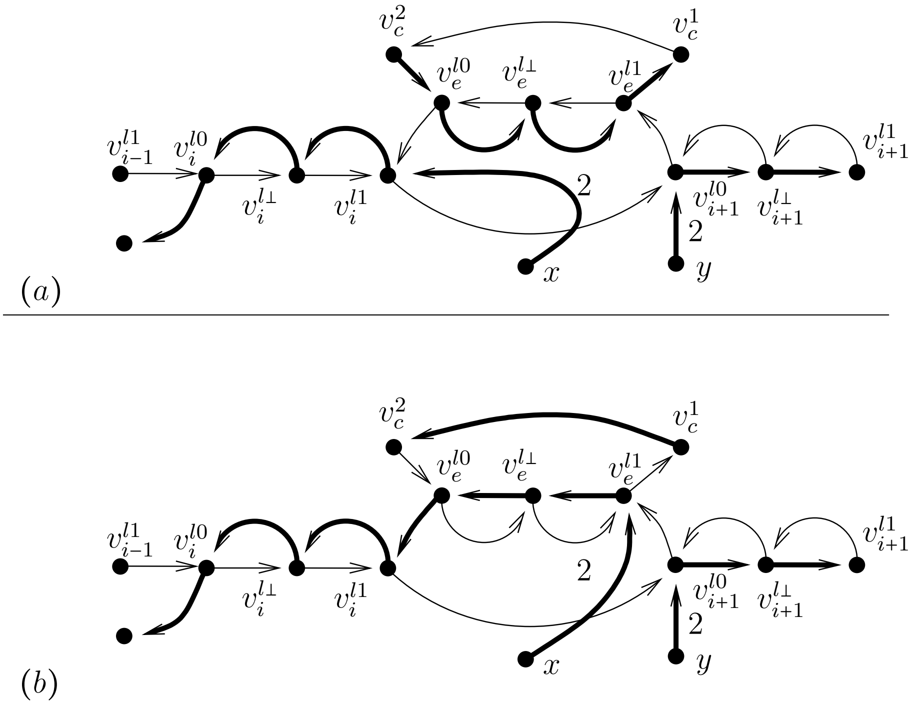


Figure 10.38: The scenario in the 4. Case with $\psi_\sigma(x_i^l) = 0$, $\psi_\sigma(x_{i+1}^l) = 1$ and $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_j^s) \oplus \psi_\sigma(x_k^r) = 0$.

Assuming $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_j^s) \oplus \psi_\sigma(x_k^r) = 1$ and the scenario displayed in Figure 10.39 (a), we transform σ such that the parity graph $\mathcal{P}_e^l$ is traversed when σ is passing through $\mathcal{G}_c^{3A}$, that is, the path $v_c^2 \to v_e^{l0} \to v_e^{l\perp} \to v_e^{l1} \to v_c^1$ is a part of the tour. In addition, we change the value of $\psi_\sigma(x_i^l)$ yielding at least $2-1$ more satisfied equation. The transformed tour is displayed in Figure 10.39 (b). In both cases, we associate the local length 2 with σ. On the other hand, ψ_σ leaves at most one equation unsatisfied.

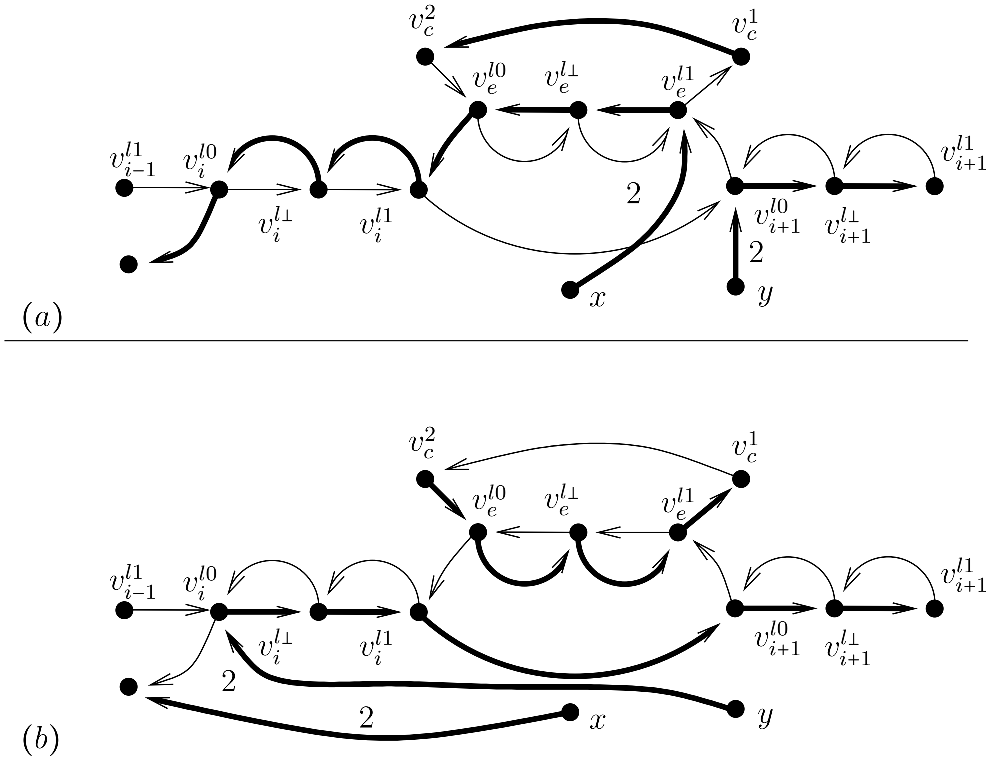


Figure 10.39: The scenario in the 4. Case with $\psi_\sigma(x_i^l) = 0$, $\psi_\sigma(x_{i+1}^l) = 1$ and $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_j^s) \oplus \psi_\sigma(x_k^r) = 1$.

Transforming σ in Graphs for Wheel Border Equations

Let $\mathcal{W}_l$ be a wheel in $\mathscr{L}$ and $x_1^l \oplus x_n^l = 0$ its wheel border equation. Furthermore, let $\ell_c^3 \equiv x_n^l \oplus x_j^s \oplus x_k^r = 1$ be an equation with three variables contained in $\mathscr{L}$. We are going to transform a given tour σ passing through the graph corresponding to $x_1^l \oplus x_n^l = 0$ such that it will have the local length 2 if $x_1^l \oplus x_n^l = 0$ is satisfied by ψ_σ and 3, otherwise. In each case, we modify σ such that it uses the $\psi(x_n^l)$-traversal of $\mathcal{P}_{\{1,n\}}^l$. Afterwards, σ will be checked in $\mathcal{G}_c^{3A}$ whether it passes the parity test.

Let us begin with the analysis starting with the case $\psi_\sigma(x_1) = 1$ and $\psi_\sigma(x_n) = 1$.

1.Case ($\psi_\sigma(x_1) = 1$ and $\psi_\sigma(x_n) = 1$):

Let us assume that ψ_σ leaves ℓ_c^3 unsatisfied meaning $\psi_\sigma(x_n^l) \oplus \psi_\sigma(x_j^s) \oplus \psi_\sigma(x_k^r) = 0$.

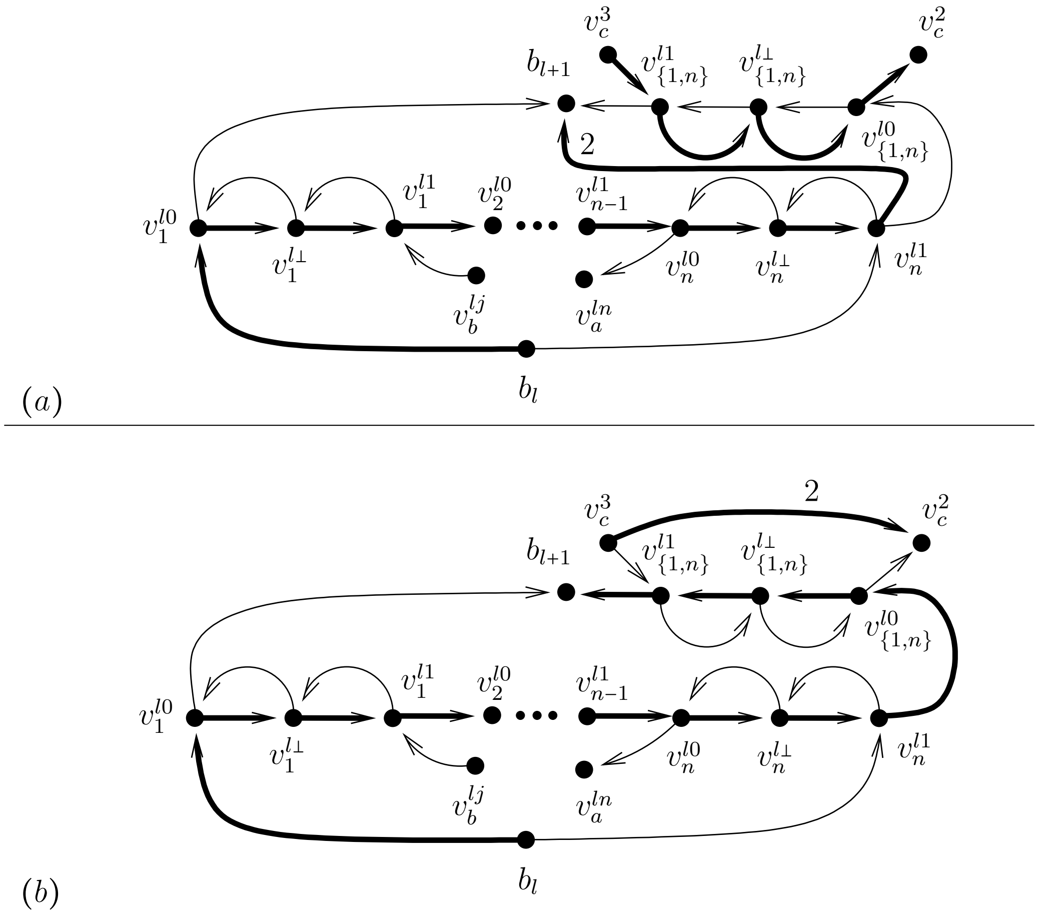


Figure 10.40: 1.Case ($\psi_\sigma(x_1) = 1$ and $\psi_\sigma(x_n) = 1$).

In addition to it, we assume that the path $v_c^3 \to v_{\{1,n\}}^{l1} \to v_{\{1,n\}}^{l\perp} \to v_{\{1,n\}}^{l0} \to v_c^2$ is a part of σ. Note that σ fails the parity check in $\mathcal{G}_c^{3A}$ if $v_c^3 \to v_{\{1,n\}}^{l1} \to v_{\{1,n\}}^{l\perp} \to v_{\{1,n\}}^{l0} \to v_c^2$ is not contained in σ. First, we modify the tour such that it includes the arc (b_l, v_1^{l0}). Furthermore, we may assume that v_n^{l1} and b_{l+1} are connected via a 2-arc. We obtain the scenario depicted in Figure 10.40 (a). As for the next step, we transform σ such that it contains the arcs $(v_n^{l1}, v_{\{1,n\}}^{l0})$ and $(v_{\{1,n\}}^{l1}, b_{l+1})$. Consequently, we use the 1-traversal of the parity graph $\mathcal{P}^{l}_{\{1,n\}}$ and connect v_c^3 and v_c^2 via a 2-arc. The modified tour is displayed in Figure 10.40 (b). If ψ_σ satisfies ℓ_c^3 and σ contains the path $v_c^3 \to v^{l1}_{\{1,n\}} \to v^{l\perp}_{\{1,n\}} \to v^{l0}_{\{1,n\}} \to v_c^2$, we modify σ in $\mathcal{G}_c^{3A}$ such that it passes the parity test in $\mathcal{G}_c^{3A}$ and contains the arc (v_c^2, v_c^3). In both cases, we associate the local length 2 with this part of σ.

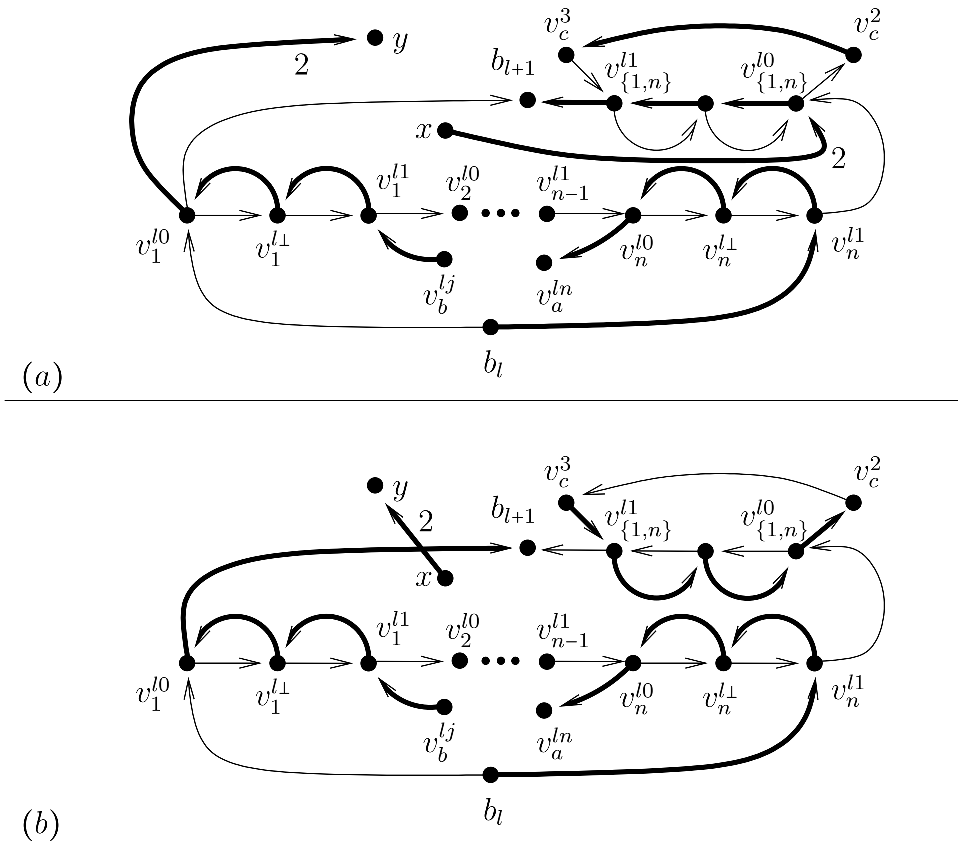


Figure 10.41: 2.Case ($\psi_\sigma(x_1) = 0$ and $\psi_\sigma(x_n) = 0$).

2.Case ($\psi_\sigma(x_1) = 0$ and $\psi_\sigma(x_n) = 0$):
Let us assume that $\psi_\sigma(x_n^l) \oplus \psi_\sigma(x_j^s) \oplus \psi_\sigma(x_k^r) = 0$ holds and σ contains the arc (v_c^2, v_c^3). Given this scenario, we may assume that (b_l, v_n^{l1}) is contained in σ due to a simple modification. Then, we are going to analyze the situation depicted in Figure 10.41 (a). We transform σ in the way described in Figure 10.41 (b). Afterwards, σ will be modified in $\mathcal{G}_c^{3A}$ such that it uses a

simple path in $\mathcal{G}_c^{3A}$ failing the parity check.

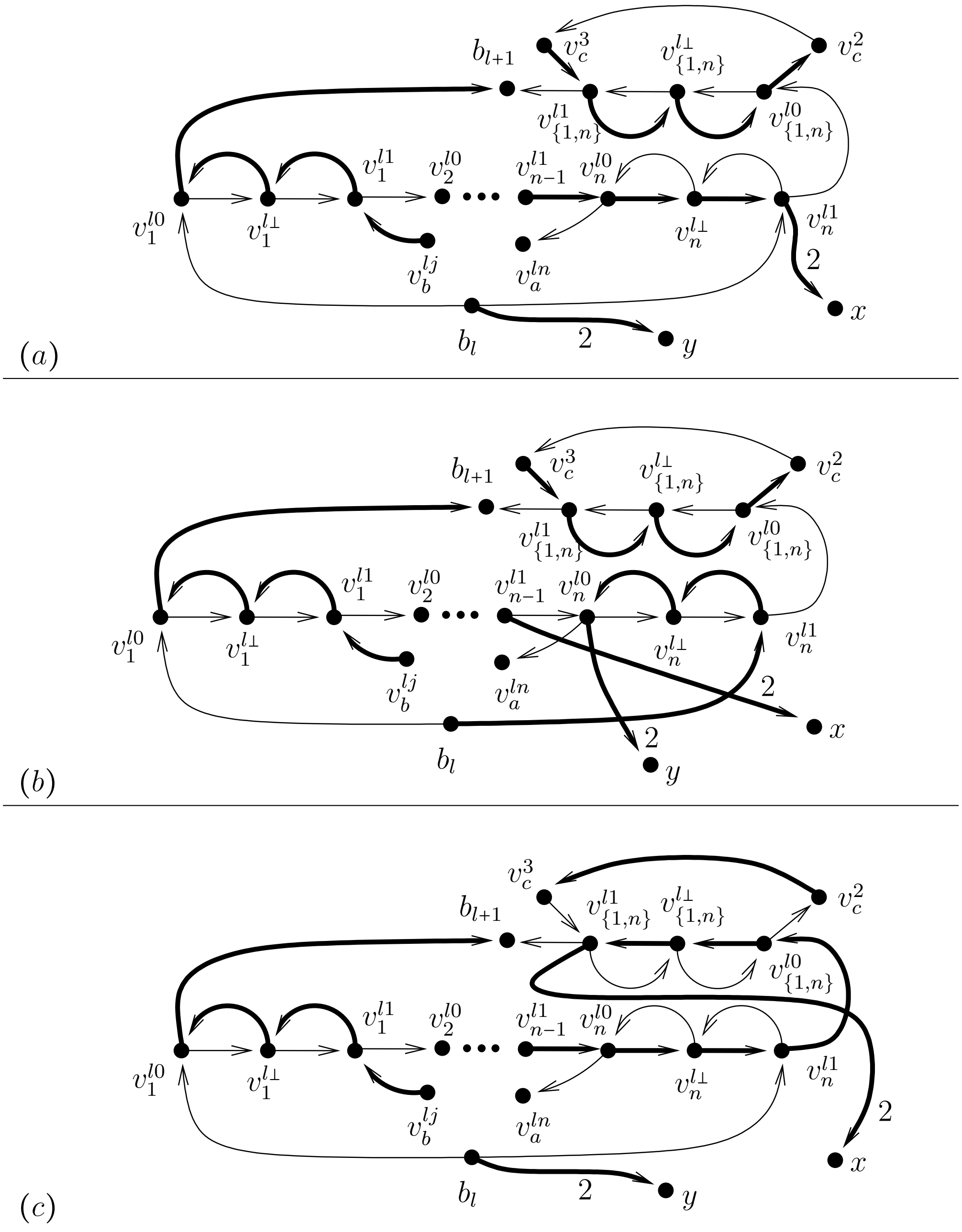


Figure 10.42: 3.Case ($\psi_\sigma(x_1) = 0$ and $\psi_\sigma(x_n) = 1$).

The case, in which $\psi_\sigma(x_i^l) \oplus \psi_\sigma(x_i^l) \oplus \psi_\sigma(x_i^l) = 1$ holds and σ contains the arc (v_c^2, v_c^3), can be discussed similarly since σ passes the parity check by including the path $v_c^3 \to v^{l1}_{\{1,n\}} \to v^{l\perp}_{\{1,n\}} \to v^{l0}_{\{1,n\}} \to v_c^2$.

In both cases, we associate the length 2 with this part of σ.

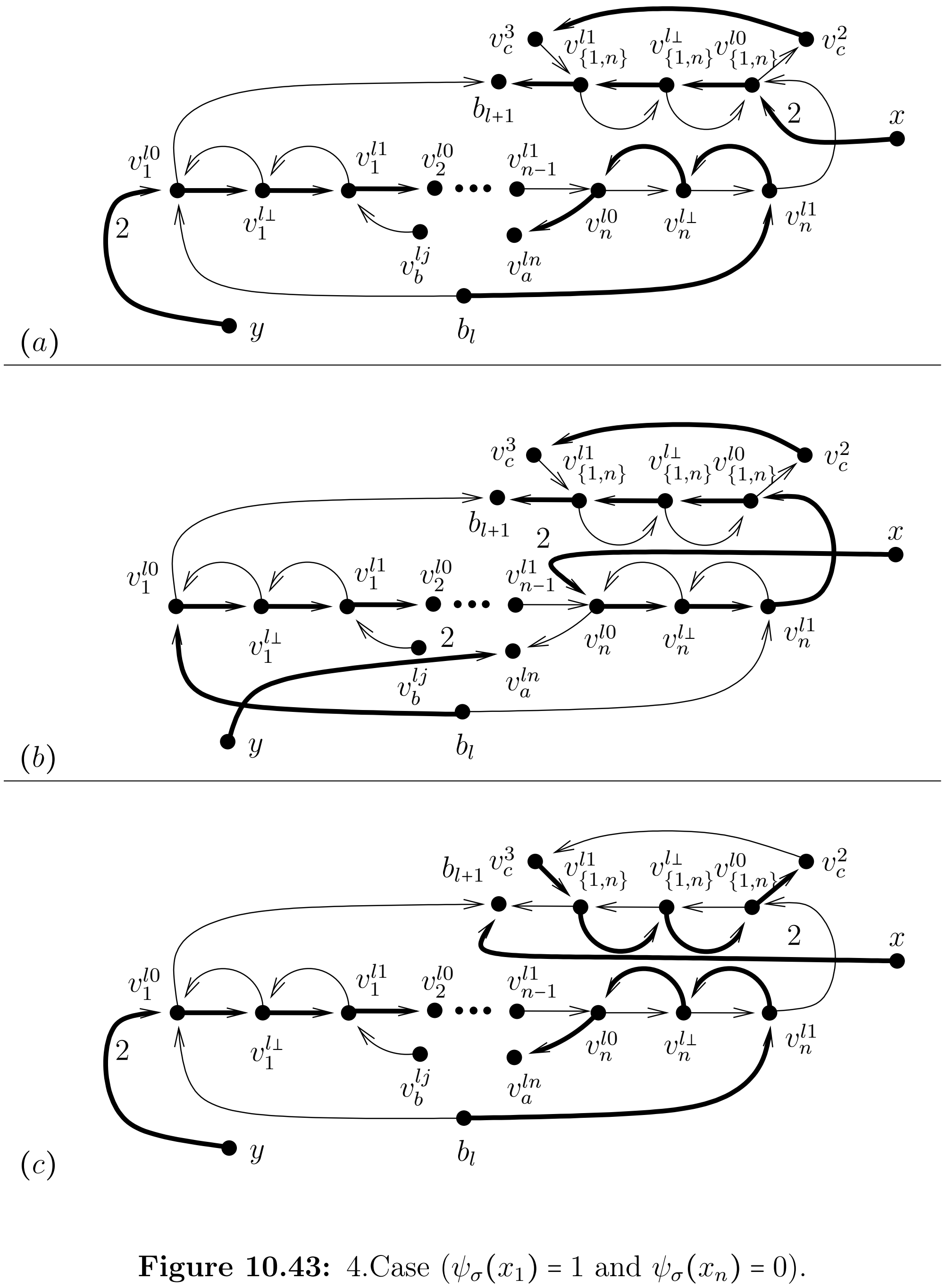


Figure 10.43: 4.Case ($\psi_\sigma(x_1) = 1$ and $\psi_\sigma(x_n) = 0$).

3.Case ($\psi_\sigma(x_1) = 0$ and $\psi_\sigma(x_n) = 1$):
Let us assume that $\psi_\sigma(x_n^l) \oplus \psi_\sigma(x_j^s) \oplus \psi_\sigma(x_k^r) = 0$ holds and σ traverses the path $v_c^3 \to v_{\{1,n\}}^{l1} \to v_{\{1,n\}}^{l\perp} \to v_{\{1,n\}}^{l0} \to v_c^2$. Then, we transform the tour σ such that it contains the arc (v_1^{l0}, b_{l+1}). Note that neither (b_l, v_1^{l0}) nor (b_l, v_n^{l1}) is included in the tour. Hence, σ contains a 2-arc to connect b_l. The same holds for the vertex v_n^{l1}. This situation is displayed in Figure 10.42 (a). We are going to invert the value of $\psi_\sigma(x_n^l)$ such that ψ_σ satisfies ℓ_c^3 and $x_1^l \oplus x_n^l = 0$. In this way, we gain at least $2-1$ more satisfied equations. The corresponding transformation is displayed in Figure 10.42 (b).

On the other hand, if we assume that $\psi_\sigma(x_n^l) \oplus \psi_\sigma(x_j^s) \oplus \psi_\sigma(x_k^r) = 1$ holds and σ traverses the path $v_c^3 \to v_{\{1,n\}}^{l1} \to v_{\{1,n\}}^{l\perp} \to v_{\{1,n\}}^{l0} \to v_c^2$, we modify the tour as depicted in Figure 10.42 (c). Note that σ passes the parity check in $\mathcal{G}_c^{3A}$ and therefore, the tour may use a simple path in $\mathcal{G}_c^{3A}$. We associate the local length 3 with σ in this case.

4.Case ($\psi_\sigma(x_1) = 1$ and $\psi_\sigma(x_n) = 0$):
Let us assume that $\psi_\sigma(x_n^l) \oplus \psi_\sigma(x_j^s) \oplus \psi_\sigma(x_k^r) = 0$ holds and σ uses the arc (v_c^2, v_c^3). Then, we transform the tour σ such that it contains the arc (b_l, v_n^{l1}). We note that neither (v_1^{l0}, b_{l+1}) nor $(v_n^{l1}, v_{\{1,n\}}^{l0})$ is included in the tour. For this reason, σ is forced to use a 2-arc to connect $v_{\{1,n\}}^{l0}$. The same holds for the vertex v_1^{l0}. The corresponding situation is displayed in Figure 10.43 (a). We modify the tour as displayed in Figure 10.43 (b) and obtain at least $2-1$ more satisfied equations.

On the other hand, if we assume that $\psi_\sigma(x_n^l) \oplus \psi_\sigma(x_j^s) \oplus \psi_\sigma(x_k^r) = 1$ holds, we need to include the path $v_c^3 \to v_{\{1,n\}}^{l1} \to v_{\{1,n\}}^{l\perp} \to v_{\{1,n\}}^{l0} \to v_c^2$ as depicted in Figure 10.43 (c). In both cases, we associate the local length 3 with this part of the tour.

In summary, our analysis yields the following statement.

Lemma 10.7.3
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem with n wheels, m_2 equations with two variables and m_3 equations with three variables. Given*

a consistent tour σ in $\mathcal{G}_{\mathscr{L}}$ with length $n + 1 + 3m_2 + 13m_3 + \tau$, it is possible to transform σ in polynomial time into a tour π such that the corresponding assignment ψ_π leaves at most τ equations in $\mathscr{L}$ unsatisfied.

Thus far, we are ready to give the proof of Theorem 10.6.1 (i).

10.7.5 Proof of Theorem 10.6.1 (i)

Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 problem consisting of n wheels $\mathcal{W}_1, \dots, \mathcal{W}_n$, m_2 equations with two variables and m_3 equations with three variables. Then, we construct in polynomial time the corresponding instance $\mathcal{G}_{\mathscr{L}}$ of the $(1,2)$-ATSP problem as described in Section 10.7.2.

• Let ϕ be an assignment to the variables in $\mathscr{L}$ leaving τ equations in $\mathscr{L}$ unsatisfied. According to Lemma 10.7.1, it is possible to construct in polynomial time a tour with length at most

$$3 \cdot m_2 + (4 + 3 \cdot 3) \cdot m_3 + n + 1 + \tau.$$

• Let σ be a tour in $\mathcal{G}_{\mathscr{L}}$ with length $3 \cdot m_2 + 13 \cdot m_3 + n + 1 + \tau$. Due to Lemma 10.7.2, we may assume that σ uses only 0/1-traversals of every parity graph included in $\mathcal{G}_{\mathscr{L}}$. According to Definition 10.7.2, we associate the corresponding assignment ψ_σ with the underlying tour σ. Recall from Lemma 10.7.3 that it is possible to convert σ in polynomial time into a tour π without increasing the length such that ψ_π leaves at most τ equations in $\mathscr{L}$ unsatisfied. ■

10.8 The $(1,4)$-ATSP Problem

In order to prove the claimed inapproximability results for the $(1,4)$-ATSP problem, we use the same construction described in Section 10.7.2 with the difference that all arcs in parity graphs have weight 1, whereas all other arcs contained in the directed graph $\mathcal{G}_{\mathscr{L}}$ obtain weight 2. The asymmetric metric space $(V_{\mathscr{L}}, d_{\mathscr{L}})$ is given by $V_{\mathscr{L}} = V(\mathcal{G}_{\mathscr{L}})$ together with the distance function $d_{\mathscr{L}}: V_{\mathscr{L}} \times V_{\mathscr{L}} \to \mathbb{Q}_{\geq 0}$, which is defined by the shortest path metric in $\mathcal{G}_{\mathscr{L}}$ bounded by the value 4. In other words, given $x, y \in V_{\mathscr{L}}$, the distance between x and y in $V_{\mathscr{L}}$ is

$$d_{\mathscr{L}}(x,y) = \min\{\text{ length of a shortest path from } x \text{ to } y \text{ in } \mathcal{G}_{\mathscr{L}},\ 4\}.$$

By construction of $(V_{\mathscr{L}}, d_{\mathscr{L}})$, we may apply similar arguments as in the proof of Theorem 10.6.1 (i). However, the remaining difficulty is to prove that given a tour σ in $(V_{\mathscr{L}}, d_{\mathscr{L}})$, we are able to transform σ in polynomial time into a tour π in $(V_{\mathscr{L}}, d_{\mathscr{L}})$, which uses only 0/1-traversals of the parity graphs, that are contained in $\mathcal{G}_{\mathscr{L}}$, without increasing the length of the tour. This statement can be proved by considering all possibilities exhaustively for each parity graph in $\mathcal{G}_{\mathscr{L}}$. We displayed some of the less obvious cases in Figure 10.46 – 10.45.

We are ready to give the proof of Theorem 10.6.1 (ii).

10.8.1 Proof of Theorem 10.6.1 (ii)

Let $\mathscr{L}$ be an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem consisting of n wheels, m_2 equations with two variables and m_3 equations with three variables. We construct in polynomial time the associated instance $(V_{\mathscr{L}}, d_{\mathscr{L}})$ of the $(1,4)$-ATSP problem.

- Let ϕ be an assignment to the variables of $\mathscr{L}$ that leaves τ equations unsatisfied in $\mathscr{L}$. Then, it is possible to construct efficiently a tour with length at most

$$m_2 \cdot (2+2) + m_3 \cdot (3 \cdot 4 + 2 \cdot 4) + 2 \cdot \tau + 2(n+1).$$

- On the other hand, suppose we are given a tour σ in $(V_{\mathscr{L}}, d_{\mathscr{L}})$ with length $4m_2 + 20m_3 + 2n + 2 + 2 \cdot \tau$. Then, it is possible to transform σ in polynomial time into a tour π without increasing the length such that the associated assignment ψ_π leaves at most τ equations in $\mathscr{L}$ unsatisfied. ■

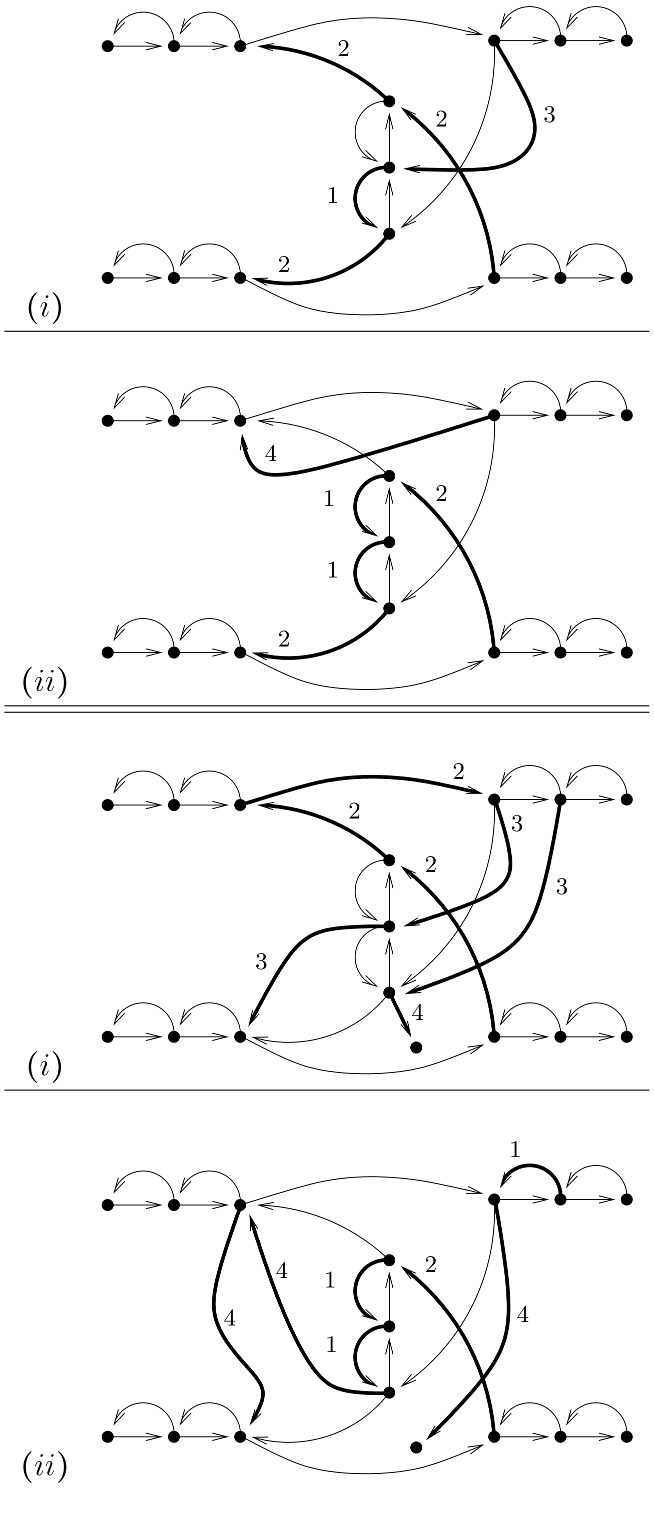


Figure 10.44: Situations before (i) and after (ii) the transformation.

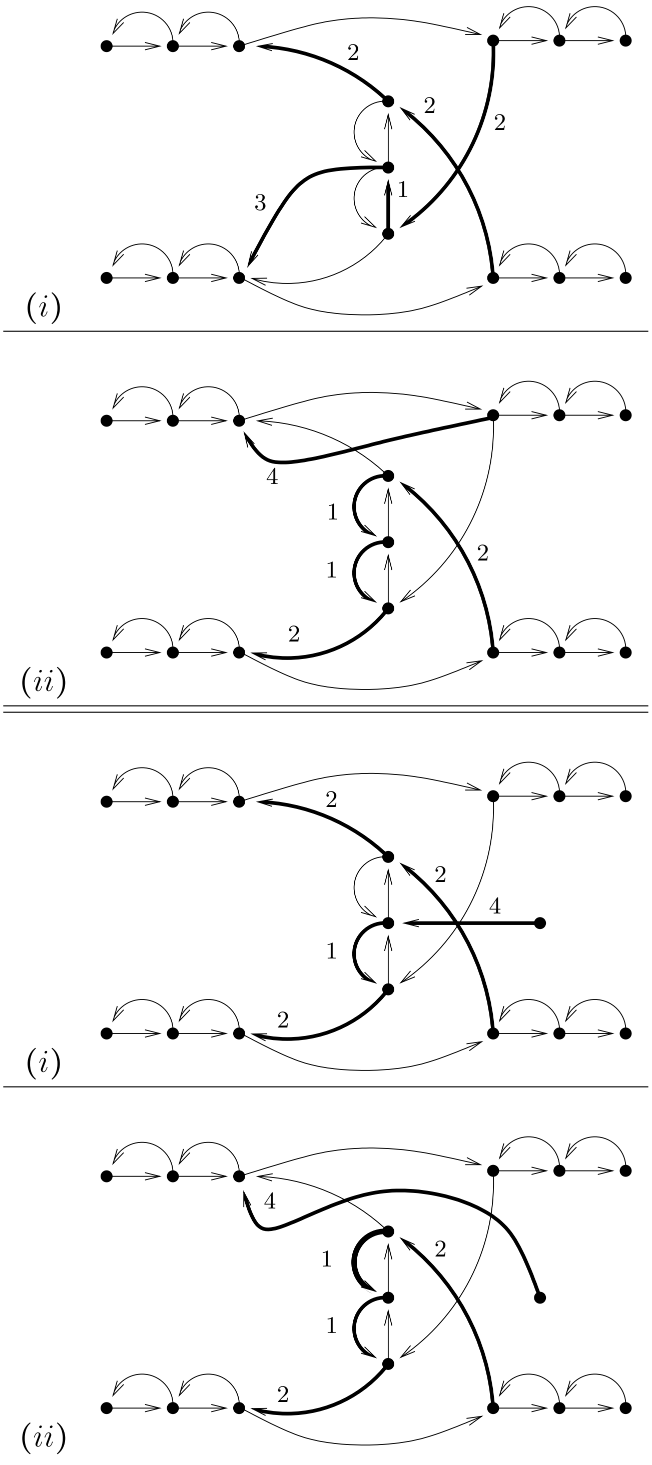


Figure 10.45: Situations before (i) and after (ii) the transformation.

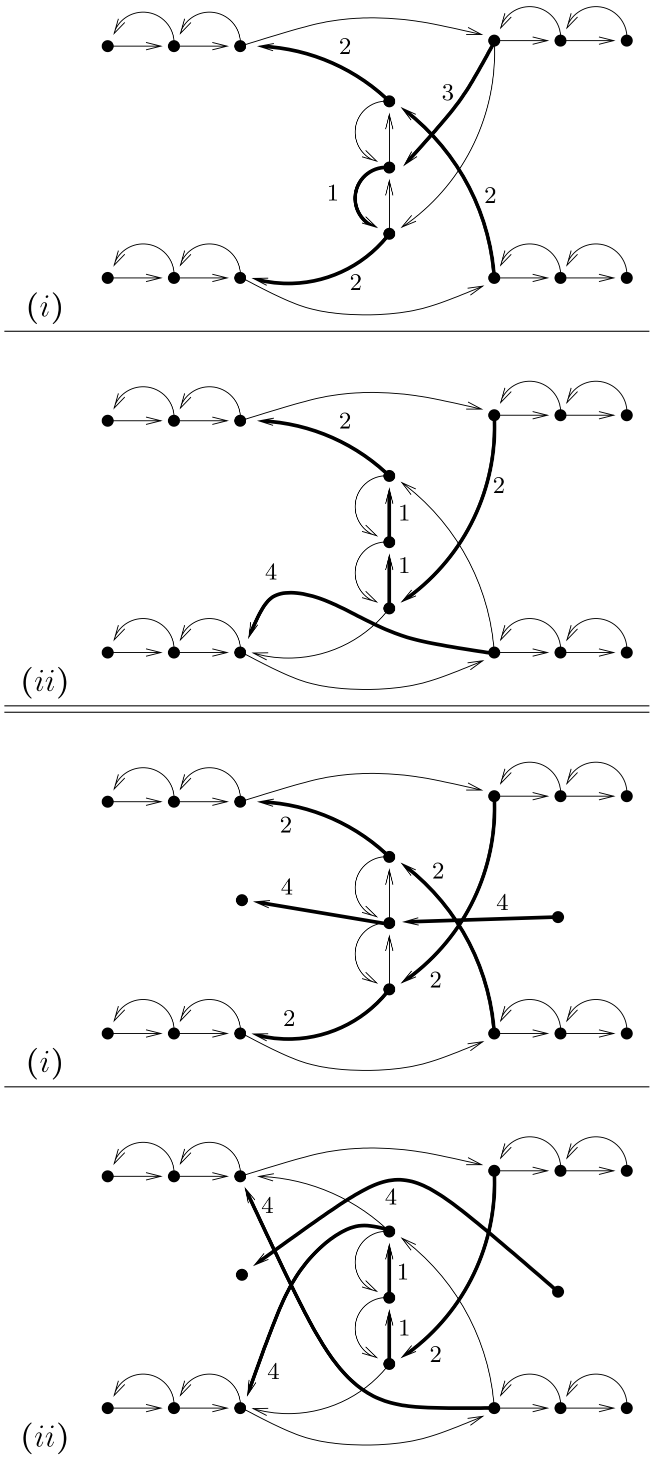


Figure 10.46: Situations before (*i*) and after (*ii*) the transformation.

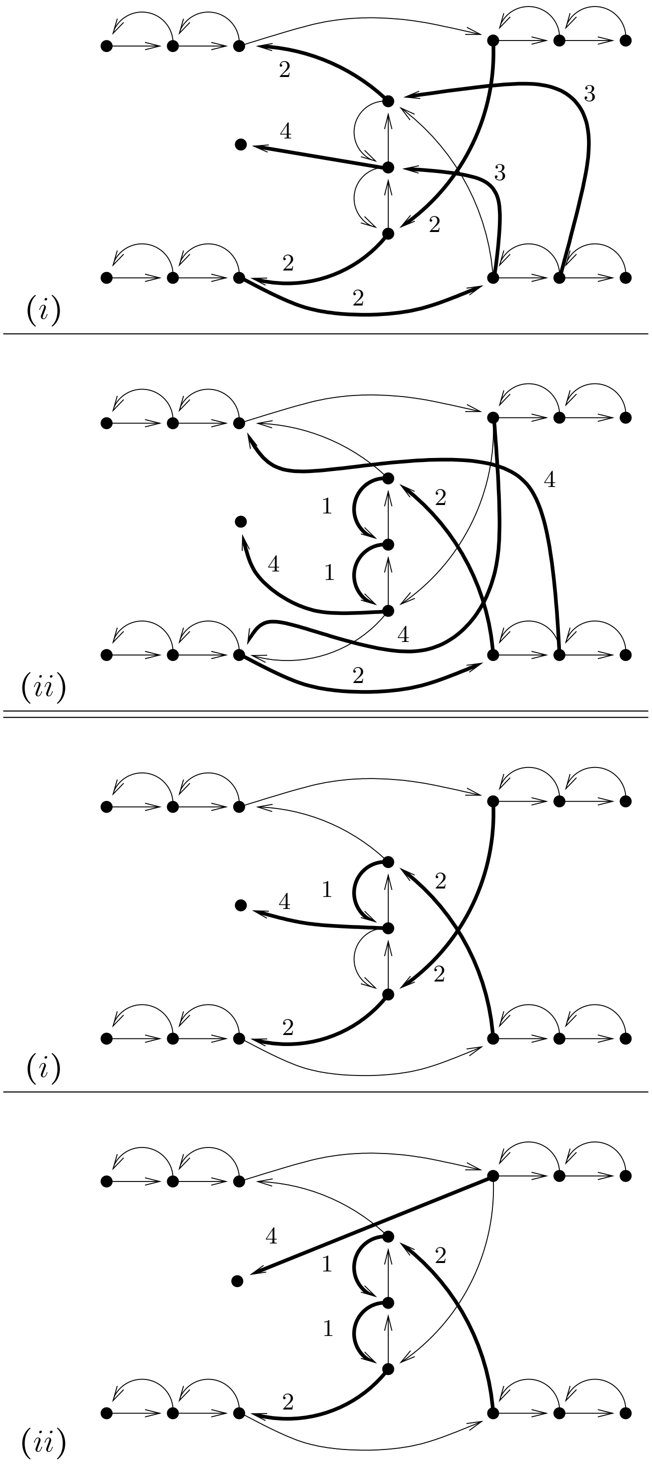


Figure 10.47: Situations before (i) and after (ii) the transformation.

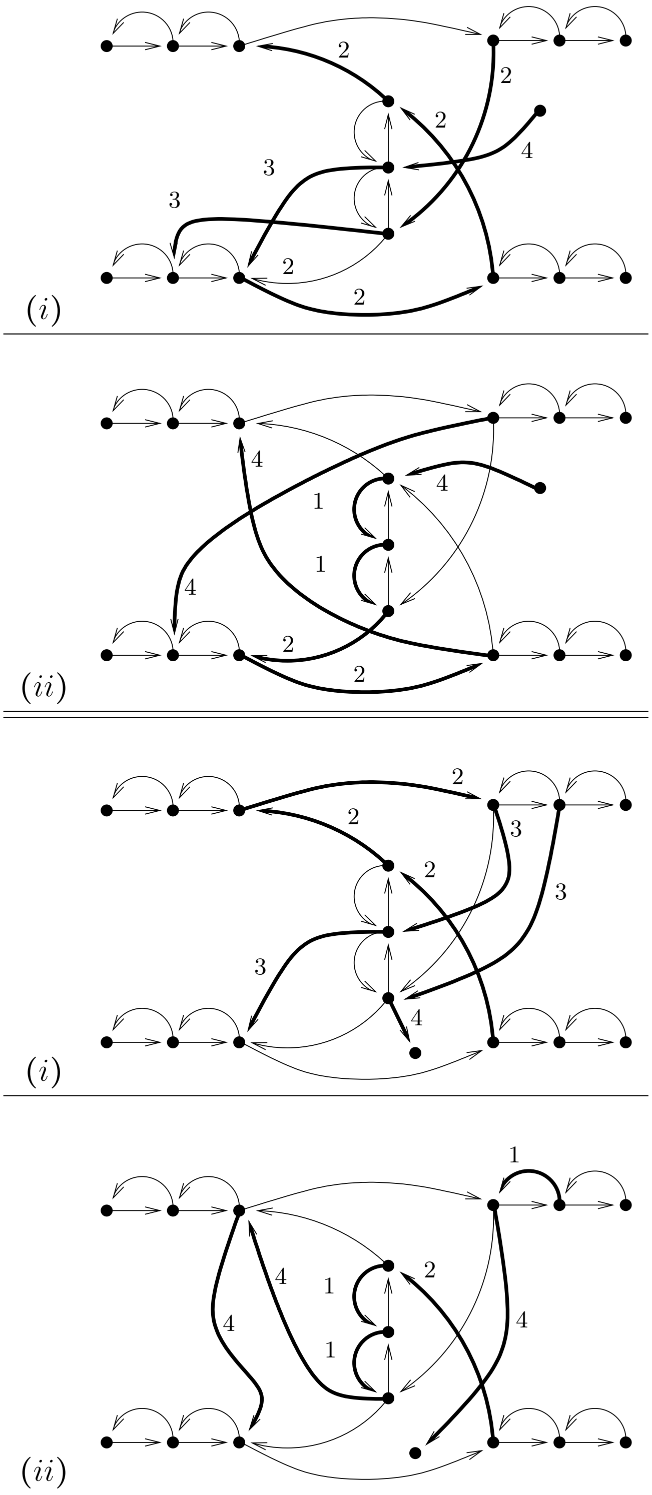


Figure 10.48: Situations before (i) and after (ii) the transformation.

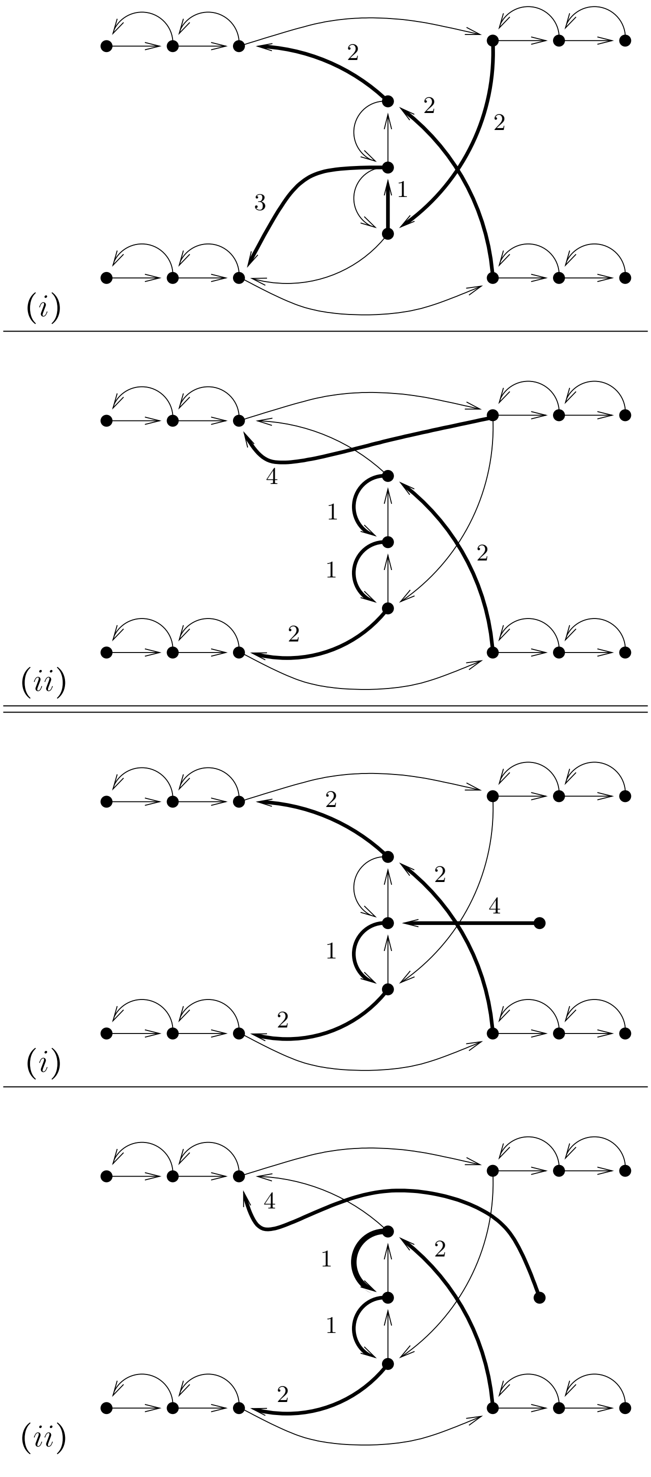
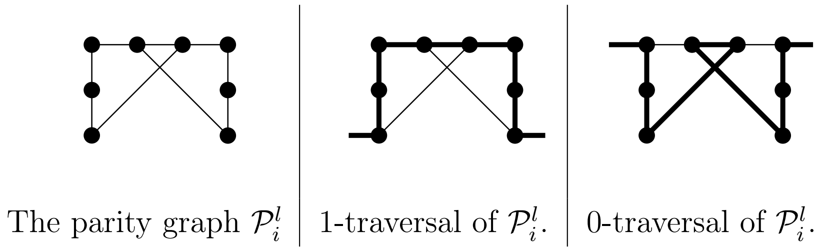


Figure 10.49: Situations before (i) and after (ii) the transformation.

10.9 The $(1,2)$-TSP Problem

In order to prove Theorem 10.6.2 (i), we adapt the construction given in Section 10.7 for the (1, 2)-ATSP problem. As for the parity gadget, we use the graph depicted in Figure 10.50 with its corresponding traversals.



The parity graph $\mathcal{P}_i^l$ | 1-traversal of $\mathcal{P}_i^l$. | 0-traversal of $\mathcal{P}_i^l$.

Figure 10.50: Traversal of the graph $\mathcal{P}_i^l$ given the assignment ϕ. The traversed edges are indicated by thick lines.

Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 problem with n wheels $\mathcal{W}_1, \ldots, \mathcal{W}_n$. Given a matching equation $x_i^l \oplus x_j^l = 0$ of $\mathcal{W}_l$ and the corresponding cycle equations $x_i^l \oplus x_{i+1}^l = 0$ and $x_j^l \oplus x_{j+1}^l = 0$, we connect the associated parity graphs $\mathcal{P}_i^l$, $\mathcal{P}_{i+1}^l$, $\mathcal{P}_{\{i,j\}}^l$, $\mathcal{P}_j^l$ and $\mathcal{P}_{j+1}^l$ as displayed in Figure 10.51.

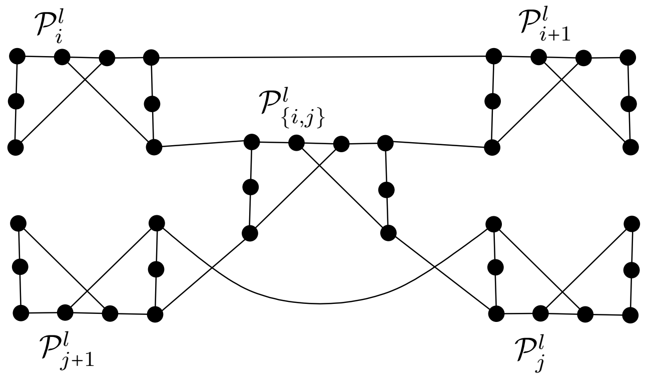


Figure 10.51: Graphs corresponding to equations $x_i^l \oplus x_j^l = 0$, $x_i^l \oplus x_{i+1}^l = 0$ & $x_j^l \oplus x_{j+1}^l = 0$.

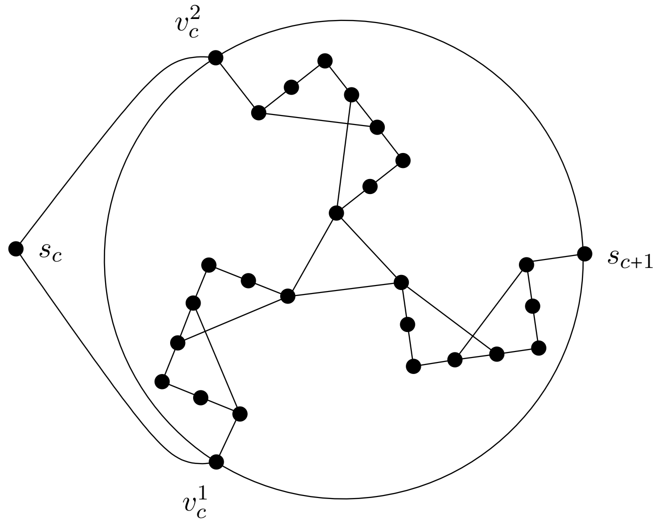


Figure 10.52: The graph $\mathcal{G}_c^{3S}$ corresponding to $\ell_c^3 \equiv x \oplus y \oplus z = 0$.

For equations with three variables of the form $x \oplus y \oplus z = 0$, we use the graph $\mathcal{G}^{3S}$ displayed in Figure 10.52. Recall from Lemma 10.5.2 that there is a simple path from s_c to s_{c+1} in Figure 10.52 containing the vertices $v \in \{v_c^1, v_c^2\}$ if and only if an even number of parity graphs is traversed. For the cycle border equation of a wheel $\mathcal{W}_l$ with associated variables $\{x_1^l, \ldots, x_p^l\}$, we introduce the path $b_l^1 - b_l^2 - b_l^3$ and the parity graph $\mathcal{P}_{\{1,p\}}^l$. In addition, we connect b_l^3 and b_{l+1}^1 to the parity graphs $\mathcal{P}_1^l$, $\mathcal{P}_p^l$ and $\mathcal{P}_{\{1,p\}}^l$ in a similar way as in the reduction from the MAX-HYBRID-LIN2 problem to the (1, 2)-ATSP problem. The graphs corresponding to equations with three variables are hooked together such that the vertex s_{c+1} of $\mathcal{G}_c^{3S}$ is identified with the vertex s_{c+1} of the graph $\mathcal{G}_{c+1}^{3S}$. This is the whole description of the corresponding graph $\mathcal{G}_{\mathscr{L}}^S$.

Due to the following lemma, we may assume that the underlying tour is using only 0/1-traversals of the parity gadgets contained in $\mathcal{G}_{\mathscr{L}}^S$.

Lemma 10.9.1

Let σ be a tour in $\mathcal{G}_{\mathscr{L}}^S$. For every parity graph P, the tour σ in $\mathcal{G}_{\mathscr{L}}^S$ can be transformed in polynomial time into a tour π in $\mathcal{G}_{\mathscr{L}}^S$ such that π is using a

0/1-*traversal of* P *without increasing length of the tour.*

Proof of Lemma 10.9.1.
For every parity graph contained in $\mathcal{G}_{\mathscr{L}}$, it can be seen by considering all possibilities exhaustively that any tour in $\mathcal{G}_{\mathscr{L}}$, that is not using the corresponding 0/1-traversals, can be modified into a tour with at most the same number of 2-edges. The less obvious cases are shown in Figure 10.53 and Figure 10.54 ■

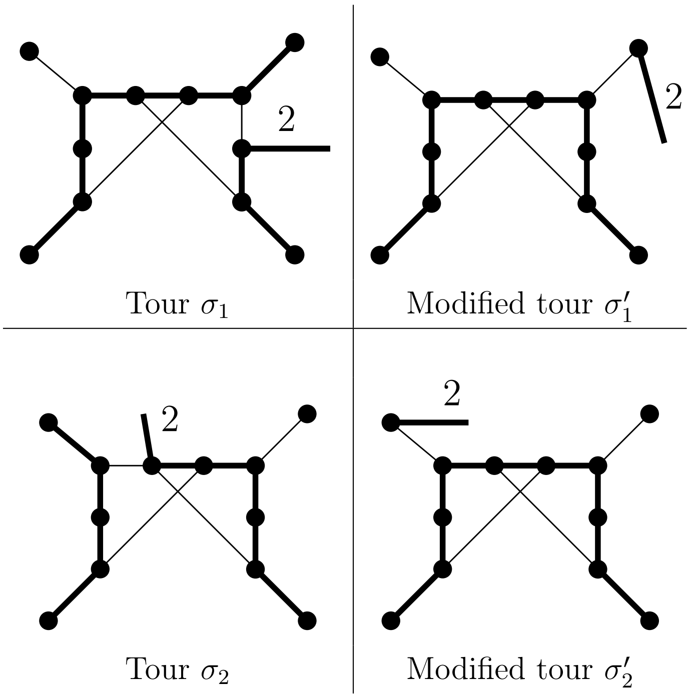


Figure 10.53: Transformations yielding a consistent tour.

10.9.1 Proof of Theorem 10.6.2 (i)

Given $\mathscr{L}$ an instance of the MAX-HYBRID-LIN2 problem consisting of n wheels $\mathcal{W}_1, \ldots, \mathcal{W}_n$, m_2 equations with two variables and m_3 equations with three variables, we construct in polynomial time the associated instance $\mathcal{G}^S_{\mathscr{L}}$ of the $(1,2)$-TSP problem.

- Given an assignment ϕ to the variables of $\mathscr{L}$ leaving τ equations unsatisfied in $\mathscr{L}$, then, there is a tour with length at most

$$8 \cdot m_2 + (3 \cdot 8 + 3) \cdot m_3 + 3(n+1) - 1 + \tau.$$

- On the other hand, if we are given a tour σ in $\mathcal{G}^S_{\mathscr{L}}$ with length $8 \cdot m_2 + (3 \cdot 8 + 3) \cdot m_3 + 3(n+1) - 1 + \tau$, it is possible to transform σ in polynomial time into a tour π which uses 0/1-traversals of all parity graphs contained in $\mathcal{G}^S_{\mathscr{L}}$ without increasing the length of the tour.

Moreover, we are able to construct in polynomial time an assignment to the variables of $\mathscr{L}$, which leaves at most τ equations in $\mathscr{L}$ unsatisfied. ■

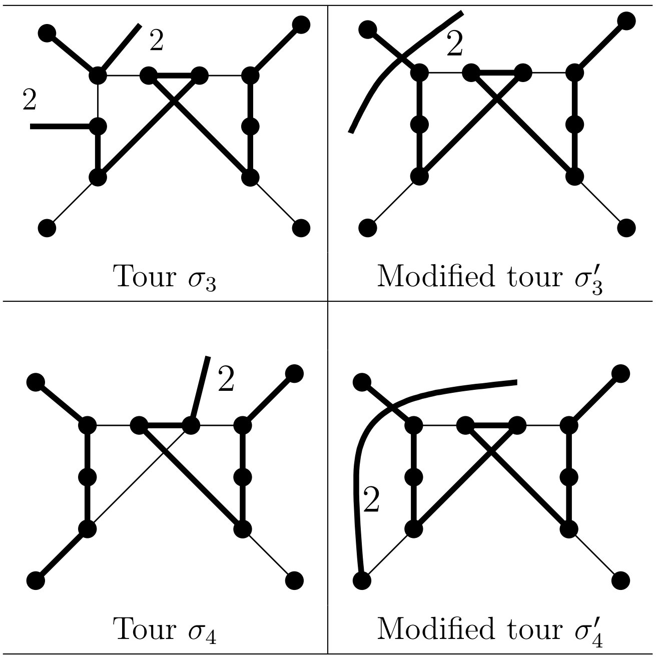
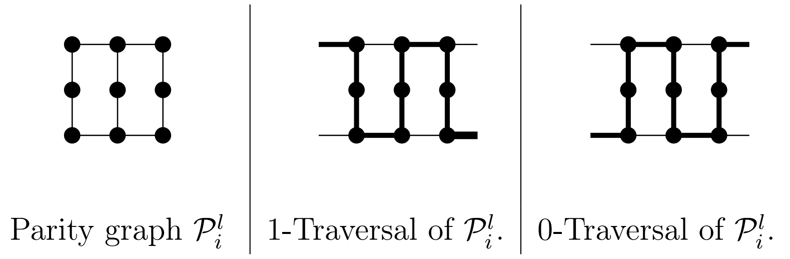


Figure 10.54: Transformations yielding a consistent tour.

10.10 The $(1,4)$-TSP Problem

In order to prove the claimed approximation hardness results for the (1,4)-TSP problem, we cannot use the same parity graphs as in the construction in the previous section since tours are not necessarily consistent in the induced metric. For this reason, we introduce the parity graph displayed in Figure 10.55 with its corresponding traversals.



Parity graph $\mathcal{P}_i^l$ | 1-Traversal of $\mathcal{P}_i^l$. | 0-Traversal of $\mathcal{P}_i^l$.

Figure 10.55: 0/1-Traversals of the graph $\mathcal{P}_i^l$. The traversed edges are indicated by thick lines.

Given a matching equation $x_i^l \oplus x_j^l = 0$ in $\mathscr{L}$ and the cycle equations $x_i^l \oplus x_{i+1}^l = 0$ and $x_j^l \oplus x_{j+1}^l = 0$, we connect the corresponding graphs as displayed in Figure 10.56.

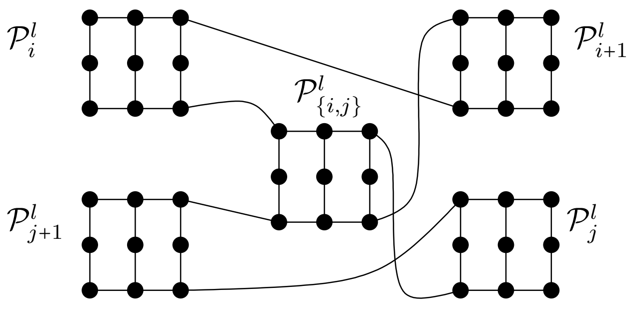
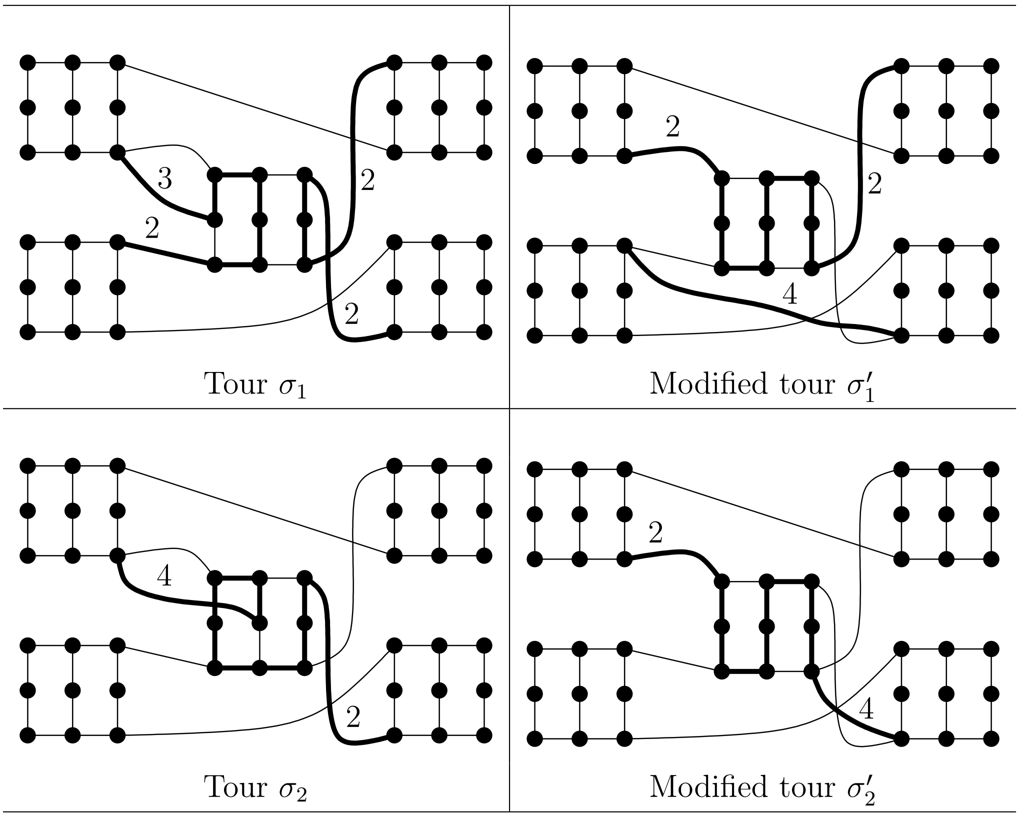


Figure 10.56: The graphs corresponding to equations $x_i^l \oplus x_j^l = 0$, $x_i^l \oplus x_{i+1}^l = 0$ and $x_j^l \oplus x_{j+1}^l = 0$.

In order to define the new instance of the $(1,4)$-TSP problem, we replace all parity graphs in $\mathcal{G}_{\mathscr{L}}^S$ by graphs displayed in Figure 10.55. In the following, we refer to this graph as $\mathcal{G}_{\mathscr{L}}^4$. All edges contained in a parity graph have weight 1, whereas all other edges have weight 2. The remaining distances in the associated metric space $(V_{\mathscr{L}}^S, d_{\mathscr{L}}^S)$ are induced by the shortest path metric in $\mathcal{G}^4_{\mathscr{L}}$ bounded by the value 4 meaning

$$d^S_{\mathscr{L}}(\{x,y\}) = \min\{\text{length of a shortest path from } x \text{ to } y \text{ in } \mathcal{G}^4_{\mathscr{L}},\ 4\}.$$

This is the whole description of the associated instance $(V_{\mathscr{L}}, d_{\mathscr{L}})$ of the $(1,4)$-TSP problem.



Tour σ_1

Modified tour σ_1'

Tour σ_2

Modified tour σ_2'

Figure 10.57: Transformations yielding a consistent tour.

We are ready to give the proof of Theorem 10.6.2 (ii).

10.10.1 Proof of Theorem 10.6.2 (ii)

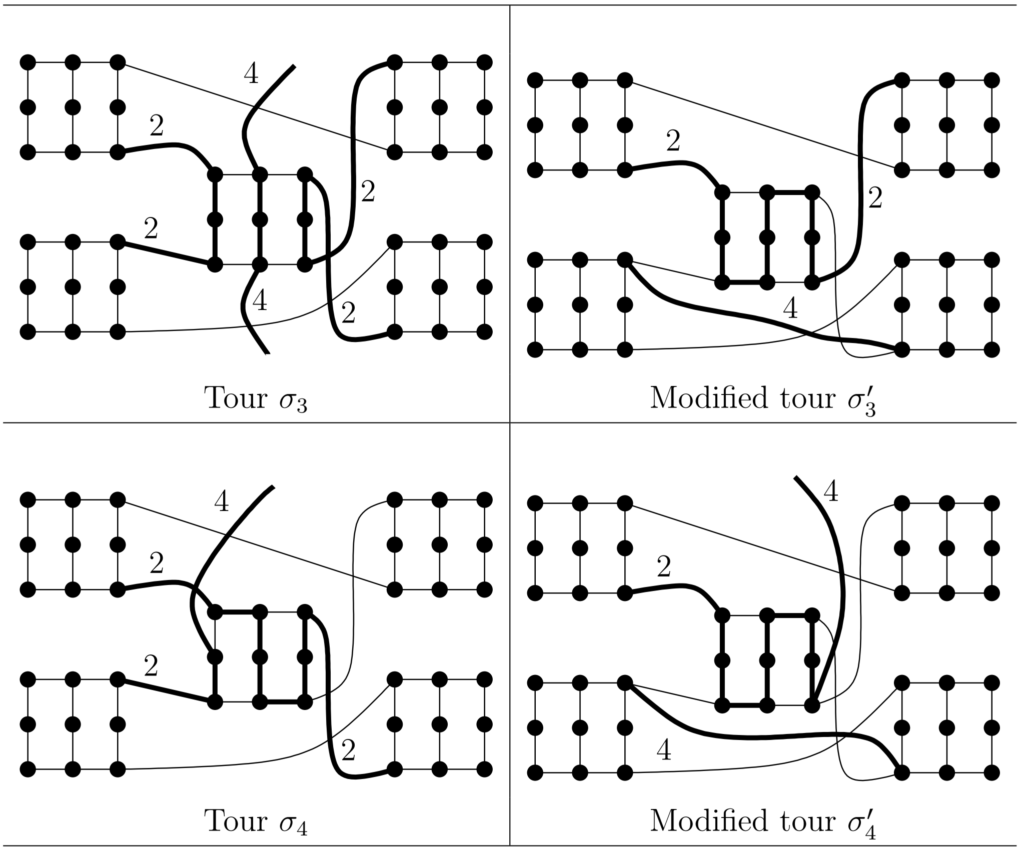
Given $\mathscr{L}$ an instance of the MAX-HYBRID-LIN2 problem consisting of n wheels $\mathcal{W}_1, \ldots, \mathcal{W}_n$, m_2 equations with two variables and m_3 equations with three variables, we construct in polynomial time the associated instance

$(V^S_{\mathscr{L}}, d^S_{\mathscr{L}})$ of the (1,4)-TSP problem.

• Given an assignment ϕ to the variables of $\mathscr{L}$ leaving u equations in $\mathscr{L}$ unsatisfied, then, there is a tour in $(V^S_{\mathscr{L}}, d^S_{\mathscr{L}})$ with length at most

$$m_2 \cdot (2+8) + m_3 \cdot (3 \cdot 10 + 2 \cdot 3) + 6(n+1) - 2 + 2 \cdot u.$$

• On the other hand, if we are given a tour σ in $(V_{\mathscr{L}}, d_{\mathscr{L}})$ with length $10m_2+36m_3+6(n+1)-2+2{\cdot}u$, it is possible to transform σ in polynomial time into a tour σ' such that it uses 0/1-traversals of all contained parity graphs in $\mathcal{G}_{\mathcal{H}}$ without increasing the length. Some cases are displayed in Figure 10.57 and 10.58. Then, we are able to construct in polynomial time an assignment to the variables of $\mathscr{L}$, which leaves at most u equations in $\mathscr{L}$ unsatisfied. ■



Tour σ_3

Modified tour σ'_3

Tour σ_4

Modified tour σ'_4

Figure 10.58: Transformations yielding a consistent tour.

10.11 The (1,2)-TSP Restricted to Subcubic Instances

This section is devoted to the proof of Theorem 10.6.3 restated below.

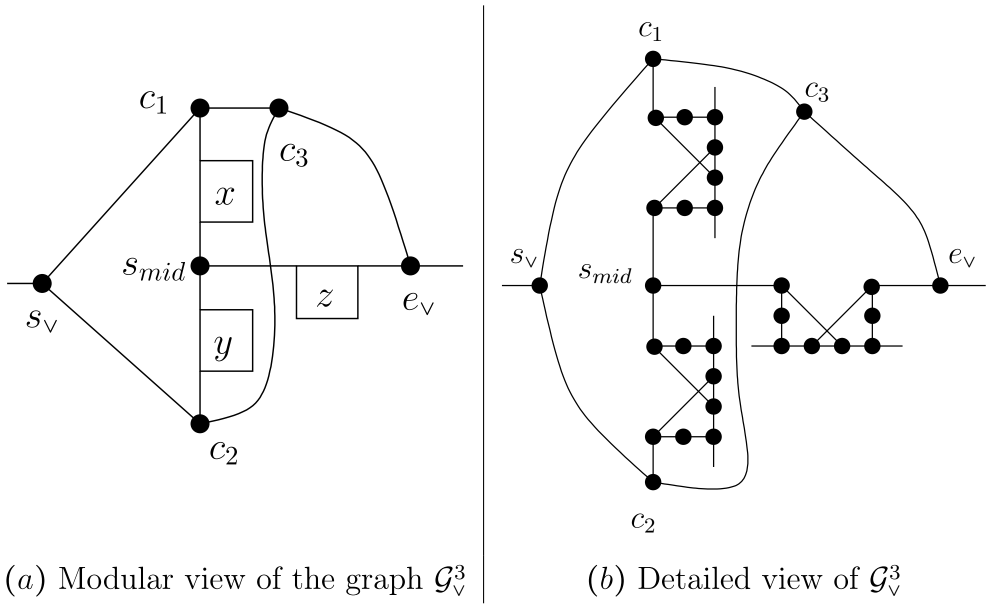
Theorem 10.6.3
The (1, 2)-TSP *problem restricted to subcubic instances is* **NP***-hard to approximate to within any factor less than* 673/672*.*

In order to prove Theorem 10.6.3, we introduce a special subcubic graph $\mathcal{G}^{12}_{SC}$ as an instance of the (1, 2)-TSP problem that is simulating the equations in the MAX-HYBRID-LIN2 problem. For this, we first construct the graph $\mathcal{G}^{S}_{\mathscr{L}}$ given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem as described in Section 10.9. Then, we define a new outer loop of $\mathcal{G}^{S}_{\mathscr{L}}$ in order to obtain an instance of the (1, 2)-TSP problem in subcubic graphs. Let us start with the description of the subcubic graph $\mathcal{G}^{12}_{SC}$ given $\mathcal{G}^{S}_{\mathscr{L}}$.

10.11.1 The Construction of $\mathcal{G}^{12}_{SC}$

The gadgets simulating equation with three variables in $\mathcal{G}^{S}_{\mathscr{L}}$ contain vertices with degree 5. We are going to replace these gadgets by cubic graphs which we will specify later on. In order to understand the cubic gadgets, we first describe a reduction from the MAX-E3LIN2 problem to the MAX-2in3SAT problem. The reduction is straightforward: Given an equation of the form $x \oplus y \oplus z = 0$, we create three clauses $(x \vee a_1 \vee a_2)$, $(y \vee a_2 \vee a_3)$ and $(z \vee a_1 \vee a_3)$. Note that if we are given an assignment to x, y and z that satisfies the equation, then, it is possible to find an assignment to a_1, a_2 and a_3 that satisfies all three corresponding clauses. In the other case, we find assignments to a_1, a_2 and a_3 that make at most two clauses satisfied.

In the next step, we are going to design a gadget that simulates the predicate 2in3SAT. This gadget is displayed in Figure 10.50 (a). The boxes can be viewed as modules, which will be replaced with a parity gadget or a graph with similar properties (see Figure 10.63). Any graph with less vertices and the properties of a parity gadget will lead to improved inapproximability factors for the corresponding problems.



(a) Modular view of the graph $\mathcal{G}^3_\vee$

(b) Detailed view of $\mathcal{G}^3_\vee$

Figure 10.59: The graph $\mathcal{G}^3_\vee$ simulating the 2in3SAT predicate $(x \vee y \vee z)$.

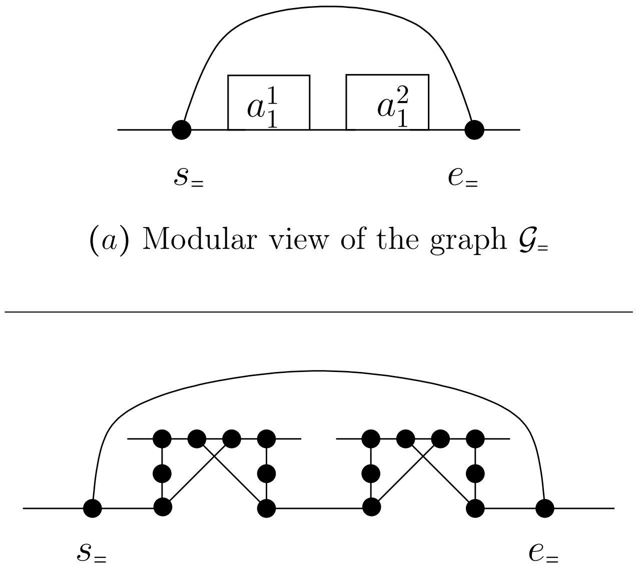
Note that the graph in Figure 10.59 (b) has degree at most 3. For this graph, we are going to prove the following statement.

Lemma 10.11.1

There is a Hamiltonian path from $s_\vee$ to $e_\vee$ in the graph displayed in Figure 10.59 (a) if and only if 2 edges with modules are traversed.

Proof. There are three possibilities to enter the vertex s_{mid}. Therefore, a Hamiltonian path in $\mathcal{G}^3_\vee$ contains (i) $c_1 - s_{mid} - c_2$, (ii) $c_1 - s_{mid} - e_\vee$ or (iii) $c_2 - s_{mid} - e_\vee$. In the case (i), we are forced to use $\{c_3, e_\vee\}$ and then, either $\{s_\vee, c_1\}$ and $\{c_3, c_2\}$ or $\{s_\vee, c_2\}$ and $\{c_3, c_1\}$. In the case (ii), we first note that we cannot use $\{e_\vee, c_3\}$. Due to the degree condition, we have to use $c_2 - c_3 - c_1$. The only remaining vertex with degree one is c_2 and has to be connected to $s_\vee$. In case (iii), we may argue similarly to case (ii). ■

As for the next step, we introduce a gadget that simulates $a^1_1 \oplus a^2_1 = 0$ displayed in Figure 10.60. We see that in order to get from the vertex s_1 to e_1, we simply use the edge $\{s_1, e_1\}$ or the three edges which are connecting the two parity gadgets.



(a) Modular view of the graph $\mathcal{G}_=$

(b) Detailed view of $\mathcal{G}_=$

Figure 10.60: Graph $\mathcal{G}_=$ corresponding to $a_1^1 \oplus a_1^2 = 0$.

We are ready to describe the construction that simulates the equation $x \oplus y \oplus z = 0$: We create three copies of the gadget $\mathcal{G}_\vee^3$, denoted by $\mathcal{G}_\vee^{31}$, $\mathcal{G}_\vee^{32}$ and $\mathcal{G}_\vee^{33}$, to simulate $(x \vee a_1^1 \vee a_2^1)$, $(y \vee a_2^2 \vee a_3^1)$ and $(z \vee a_1^2 \vee a_3^2)$. For each $i \in [3]$, the vertex set of $\mathcal{G}_\vee^{3i}$ is defined by $\{s_\vee^i, c_1^i, c_1^i, c_2^i, c_3^i, e_\vee^i, s_{mid}^i\}$. In order to connect those three copies, we add the edge $\{e_\vee^i, s_\vee^{i+1}\}$ for each $i \in [2]$. In the next step, we create three copies of the gadget $\mathcal{G}_=$, denoted by $\mathcal{G}_=^1$, $\mathcal{G}_=^2$ and $\mathcal{G}_=^3$, to simulate $a_1^1 \oplus a_1^2 = 0$, $a_2^1 \oplus a_2^2 = 0$ and $a_3^1 \oplus a_3^2 = 0$. For each $i \in [3]$, the vertex set of $\mathcal{G}_=^i$ is defined by $\{s_=^i, e_=^i\}$. Again, we connect those three copies by adding $\{e_=^i, s_=^{i+1}\}$ for each $i \in [2]$ and we also create $\{e_\vee^3, s_=^1\}$ in order to connect $\mathcal{G}_\vee^3$ with $\mathcal{G}_=^1$. The whole construction is illustrated in Figure 10.61.

Finally, we connect the graphs that we introduced by parity gadgets as follows: For each graph $\mathcal{G}_=^i$, we create two parity gadgets and connect them to the graph $\mathcal{G}_\vee^{3j}$ corresponding the clause, in which the variable a_i^k with $k \in \{1, 2\}$ appear (See Figure 10.62 for a detailed view). The parity gadgets,

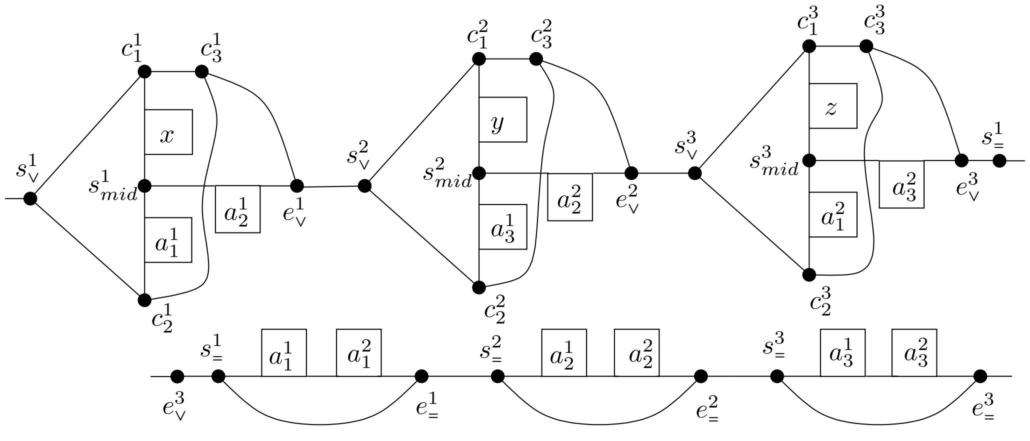


Figure 10.61: Modular view of the construction simulating $x \oplus y \oplus z = 0$.

which are associated to the variables x, y and z, are attached to $\mathcal{G}_\vee^{3j}$ with $j \in \{1, 2, 3\}$ similarly as in the construction described in Section 10.9 for the graph $\mathcal{G}_c^{3S}$. Hence, the parity gadget is also connected to the graph which is associated to the wheel $\mathcal{W}_\alpha$, where $\alpha \in \{x, y, z\}$.

Given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem, we refer to the corresponding instance of the (1, 2)-TSP problem on subcubic graphs as $\mathcal{G}_{SC}^{12}$.

10.11.2 Tours in $\mathcal{G}_{SC}^{12}$ From Assignments

We are going now to construct a tour from a given assignment and prove the following lemma.

Lemma 10.11.2
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem with n wheels, 60ν equation with two variables, 2ν equations with three variables and ϕ an assignment that leaves at most $\delta\nu$ equations unsatisfied. Then, there is a tour in G_{SC}^{12} with cost at most $672\nu + 3(n + 1) - 1 + \delta\nu$*

Proof. Let ϕ be an assignment to the variables of $\mathscr{L}$. Due to the properties of a wheel amplifier graph, we may assume that the variables associated to wheel in $\mathscr{L}$ take the same value under ϕ.

Given the assignment ϕ, we define the inner loop of the tour in $\mathcal{G}_{SC}^{12}$ in the same way as in the proof of Theorem 10.6.2 (i). This means that some of the parity gadgets which are connected to gadgets simulating equations with three variables may have been traversed in the inner loop of the tour.

In the outer loop of the tour, if the assignment satisfies the underlying equation $x \oplus y \oplus z = 0$, then there is a Hamiltonian path traversing all graphs corresponding to $(x \vee a_1^1 \vee a_2^1)$, $(y \vee a_2^2 \vee a_3^1)$, $(z \vee a_1^2 \vee a_3^2)$, $a_1^1 \oplus a_1^2 = 0$, $a_2^1 \oplus a_2^2 = 0$ and $a_3^1 \oplus a_3^2 = 0$. For each satisfied equation with three variables, we associate the cost $3 \cdot (6+3\cdot 8+2)$. If the underlying equation is not satisfied, we have to introduce a 2-edge. Thus, we associate the cost $3\cdot(6+3\cdot 8+2)+1$. Summing up, we obtain a tour in $\mathcal{G}_{SC}^{12}$ with cost at most

$$8 \cdot 60\nu + 3 \cdot (6+3\cdot 8+2) \cdot 2\nu + 3(n+1) - 1 + \delta\nu = 672\nu + 3(n+1) - 1 + \delta\nu$$

and the proof of Lemma 10.11.2 follows. ■

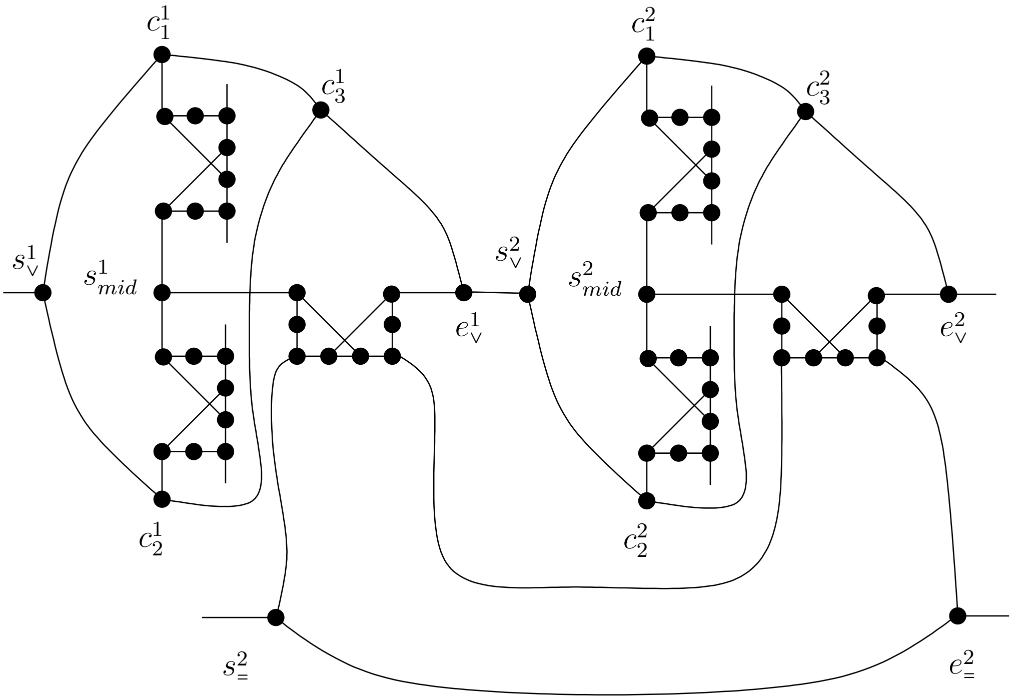


Figure 10.62: Detailed view of the gadget for $(x \vee a_1^1 \vee a_2^1)$, $(y \vee a_2^2 \vee a_3^1)$ and $a_2^1 \oplus a_2^2 = 0$.

10.11.3 Assignments From Tours in $\mathcal{G}_{SC}^{12}$

Given a tour in $\mathcal{G}_{SC}^{12}$, we are going to construct an assignment to the variables of the corresponding instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem and prove the following lemma.

Lemma 10.11.3
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem with n wheels, 60ν equations with two variables, 2ν equations with three variables and π a tour in $\mathcal{G}_{SC}^{12}$ with cost $672\nu + 3(n+1) - 1 + \delta\nu$. Then, it is possible to construct efficiently an assignment that leaves at most $\delta\nu$ equations in $\mathscr{L}$ unsatisfied.*

Proof. In the first step, we convert the underlying tour in $\mathcal{G}_{SC}^{12}$ into a consistent one without increasing its cost. This is done by applying Lemma 10.9.1 to each parity gadget in $\mathcal{G}_{SC}^{12}$. In the second step, we use the same 0/1-traversals of the parity gadgets in the inner loop of the tour which enables us to construct a tour in the corresponding instance $\mathcal{G}_{\mathscr{L}}^{S}$ with cost at most

$$672\nu+3(n+1)-1+\delta\nu-3\cdot(6+3\cdot8+2)\cdot2\nu+(3\cdot8+3)\cdot2\nu = 534\nu+3(n+1)-1+\delta\nu.$$

Finally, we apply Theorem 10.6.2 (i) and compute efficiently an assignment that leaves at most $\delta\nu$ equations in $\mathscr{L}$ unsatisfied. ∎

We are ready to give the proof of Theorem 10.6.3.

Proof of Theorem 10.6.3.
Given $\mathscr{L}$ an instance of the MAX-HYBRID-LIN2 problem consisting of n wheels, 60ν equations with two variables and 2ν equations with three variables, we construct in polynomial time the associated instance $\mathcal{G}_{SC}^{12}$ of the $(1,2)$-TSP problem.

Given an assignment ϕ to the variables of $\mathscr{L}$ leaving $\delta\cdot\nu$ equations unsatisfied with $\delta \in (0,1)$, then, according to Lemma 10.11.2, it is possible to find a tour with cost at most $672\nu + 3(n+1) - 1 + \delta\cdot\nu$.

On the other hand, if we are given a tour σ in $\mathcal{G}_{SC}^{12}$ with cost $672\nu + 3(n+1) - 1 + \delta\cdot\nu$, due to Lemma 10.11.3, we are able to construct efficiently an assignment to the variables of $\mathscr{L}$, which leaves at most $\delta\nu$ equations in $\mathscr{L}$ unsatisfied.

Similarly to the proof of Corollary 10.6.1, for a constant $\tau > 0$, we may assume that $(3n+4)/\nu \le \tau$ holds. According to Theorem 4.9.1, we know that for all $\varepsilon > 0$, it is **NP**-hard to decide whether there is a tour with cost at most $672\nu + 3(n+1) - 1 + \varepsilon \cdot \nu \le 672 \cdot \nu + \varepsilon'\nu$ or all tours have cost at least $672 \cdot \nu + (1-\varepsilon)\nu + 3(n+1) - 1 \ge 673 \cdot \nu - \varepsilon' \cdot \nu$, for some ε' that depends only on ε and τ. By appropriate choices for ϵ and τ, the ratio between these two cases can get arbitrarily close to 673/672. ■

10.12 (1,2)-TSP Restricted to Cubic Instances

This section is devoted to the proof of Theorem 10.6.4 restated below.

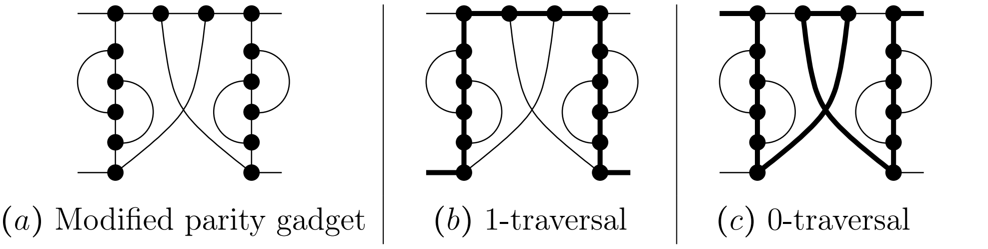
Theorem 10.6.4
The (1, 2)-TSP *problem restricted to subcubic instances is* **NP***-hard to approximate to within any factor less than* 1141/1140.

In order to prove Theorem 10.6.4, we are going to define the cubic graph $\mathcal{G}_{CU}^{12}$ as an instance of the (1, 2)-ATSP problem.

10.12.1 The Construction of the Graph $\mathcal{G}_{CU}^{12}$

Given an instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem with n wheels, 60ν equations with two variables and 2ν equations with three variables, we construct the corresponding graph $\mathcal{G}_{SC}^{12}$. In order to convert the instance $\mathcal{G}_{SC}^{12}$ of the (1, 2)-TSP problem in subcubic graphs into an instance $\mathcal{G}_{CU}^{12}$ of the (1, 2)-TSP problem in cubic graphs, we replace all vertices with degree exactly two by a path in which all vertices will have degree exactly three. Let us describe this in detail: Let w be a vertex with degree two in $\mathcal{G}_{SC}^{12}$, which is connected to x and y. Replace w with the path $p_w = v_w^1 - v_w^2 - v_w^3 - v_w^4$. In addition, we add edges $\{v_w^1, v_w^3\}$, $\{v_w^2, v_w^4\}$, $\{x, v_w^1\}$ and $\{y, v_w^4\}$. By applying this modification to each vertex of degree exactly two, we create a cubic graph and refer to it as $\mathcal{G}_{CU}^{12}$.

A modified parity gadget is displayed in Figure 10.63 (a). The corresponding traversals are defined in Figure 10.63 (b) and (c).



(a) Modified parity gadget | (b) 1-traversal | (c) 0-traversal

Figure 10.63: 0/1-Traversals of a modified parity gadget. The traversed edges are pictured by thick lines.

The following lemma enables us to construct a tour in $\mathcal{G}_{CU}^{12}$ given an assignment ϕ to the variables of the corresponding instance $\mathscr{L}$ of the MAX-HYBRID-LIN2 problem with a certain cost that depends on the number on unsatisfied equations in $\mathscr{L}$ by ϕ.

Lemma 10.12.1
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem with n wheels, 60ν equation with two variables, 2ν equations with three variables and ϕ an assignment that leaves $\delta \cdot \nu$ equations unsatisfied for some $\delta \in (0,1)$. Then, it is possible to construct efficiently a tour in $\mathcal{G}_{CU}^{12}$ with cost at most $1140\nu + 6(n+1) - 1 + \delta \cdot \nu$*

Proof. Basically, we use the same tour as constructed in Lemma 10.11.2 for the graph $\mathcal{G}_{SC}^{12}$ with the difference that instead of traversing a vertex w of degree exactly two in $\mathcal{G}_{SC}^{12}$, we have to use the path v_w^1–v_w^2–v_w^3–v_w^4 consisting of 3 more vertices. Thus, if we have given a tour σ in $\mathcal{G}_{SC}^{12}$, that was constructed according to Lemma 10.11.2, we have to add $6 \cdot 60\nu$ (for each equation with two variables), $9 \cdot 6 \cdot 2\nu$ (for each equation with three variables), and $3(n+1)$ (for each wheel) to the cost of σ and obtain a tour in $\mathcal{G}_{CU}^{12}$ with cost at most

$$672\nu + 3(n+1) - 1 + \delta \cdot \nu + (6 \cdot 60\nu) + 9 \cdot 6 \cdot 2\nu + 3(n+1) = 1140\nu + 6(n+1) - 1 + \delta \cdot \nu$$

and the proof of Lemma 10.12.1 follows. ■

10.12.2 Tours in $\mathcal{G}_{CU}^{12}$ to Assignments

We are going to prove the other direction of the reduction and give the proof of the following lemma.

Lemma 10.12.2
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem with n wheels, 60ν equation with two variables, 2ν equations with three variables and π a tour in $\mathcal{G}_{CU}^{12}$ with cost $1140\nu + 6(n+1) - 1 + \delta \cdot \nu$. Then, it is possible to construct efficiently an assignment that leaves at most $\delta \cdot \nu$ equations in $\mathscr{L}$ unsatisfied.*

Proof. Let π be a tour in $\mathcal{G}_{CU}^{12}$ with cost $1140\nu + 6(n+1) - 1 + \delta \cdot \nu$. We are going to show that we can convert efficiently π into a tour π' in $\mathcal{G}_{SC}^{12}$ with cost $672\nu + 3(n+1) - 1 + \delta \cdot \nu$. For this, we consider the path $x - v_c^1 - v_c^2 - v_c^3 - v_c^4 - y$ in $\mathcal{G}_{CU}^{12}$, where $p_c = v_c^1 - v_c^2 - v_c^3 - v_c^4$ corresponds to the vertex c of degree exactly two in the instance $\mathcal{G}_{SC}^{12}$. As we want to contract the path p_c into one vertex, we will ensure that the (1,2)-tour is using either the path $v_c^1 - v_c^2 - v_c^3 - v_c^4$ or $v_c^1 - v_c^3 - v_c^2 - v_c^4$.

Let us assume that either v_c^2 or v_c^3 is an endpoint of a 2-edge, say v_c^2. Clearly, it implies that there is another endpoint in $\{v_c^1, v_c^3, v_c^4\}$ or v_c^2 is an endpoint of another 2-edge. We delete all edges of weight 1 that the tour is using and are incident on v_c^2 and v_c^3. Then, we add $\{v_c^1, v_c^2\}$, $\{v_c^2, v_c^3\}$ and $\{v_c^3, v_c^4\}$ to connect v_c^4 and v_c^1 by edges of weight 1. Note that this transformation decreased the total number of 2-edges and the cost of the (1,2)-tour.

By applying this transformation successfully to each such path p_c, we obtain a tour which is using the complete path that corresponds to a vertex of degree 2 in the instance $\mathcal{G}_{SC}^{12}$ without increasing the cost of the tour. By contracting each path p_c into the vertex c, it yields a (1,2)-tour in $\mathcal{G}_{SC}^{12}$ with cost at most $672\nu + 3 \cdot (n+1) + 1 + \delta \cdot \nu$. Finally, we apply Lemma 10.11.3 and obtain an assignment that leaves at most $\delta \cdot \nu$ equations in $\mathscr{L}$ unsatisfied. ∎

Analogously to the proof of Theorem 10.6.3, we combine Lemma 10.12.1 with Lemma 10.12.2 and obtain Theorem 10.6.4.

10.13 Graphic TSP on Subcubic and Cubic Graphs

In this section, we are going to give the proof of Theorem 10.6.5 restated below.

Theorem 10.6.5
The GRAPHIC-TSP *problem restricted to subcubic and cubic graphs is* **NP**-*hard to approximate to within any factor less than* 685/684 *and* 1153/1152*, respectively.*

For the reduction to the GRAPHIC-TSP problem on cubic and subcubic Graphs, we are going to define the graphs $\mathcal{G}^{12}_{CU}$ and $\mathcal{G}^{12}_{SC}$.

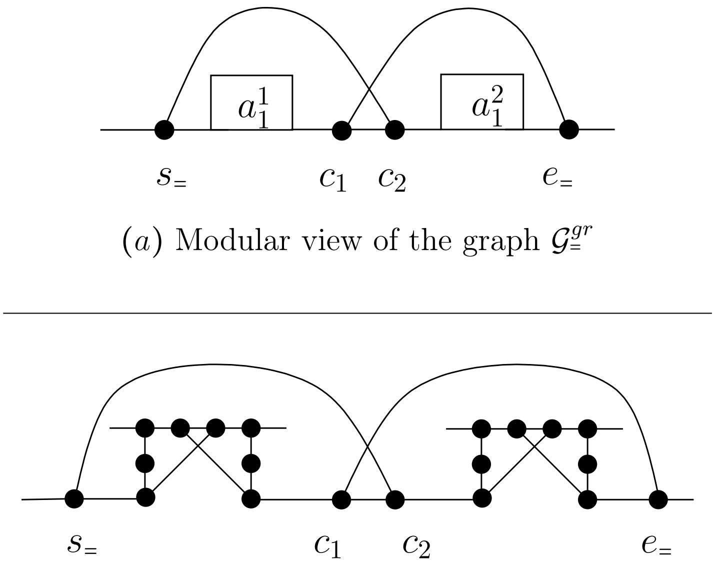
10.13.1 The Construction of $\mathcal{G}^{12}_{CU}$ and $\mathcal{G}^{12}_{SC}$

Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 problem. We first construct the corresponding instances $\mathcal{G}^{12}_{CU}$ and $\mathcal{G}^{12}_{SC}$ of the (1, 2)-TSP problem in cubic and subcubic graphs, respectively. Each gadget $\mathcal{G}_=$ in $\mathcal{G}^{12}_{SC}$ is replaced by the graph $\mathcal{G}^{gr}_=$ displayed in Figure 10.64. We refer to this construction as the graph $\mathcal{G}^{gr}_{SC}$. In order to obtain an instance of the GRAPHIC-TSP problem on cubic graphs, we use the modified parity gadgets in $\mathcal{G}^{gr}_=$ and denote this instance as $\mathcal{G}^{gr}_{CU}$.

Let us prove one direction of the reductions.

Lemma 10.13.1
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem with n wheels, 60ν equation with two variables, 2ν equations with three variables and ϕ an assignment that leaves at most $\delta\nu$ equations unsatisfied. Then, there is a tour in $\mathcal{G}^{gr}_{SC}$ and in $\mathcal{G}^{gr}_{CU}$ with cost at most $684\nu + 3(n+1) - 1 + \delta\nu$ and $1152\nu + 6(n+1) - 1 + \delta\nu$, respectively.*

Proof. Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 problem and ϕ an assignment to the variables of $\mathscr{L}$. Recall that we may assume that the



(a) Modular view of the graph $\mathcal{G}_=^{gr}$

(b) Detailed view of $\mathcal{G}_=^{gr}$

Figure 10.64: Graph $\mathcal{G}_=^{gr}$ corresponding to $a_1^1 \oplus a_1^2 = 0$.

variables associated to a wheel in $\mathscr{L}$ take the same value under ϕ. Let us start with the description of the tour in $\mathcal{G}_{SC}^{gr}$. As for the inner loop, we use the same tour as in Lemma 10.11.2. Note that we traversed only edges with weight 1 in the inner loop of the tour in $\mathcal{G}_{SC}^{12}$. In the outer loop, we cannot use the same shortcuts as in the $(1,2)$-TSP problem, since in some cases the weight of an edge can be greater than 2. To ensure that the cost traversing a gadget corresponding to an equation with three variables increases only by one if the equation is unsatisfied by the assignment, we will use the following trick: Consider an equation of the form $x \oplus y \oplus z = 0$ that is simulated by $(x \vee a_1^1 \vee a_2^1)$, $(y \vee a_2^2 \vee a_3^1)$, $(z \vee a_1^2 \vee a_3^2)$, $a_1^1 \oplus a_1^2 = 0$, $a_2^1 \oplus a_2^2 = 0$ and $a_3^1 \oplus a_3^2 = 0$. If we have an assignment that satisfies $x \oplus y \oplus z = 0$, then there is also an assignment that satisfies all 6 associated predicates. Furthermore, we see that in the other case, we can find an assignment that satisfies all predicates except exactly one equation with two variables.

In particular, it implies for a tour traversing the gadget $\mathcal{G}_=^{gr}$ simulating $a_1^1 \oplus a_2^1 = 0$ that if $(a_1^1 + a_2^1 = 0)$ and $(a_1^1 + a_2^1 = 2)$ holds, we use $s_= - c_2 - c_1 - e_=$

and $s_= - c_2 - c_1 - e_=$, respectively. On the other hand, assuming $(a^1_1 + a^1_2 = 1)$, we traverse either $s_= - c_1 - c_2 - c_1 - e_=$ or $s_= - c_2 - c_1 - c_2 - e_=$. Thus, we use the edge $\{c_1, c_2\}$ twice increasing the cost only by 1.

Summarizing, given an assignment leaving $\delta\nu$ equations unsatisfied, we find a tour in $\mathcal{G}^{12}_{SC}$ with cost at most $672\nu + 3(n+1) - 1 + \delta\nu$ and a tour in $\mathcal{G}^{gr}_{SC}$ with cost at most $684\nu + 3(n+1) - 1 + \delta\nu$, since we have to take into account the small detour and add $3 \cdot 2 \cdot 2\nu$ to the cost.

Under the same conditions, we find a tour in $\mathcal{G}^{12}_{CU}$ with cost at most $1140\nu + 6(n+1) - 1 + \delta\nu$ and a tour in $\mathcal{G}^{gr}_{CU}$ with cost at most $1152\nu + 6(n + 1) - 1 + \delta\nu$. ■

10.13.2 Tours to Assignments

We now give the other direction of the reductions and prove the following lemma.

Lemma 10.13.2
Let $\mathscr{L}$ be an instance of the MAX-HYBRID-LIN2 *problem with n wheels, 60ν equation with two variables, 2ν equations with three variables, π a tour in $\mathcal{G}^{gr}_{SC}$ with cost $684\nu + 3(n+1) - 1 + \delta\nu$ and σ a tour in $\mathcal{G}^{gr}_{CU}$ with cost $1152\nu + 6(n+1) - 1 + \delta\nu$. By using either π or σ, it is possible to construct efficiently an assignment that leaves at most $\delta\nu$ equations in $\mathscr{L}$ unsatisfied.*

Proof. Let us consider a tour π in $\mathcal{G}^{gr}_{SC}$ with cost $684\nu + 3(n+1) - 1 + \delta\nu$. We interpret π as a (1,2)-tour in $\mathcal{G}^{gr}_{SC}$ with cost at most $684\nu + 3(n+1) - 1 + \delta\nu$. In the first step, we convert the underlying tour in $\mathcal{G}^{gr}_{SC}$ into a consistent one without increasing its cost by applying Lemma 10.9.1 to each parity gadget in $\mathcal{G}^{gr}_{SC}$. In the second step, we use the same 0/1-traversals of the parity gadgets in the inner loop which enables us to construct a tour in the corresponding instance $\mathcal{G}^{12}_{SC}$ with cost at most $672\nu + 3(n+1) - 1 + \delta\nu$. Finally, we apply Lemma 10.11.3 and construct an assignment leaving at most $\delta\nu$ equations in $\mathscr{L}$ unsatisfied.

Analogously, if we have given a tour in $\mathcal{G}^{gr}_{SC}$ with cost $1152\nu + 6(n+1) - 1 + \delta\nu$, we convert it into a (1,2)-tour without increasing its cost. By applying the contractions defined in Lemma 10.12.2, we obtain a (1,2)-tour in $\mathcal{G}_{SC}^{gr}$ with cost at most $684\nu + 3(n+1) - 1 + \delta\nu$, for which we already know how to construct an assignment with the desired properties. ■

By combining Lemma 10.13.1 and Lemma 10.13.2, we obtain immediately Theorem 10.6.5.

10.14 The Metric TSP Problem

This section is devoted to the proof of the following theorem.

Theorem 10.6.6
It is **NP***-hard to approximate the* TSP *problem to within any constant approximation ratio less than* 123/122.

Let us first give an general overview of the reduction and the new techniques.

10.14.1 The General Overview of the Reduction

The hardness proof proceeds in two steps. First, we start from the MAX-E3LIN2 problem. Optimal inapproximability results for this problem were shown by Håstad [H01]. We reduce this problem to a special CSP where variables appear exactly 3 times. The main tool here is a new variant of the wheel amplifier graphs of Berman and Karpinski [BK01]. The construction of the bi-wheel amplifier is described in Section 10.14.3. In the second step (Section 10.14.5), we reduce this bounded occurrence CSP to the TSP problem and manage to obtain an improvement by exploiting the special properties of the bounded occurrence CSP. In particular, we show that it is only necessary to construct gadgets for roughly one third of the constraints of the CSP instance, while the remaining constraints are simulated without additional cost using the consistency properties of our gadgets.

In Section 10.15, we use our approach to derive the best up to now approximation lower bound for the ATSP problem.

Thus, overall we follow an approach unlike that of [PV06], where the reduction is performed in one step, and closer to [L12]. The improvement over [L12] comes mainly from the idea mentioned above, which is made possible using the new wheel amplifiers, as well as several other tweaks. The end result is a more economical reduction which improves the bounds for both the TSP and ATSP problem.

An interesting question may be whether our techniques can also be used to derive improved inapproximability results for variants of the ATSP and TSP problem (cf. [EK06], [KS13] and [KS12]), or other graph problems, such as the STEINER TREE problem.

10.14.2 Notations and Conventions

In the following, we give some definitions concerning directed (multi-)graphs and omit the corresponding definitions for undirected (multi-)graphs if they follow from the directed case.

For a multiset E_T of directed edges and a vertex that is incident to an arc in E_T, we define the outdegree (indegree) of v with respect to E_T, denoted by $outd_T(v)$ ($ind_T(v)$), to be the number of edges in E_T that are outgoing of (incoming to) v. The *balance* of a vertex v with respect to E_T is defined as $bal_T(v) = ind_T(v) - outd_T(v)$. In the case of a multiset E_T of undirected edges, we define the balance $bal_T(v)$ of a vertex $v \in V(E_T)$ to be one if the number of incident edges in E_T is odd and zero otherwise. We refer to vertices $v \in V(E_T)$ with $bal_T(v) = 0$ as *balanced* with respect to E_T. It is well known that a (directed) (multi-)graph $\mathcal{G}$ is Eulerian if and only if all edges are in the same (weakly) connected component and all vertices $v \in V(\mathcal{G})$ are balanced with respect to $E(\mathcal{G})$.

Given a multiset of edges E_T, we denote by con_T the number of (weakly) connected components in the graph induced by E_T. A *quasi-tour* E_T in a (directed) graph $\mathcal{G}$ is a multiset of edges from $E(\mathcal{G})$ such that all vertices are balanced with respect to E_T and $V(E_T) = V(\mathcal{G})$. We refer to a quasi-tour E_T in $\mathcal{G}$ as a *tour* if $con_T = 1$. Given a cost function $w : E(\mathcal{G}) \to \mathbb{R}_+$, the cost of a quasi-tour E_T in $\mathcal{G}$ is defined by $\sum_{e \in E_T} w(e) + 2(con_T - 1)$.

In this section, we will use the following equivalent reformulation of the ATSP problem: Given a directed graph $\mathcal{G}$ with weights on edges, we want to find a tour E_T in $\mathcal{G}$, that is, a spanning connected multi-set of edges that balances all vertices, with minimum cost.

10.14.3 Bi-Wheel Amplifiers

In this section, we define the bi-wheel amplifier graphs which will be our main tool for proving hardness of approximation for a bounded occurrence CSP with some special properties. Bi-wheel amplifiers are a variation of the wheel amplifier graphs given in [BK01]. Let us first recall the definition of a regular amplifier graph (see also Definition 4.9.2).

Let $\mathcal{G}$ be a graph and $X \subset V(\mathcal{G})$ a set of vertices. Then, we say that $\mathcal{G}$ is a Δ-*regular amplifier* for X if the following conditions hold:

- All vertices of X have degree $\Delta - 1$ and all vertices of $V(\mathcal{G})\backslash X$ have degree Δ.
- For every non-empty subset $U \subset V(\mathcal{G})$, we have the condition that $|E(U, V(\mathcal{G})\backslash U)| \geq \min\{\,|U \cap X|, |(V(\mathcal{G})\backslash U) \cap X|\,\}$, where $E(U, V(\mathcal{G})\backslash U)$ is the set of edges with exactly one endpoint in U.

We refer to the set X as the set of *contact* vertices and to $V(\mathcal{G})\backslash X$ as the set of *checker* vertices. Amplifier graphs are useful in proving inapproximability for CSPs, in which every variable appears a bounded number of times. Here, we will rely on 3-regular amplifiers. A probabilistic argument for the existence of such graphs was given in [BK01], with the definition of wheel amplifiers.

A *wheel amplifier* with $2n$ contact vertices is constructed as follows: first construct a cycle on $14n$ vertices. Number the vertices $1, 2, \ldots, 14n$ and select uniformly at random a perfect matching of the vertices whose number is not a multiple of 7. The matched vertices will be our checker vertices, and the rest our contacts. It is easy to see that the degree requirements are satisfied.

Berman and Karpinski [BK01] gave a probabilistic argument to prove that with high probability the above construction indeed produces an amplifier graph, that is, all partitions of the sets of vertices give large cuts.

Theorem 10.14.1 ([BK01])
With high probability, wheel amplifiers are 3-regular amplifiers.

Here, we will use a slight variation of this construction, called a bi-wheel.

Definition 10.14.1 (Construction of a Bi-wheel Amplifier)
A bi-wheel amplifier with $2n$ contact vertices can be generated as follows: First, we construct two disjoint cycles, each on $7n$ vertices and number the vertices of each $1, 2, \ldots, 7n$. The contacts are the vertices whose number is a multiple of 7, while the remaining vertices are checkers. To complete the construction, select uniformly at random a perfect matching from the checkers of one cycle to the checkers of the other.

Intuitively, the reason that amplifiers are a suitable tool here is that, given a CSP instance, we can use a wheel amplifier to replace a variable that appears $2n$ times with $14n$ new variables (one for each wheel vertex) each of which appears 3 times. Each appearance of the original variable is represented by a contact vertex and for each edge of the wheel we add an equality constraint between the corresponding variables. We can then use the property that all partitions give large cuts to argue that in an optimal assignment all the new vertices take the same value.

We use the bi-wheel amplifier in our construction in a similar way. The main difference is that while cycle edges will correspond to equality constraints, matching edges will correspond to inequality constraints. The contacts of one cycle will represent the positive appearances of the original variable, and the contacts of the other the negative ones. The reason we do this is that we can encode inequality constraints more efficiently than equality with a TSP gadget, while the equality constraints that arise from the cycles will be encoded in our construction “for free” using the consistency of the inequality gadgets.

Before we apply the construction however, we have to prove that the bi-wheel amplifiers still have the desired amplification properties.

Theorem 10.14.2

With high probability, bi-wheels are 3-regular amplifiers.

Proof of Theorem 10.14.2.

Exploiting the similarity between bi-wheels and the standard wheel amplifiers of [BK01], we will essentially reuse the proof given there. First, some definitions: We say that U is a *bad set* if the size of its cut is too small, violating the second property of amplifiers. We say that it is a *minimal bad set* if U is bad but removing any vertex from U gives a set that is not bad.

Recall the strategy of the proof from [BK01]: for each partition of the vertices into U and $V(\mathcal{G})\backslash U$, they calculate the probability (over the random matchings) that this partition gives a minimal bad set. Then, they take the sum of these probabilities over all potentially minimal bad sets and prove that the sum is at most γ^{-n} for some constant $\gamma < 1$. It follows by union bound that with high probability, no set is a minimal bad set and therefore, the graph is a proper amplifier.

Our first observation is the following: consider a wheel amplifier on $14n$ vertices where, rather than selecting uniformly at random a perfect matching among the checkers, we select uniformly at random a perfect matching from checkers with labels in the set $\{1, \ldots, 7n-1\}$ to checkers with labels in the set $\{7n, \ldots, 14n\}$. This graph is almost isomorphic to a bi-wheel. More specifically, for each bi-wheel, we can obtain a graph of this form by rewiring two edges, and vice-versa. It easily follows that properties that hold for this graph, asymptotically with high probability also hold for the bi-wheel.

Thus, we just need to prove that a wheel amplifier still has the amplification property if, rather than selecting a random perfect matching, we select a random matching from one half of the checker vertices to the other. We will show this by proving that, for each set of vertices S, the probability that S is a minimal bad set is roughly the same in both cases. After establishing this fact, we can simply rely on the proof of [BK01].

Recall that the wheel has $12n$ checker vertices. Given a set S with $|S| = u$, what is the probability that exactly c edges have exactly one endpoint in S?

In a standard wheel amplifier, the probability is

$$P(u,c) = \binom{u}{c}\binom{12n-u}{c}\frac{c!(u-c)!!(12n-u-c)!!}{(12n)!!},$$

where we denote by $n!!$ the product of all odd natural numbers less than or equal to n, and we assume without loss of generality that $u-c$ is even. Let us explain this: the probability that exactly c edges cross the cut in this graph is equal to the number of ways we can choose their endpoints in S and in its complement, times the number of ways we can match the endpoints, times the number of matchings of the remaining vertices, divided by the number of matchings overall.

How much does this probability change if we only allow matchings from one half of the checkers to the other? Intuitively, we need to consider two possibilities: one is that S is a balanced set, containing an equal number of checkers from each side, while the other is that S is unbalanced. It is not hard to see that if S is unbalanced, then, we can easily establish that the cut must be large. Thus, the main interesting case is the balanced one (and we will establish this fact more formally).

Suppose that $|S| = u$ and S contains exactly $u/2$ checkers from each side. Then, the probability that there are exactly c edges crossing the cut is $P'(u,c)$, where $P'(u,c)$ is defined below.

$$P'(u,c) = \binom{u/2}{c/2}^2 \binom{12n/2 - u/2}{c/2}^2 \left(\frac{c}{2}\right)!^2 \frac{(u/2 - c/2)!\,(12n/2 - u/2 - c/2)!}{(6n)!}$$

Let us explain this. If S is balanced and there are c matching edges with exactly one endpoint in S, then, exactly $c/2$ of them must be incident on a vertex of S on each side, since the remaining vertices of S must have a perfect matching. Again, we pick the endpoints on each side, and on the complement of S, select a way to match them, select matchings on the remaining vertices and divide by the number of possible perfect matchings.

Using Stirling formulas, it is not hard to see that $\left(\frac{n}{2}\right)!^2 = \Theta(n!2^{-n}\sqrt{n})$. Also $n!! = \Theta(\left(\frac{n}{2}\right)!2^{n/2})$. It follows that P' is roughly the same as P in this case, modulo some polynomial factors which are not significant since the probabilities we are calculating are exponentially small.

Let us now also show that if S is unbalanced, the probability that it is a minimal bad set is even smaller. First, observe that if S is a minimal bad set whose cut has c edges, we have $c \leq u/6$. The reason for this is that since S is bad, then, c is smaller than the number of contacts in S minus the number of cycle edges cut. It is not hard to see that, in each fragment, that is, each subset of S made up of a contiguous part of the cycle, two cycle edges are cut. Thus, the extra edges we need for the contacts the fragment contains are at most 1/6 of its checkers.

Suppose now that S contains $u/2 + k$ checkers on one side and $u/2 - k$ checkers on the other. The probability $P''(u, c, k)$ that c matching edges have one endpoint in S is as follows.

$$\binom{\frac{u}{2}+k}{\frac{c}{2}+k}\binom{\frac{u}{2}-k}{\frac{c}{2}-k}\binom{\frac{12n-u}{2}-k}{\frac{c}{2}-k}\binom{\frac{12n-u}{2}+k}{\frac{c}{2}+k}\left(\frac{c}{2}+k\right)!\left(\frac{c}{2}-k\right)!\frac{\left(\frac{u-c}{2}\right)!\left(\frac{12n-u-c}{2}\right)!}{(6n)!}$$

The reasoning is the same as before, except we observe that we need to select more endpoints on the side where S is larger, since after we remove checkers matched to outside vertices S must have a perfect matching. Observe that for $k = 0$ this gives P'. We will show that for the range of values we care about P'' achieves a maximum for $k = 0$, and can thus be upper-bounded by (essentially) P, which is the probability that a set is bad in the standard amplifier. The rest of the proof follows from the argument given in [BK01]. In particular, we can assume that $k \leq c/2$, since $2k$ edges are cut with probability 1. To show that the maximum is achieved for $k = 0$, we look at

$$\frac{P''(u, c, k+1)}{P''(u, c, k)}.$$

We will show that this is less than 1. By using the identity $\binom{n+1}{k+1}/\binom{n}{k} = (n+1)/(k+1)$, we obtain after short calculation the following.

$$\frac{P''(u, c, k+1)}{P''(u, c, k)} = \left(1 - \frac{2k+1}{\frac{c}{2}+k+1}\right)\left(1 + \frac{2k+1}{\frac{u}{2}-k}\right)\left(1 + \frac{2k+1}{\frac{12n-u}{2}-k}\right)$$

Using the fact that $1 + x < e^x$, we end up needing to prove that the following inequality holds.

$$\frac{2k+1}{\frac{c}{2}+k+1} > \frac{2k+1}{\frac{u}{2}-k} + \frac{2k+1}{\frac{12n-u}{2}-k} \tag{10.1}$$

Combining $u \le 6n$ with the bounds of c and k we have already mentioned, the inequality (10.1) is straightforward to establish. ■

10.14.4 Special Instances of the MAX-Hybrid-Lin2 Problem

By using the bi-wheel amplifier from the previous section, we are going to prove hardness of approximation for a bounded occurrence CSP with very special properties. This particular CSP will be well-suited for constructing a reduction to the TSP problem given in the next section. As the starting point of our reduction, we make use of the inapproximability result due to Håstad [H01] for the MAX-E3LIN2 problem.

Let $\mathscr{L}_1$ be an instance of the MAX-E3LIN2 problem and $\{x_i\}_{i=1}^{\nu}$ the set of variables, that appear in $\mathscr{L}_1$. We denote by $d(i)$ the number of appearances of x_i in $\mathscr{L}_1$.

Theorem 10.14.3 ([H01])
For every $\epsilon > 0$, there exists a constant B_ε such that given an instance $\mathscr{L}_1$ of the MAX-E3LIN2 *problem with m equations and $\max_{i\in[\nu]} d(i) \le B_\varepsilon$, it is* **NP***-hard to decide whether there is an assignment that leaves at most $\varepsilon \cdot m$ equations unsatisfied, or all assignment leave at least $(0.5-\epsilon)m$ equations unsatisfied.*

Similarly to the work by Berman and Karpinski [BK99] (see also [BK01] and [BK03]), we will reduce the number of occurrences of each variable to 3. For this, we use our amplifier construction to create special instances of the MAX-HYBRID-LIN2 problem and prove the following theorem.

Theorem 10.14.4
For every constant $\varepsilon > 0$ and $b \in \{0,1\}$, there exist instances of the MAX-HYBRID-LIN2 *problem with $31m$ equations such that: (i) Each variable occurs exactly three times. (ii) $21m$ equations are of the form $x \oplus y = 0$,*

$9m$ equations are of the form $x \oplus y = 1$ and m equations are of the form $x \oplus y \oplus z = b$. (iii) It is **NP***-hard to decide whether there is an assignment to the variables that leaves at most $\varepsilon \cdot m$ equations unsatisfied, or every assignment to the variables leaves at least $(0.5 - \varepsilon)m$ equations unsatisfied.*

Proof. Let $\epsilon > 0$ be a constant and $\mathscr{L}_1$ an instance of the MAX-E3LIN2 problem with $\max_{i \in [\nu]} d(i) \leq B_\varepsilon$. For a fixed $b \in \{0, 1\}$, we can flip some of the literals such that all equations in the instance $\mathscr{L}_1$ are of the form $x \oplus y \oplus z = b$, where x, y, z are variables or negations. By constructing three more copies of each equation, in which all possible pairs of literals appear negated, we may assume that each variable occurs the same number of times negated as unnegated.

Let us fix a variable x_i in $\mathscr{L}_1$. Then, we create $7 \cdot d(i) = 2 \cdot \alpha$ new variables $Var(i) = \{x_j^{ui}, x_j^{ni}\}_{j=1}^{\alpha}$. In addition, we construct a bi-wheel amplifier $\mathcal{W}_i$ on $2 \cdot \alpha$ vertices (that is, a bi-wheel with $d(i)$ contact vertices) with the properties described in Theorem 10.14.2. Since $d(i) \leq B_\varepsilon$ is a constant, this can be accomplished in constant time. In the remainder, we refer to *contact* and *checker* variables as the elements in $Var(i)$, whose corresponding index is a contact and checker vertex in $\mathcal{W}_i$, respectively. We denote by $M(\mathcal{W}_i) \subseteq E(\mathcal{W}_i)$ the associated perfect matching on the set of checker vertices of $\mathcal{W}_i$. In addition, we denote by $C_n(\mathcal{W}_i)$ and $C_u(\mathcal{W}_i)$ the set of edges contained in the first and second cycle of $\mathcal{W}_i$, respectively.

Let us now define the equations of the corresponding instance of the MAX-HYBRID-LIN2 problem. For each edge $\{j, k\} \in M(\mathcal{W}_i)$, we create the equation $x_j^{ui} \oplus x_k^{ni} = 1$ and refer to equations of this form as *matching equations.* On the other hand, for each edge $\{l, t\}$ in the cycle $C_q(\mathcal{W}_i)$ with $q \in \{u, n\}$, we introduce the equation $x_l^{qi} \oplus x_t^{qi} = 0$. Equations of this form will be called *cycle equations.* Finally, we replace the j-th unnegated appearance of x_i in $\mathscr{L}_1$ by the contact variable x_λ^{ui} with $\lambda = 7 \cdot j$, whereas the j-th negated appearance is replaced by x_λ^{ni}. The former construction yields m equations with three variables in the instance of the MAX-HYBRID-LIN2 problem, which we will denote by $\mathscr{L}_2$. Notice that each variable appears in exactly 3 equations in $\mathscr{L}_2$. Clearly, we have $|\mathscr{L}_2| = 31m$ equations, thereof

$9m$ matching equations, $21m$ cycle equations and m equations of the form $x \oplus y \oplus z = b$.

A consistent assignment to $Var(i)$ is an assignment with $x_j^{ui} = b$ and $x_j^{ni} = (1 - b)$ for all $j \in [\alpha]$, where $b \in \{0, 1\}$. A consistent assignment to the variables of $\mathscr{L}_2$ is an assignment that is consistent to $Var(i)$ for all $i \in [\nu]$. By standard arguments using the amplifier constructed in Theorem 10.14.2, it is possible to convert an assignment to a consistent assignment without decreasing the number of satisfied equations and the proof of Theorem 10.14.4 follows. ■

10.14.5 Construction of the Instances of the TSP

In this section, we are going to describe the instance of the TSP problem given an instance the MAX-HYBRID-LIN2 problem. Furthermore, it is devoted to the proof of Theorem 10.6.6.

Let us first sketch the high-level idea of the construction. Starting with an instance of the MAX-HYBRID-LIN2 problem, we will construct a graph, where gadgets represent the equations. We will design gadgets for equations of size three (Figure 10.66) and for equations of size two corresponding to *matching* edges of the bi-wheel (Figure 10.65). We will not construct gadgets for the cycle edges of the bi-wheel; instead, the connections between the matching edge gadgets will be sufficient to encode these extra constraints. This may seem counter-intuitive at first, but the idea here is that if the gadgets for the matching edges are used in a consistent way (that is, the tour enters and exits in the intended way) then it follows that the tour is using all edges corresponding to one wheel and none from the other. Thus, if we prove consistency for the matching edge gadgets, we implicitly get the cycle edges "for free". This observation, along with an improved gadget for size-three equations and the elimination of the variable part of the graph, are the main sources of improvement over the construction of [L12].

Let us describe the construction that encodes an instance $\mathscr{L}_2$ of the MAX-HYBRID-LIN2 problem into an instance of the TSP problem: Due to Theorem 10.14.4, we may assume that the equations with three variables in

$\mathscr{L}_2$ are all of the form $x \oplus y \oplus z = 0$.

In order to ensure that some edges are to be used at least once in any valid tour, we apply the following simple trick that was already used in the work by Lampis [L12]: Let e be an edge with weight w that we want to be traversed by every tour. We remove e and replace it with a path of L edges and $L - 1$ newly created vertices each of degree two, where we think of L as a large constant. Each of the L edges has weight w/L and any tour that fails to traverse at least two newly created edges is not connected. On the other hand, a tour that traverses all but one of those edges can be extended by adding two copies of the unused edge increasing the cost of the underlying tour by a negligible value. In summary, we may assume that our construction contains *forced* edges that need to be traversed at least once by any tour. If x and y are vertices, which are connected by a forced edge e, we write $\{x, y\}_F$ or simply $x -_F y$. In the following, we refer to unforced edges e with $w(e) = 1$ as *simple*. All unforced edges in our construction will be simple.

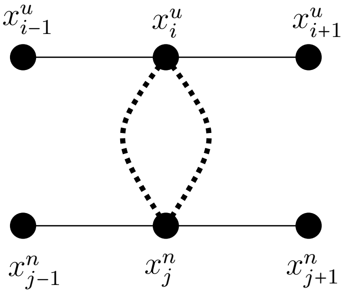


Figure 10.65: Gadget simulating equations with two variables. Dotted and straight lines represent forced and simple edges, respectively.

Let us start with the description of the corresponding graph $\mathcal{G}_S$: For each bi-wheel $\mathcal{W}_p$, we will construct the subgraph $\mathcal{G}^p$ of $\mathcal{G}_S$. For each vertex of the bi-wheel, we create a vertex in the graph and for each cycle equation $x \oplus y = 0$, we create a simple edge $\{x, y\}$. Given a matching equation between two checkers $x^u_i \oplus x^n_j = 1$, we connect the vertices x^u_i and x^n_j with two forced edges $\{x^u_i, x^n_j\}^1_F$ and $\{x^u_i, x^n_j\}^2_F$. We have $w(\{x^u_i, x^n_j\}^i_F) = 2$ for each $i \in \{1, 2\}$.

Additionally, we create a central vertex s that is connected to gadgets simulating equations with three variables. Let $x \oplus y \oplus z = 0$ be the j-th equation with three variables in $\mathscr{L}_2$. We now create the graph $\mathcal{G}_j^{3S}$ displayed in Figure 10.66, where the (contact) vertices for x, y, z have already been constructed in the cycles. The edges $\{\gamma^\alpha, \gamma\}_F$ with $\alpha \in \{r, l\}$ and $\gamma \in \{x, z, y\}$ are all forced edges with $w(\{\gamma^\alpha, \gamma\}_F) = 1.5$. Furthermore, we have $w(\{e_j^\alpha, s\}_F) = 0.5$ for all $\alpha \in \{r, l\}$. $\{e_j^r, s\}_F$ and $\{e_j^l, s\}_F$ are both forced edges, whereas all remaining edges of $\mathcal{G}_j^{3S}$ are simple. This is the whole description of $\mathcal{G}_S$.

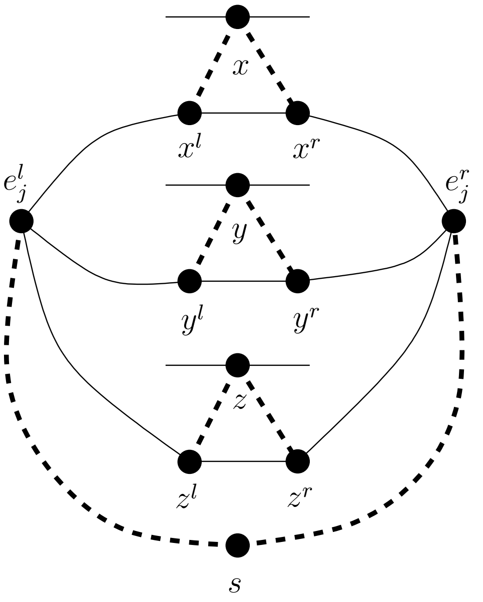


Figure 10.66: Gadgets simulating equations with three variables of the form $x \oplus y \oplus z = 0$. Dotted and straight lines represent forced and simple edges, respectively.

10.14.6 Tours in $\mathcal{G}_S$ from Assignments

Given an instance $\mathscr{L}_2$ of the MAX-HYBRID-LIN2 problem and an assignment ϕ to the variables in $\mathscr{L}_2$, we are going to construct a tour in $\mathcal{G}_S$ according to ϕ and give the proof of one direction of the reduction. In particular, we are going to prove the following lemma.

Lemma 10.14.1

If there is an assignment to the variables of a given instance $\mathscr{L}_2$ of the MAX-HYBRID-LIN2 *problem with $31m$ equations and ν bi-wheels, that leaves k equations unsatisfied, then, there exists a tour in $\mathcal{G}_S$ with cost at most $61m + 2\nu + k + 2$.*

Before we proceed, let us give a useful definition. Let $\mathcal{G}$ be an edge-weighted graph and E_T a multi-set of edges of $E(\mathcal{G})$ that defines a quasi-tour. Consider a set $V' \subseteq V(\mathcal{G})$. The local edge cost of the set V' is then defined as follows.

$$c_T(V') = \sum_{u \in V'} \sum_{e \in E_T,\ e=\{u,v\}} \frac{w(e)}{2}$$

In words, for each vertex in V', we count half the total weight of its incident edges used in the quasi-tour (including multiplicities). Observe that this sum contains half the weight of edges with one endpoint in V' but the full weight for edges with both endpoints in V' (since we count both endpoints in the sum). Also note that for two sets V_1, V_2, we have

$$c_T(V_1 \cup V_2) \le c_T(V_1) + c_T(V_2)$$

(with equality for disjoint sets) and that $c_T(V(\mathcal{G})) = \sum_{e \in E_T} w(e)$.

Proof of Lemma 10.14.1.

First, note that it is sufficient to prove that we can construct a quasi-tour of the promised cost which uses all forced edges exactly once. Since all unforced edges have cost 1, if we are given a quasi-tour we can connect two disconnected components by using an unforced edge that connects them twice (this is always possible since the underlying graph we constructed is connected). This does not increase the cost, since we added two unit-weight edges and decreased the number of components. Repeating this results in a connected tour.

Let $\{\mathcal{W}_a\}_{a=1}^{\nu}$ be the associated set of bi-wheels of $\mathscr{L}_2$. For a fixed bi-wheel $\mathcal{W}_p$, let $\{x_i^u, x_i^n\}_{i=1}^{z}$ be its associated set of variables. Due to the construction of instances of the MAX-HYBRID-LIN2 problem in Section 10.14.4, we may assume that all equations with two variables are satisfied by the given assignment. Thus, we have $x_i^u \neq x_j^n$, $x_i^u = x_j^u$ and $x_i^n = x_j^n$ for all $i, j \in [z]$.

Assuming $x_1^\alpha = 1$ for some $\alpha \in \{u, n\}$, we use once all simple edges $\{x_i^\alpha, x_{i+1}^\alpha\}$ with $i \in [z-1]$ and the edge $\{x_z^\alpha, x_1^\alpha\}$. We also use all forced edges corresponding to matching equations once. In other words, for each bi-wheel, we select the cycle that corresponds to the assignment 1 and use all the simple edges from that cycle. This creates a component that contains all checker vertices from both cycles and all contacts from one cycle.

As for the next step, we are going to describe the tour traversing $\mathcal{G}_j^{3S}$ with $j \in [m]$ given an assignment to contact variables. Let us assume that $\mathcal{G}_j^{3S}$ simulates $x \oplus y \oplus z = 0$. According to the assignment to x, y and z, we will traverse $\mathcal{G}_j^{3S}$ as follows: In all cases, we will use all forced edges once.

Case $(x + y + z = 2)$**:**
Then, we use $\{\gamma^l, \gamma^r\}$ for all $\alpha \in \{r, l\}$ and $\gamma \in \{x, y, z\}$ with $\gamma = 1$. For $\delta \in \{x, z, y\}$ with $\delta = 0$, we use $\{e_j^\alpha, \delta^\alpha\}$ for all $\alpha \in \{r, l\}$.

Case $(x + y + z = b$ **with** $b \in \{0, 1\})$**:**
In both cases, we traverse $\{\gamma^\alpha, e_j^\alpha\}$ for all $\gamma \in \{x, y, z\}$ and $\alpha \in \{r, l\}$.

Case $(x + y + z = 3)$**:**
We use $\{\gamma^r, \gamma^l\}$ with $\gamma \in \{y, z\}$. Furthermore, we include $\{x^\alpha, e_j^\alpha\}$ for both $\alpha \in \{r, l\}$.

Let us now analyze the cost of the edges of our quasi-tour given an assignment. For each matching edge $\{x_i^u, x_j^n\}$ consider the set of vertices made up of its endpoints. Its local cost is 5: we pay 4 for the forced edges and there are two used simple edges with one endpoint in the set. Let us also consider the local cost for a size-three equation gadget, where we consider the set to contain the contact vertices $\{x, y, z\}$ as well the other 8 vertices of the gadget. The local cost here is 9.5 for the forced edges. We also pay 6 more (for a total of 15.5) when the assignment satisfies the equation or 7 more when it does not.

Thus, we have given a covering of the vertices of the graph by $9m$ sets of size two, m sets of size 11 and $\{s\}$. The total edge cost is thus at most $5 \cdot 9m + 15.5 \cdot m + 0.5 \cdot m + k = 61m + k$. To obtain an upper bound on the cost of the quasi-tour, we observe that the tour has at most $\nu + 1$ components (one for each bi-wheel and one containing s). The lemma follows. ■

10.14.7 Assignments from Tours in $\mathcal{G}_S$

In this section, we are going prove the other direction of our reduction. Given a tour in $\mathcal{G}_S$, we are going to define an assignment to the variables of the associated instance of the MAX-HYBRID-LIN2 problem and give the proof of the following lemma.

Lemma 10.14.2
If there is a tour in $\mathcal{G}_S$ with cost $61m + k - 2$, then, there is an assignment to the variables of the corresponding instance of the MAX-HYBRID-LIN2 *problem that leaves at most k equations unsatisfied.*

Again, let us give a useful definition. Consider a quasi-tour E_T and a set $V' \subseteq V(\mathcal{G})$. Let $con_T(V')$ be the number of connected components induced by E_T which are fully contained in V'. Then, the full local cost of the set V' is defined as $c_T^F(V') = c_T(V') + 2 \cdot con_T(V')$. By the definition, the full local cost of $V(\mathcal{G})$ is equal to the cost of the quasi-tour (plus 2).

Intuitively, $c_T^F(V')$ captures the cost of the quasi-tour restricted to V': it includes the cost of edges and the cost of added connected components. Note that now for two disjoint sets V_1, V_2, we have $c_T^F(V_1 \cup V_2) \geq c_T^F(V_1) + c_T^F(V_2)$ since $V_1 \cup V_2$ could contain more connected components than V_1, V_2 together. If we know that the total cost of the quasi-tour is small, then $c_T^F(V(\mathcal{G}))$ is small (less than $61m + k$). We can use this to infer that the sum of the local full costs of all gadgets is small.

The high-level idea of the proof is the following: we will use the same partition of $V(\mathcal{G})$ into sets as in the proof of Lemma 10.14.1. For each set, we will give a lower bound on its full local cost for any quasi-tour, which will be equal to what the tour we constructed in Lemma 10.14.1 pays. If a given quasi-tour behaves differently its local cost will be higher. The difference between the actual local cost and the lower bound is called the credit of that part of the graph. We construct an assignment for the variables of $\mathscr{L}_2$ and prove that the total sum of credits is higher that the number of unsatisfied equations. But using the reasoning of the previous paragraph, the total sum of credits will be at most k.

Proof of Lemma 10.14.2. We are going to prove a slightly stronger statement and show that if there exists a quasi-tour in $\mathcal{G}_S$ with cost $61m + k - 2$, then, there exists an assignment leaving at most k equations unsatisfied. Recall that the existence of a tour in $\mathcal{G}_S$ with cost C implies the existence of a quasi-tour in $\mathcal{G}_S$ with cost at most C.

We may assume that simple edges are contained only once in E_T due to the following preprocessing step: If E_T contains two copies of the same simple edge, we remove them without increasing the cost, since the number of components can only increase by one.

In the following, given a quasi-tour E_T in $\mathcal{G}_S$, we are going to define an assignment ϕ_T and analyze the number of satisfied equations by ϕ_T compared to the cost of the quasi-tour.

The general idea is that each vertex of $\mathcal{G}_S$ that corresponds to a variable of $\mathscr{L}_2$ has exactly two forced and exactly two simple edges incident to it. If the forced edges are used once each, the variable is called honest. We set it to 1 if the simple edges are both used once and to 0 otherwise. It is not hard to see that, because simple cycle edges connect vertices that represent the variables, this procedure will satisfy all cycle equations involving honest variables. We then argue that if other equations are unsatisfied the tour is also paying extra, and the same happens if a variable is dishonest.

Let us give more details. First, we concentrate on the assignment for checker variables.

Assignment for Checker Variables

Let us consider the following equations with two variables $x^u_{i-1} \oplus x^u_i = 0$, $x^u_i \oplus x^u_{i+1} = 0$, $x^n_{j-1} \oplus x^n_j = 0$, $x^n_j \oplus x^n_{j+1} = 0$ and $x^u_i \oplus x^n_j = 1$. We are going to analyze the cost of a quasi-tour traversing the gadget displayed in Figure 10.65 and define an assignment according to E_T. Let us first assume that our quasi-tour is honest, that is, the underlying quasi-tour traverses forced edges only once.

Honest tours: For $x \in \{x_i^u, x_j^n\}$, we set $x = 1$ if the quasi-tour traverses both simple edges incident on x and $x = 0$, otherwise. Since we removed all copies of the same simple edge, we may assume that cycle equations are always satisfied. If the tour uses $x_{i-1}^u - x_i^u -_F x_j^n -_F x_i^u - x_{i+1}^u$, we get $x_{i-1}^u = x_{i+1}^u = 1$, $x_{j-1}^n = x_{j+1}^n = 0$ and 5 satisfied equations. Given $x_{j-1}^n - x_j^n -_F x_i^u -_F x_j^n - x_{j+1}^n$, we obtain 5 satisfied equations as well. Let us define $V_i^p := \{x_i^u, x_j^n\}$. Notice that in both cases, we have local cost $c_T^F(V_i^p) = 5$. We claim that $c_T^F(V_i^p) \geq 5$ for a valid quasi-tour. In order to obtain a valid quasi-tour, we need to traverse both forced edges in $\mathcal{G}_i^p$ and use at least two simple edges, as otherwise, it implies $c_T^F(\mathcal{G}_i^p) \geq 6$. Given a quasi-tour E_T, we introduce a local credit function defined by $cr_T(V_i^p) = c_T^F(V_i^p) - 5$. If $x_i^u -_F x_j^n -_F x_i^u$ forms a connected component, we get 4 satisfied equations and $cr_T(V_i^p) = 1$, which is sufficient to pay for the unsatisfied equation $x_i^u \oplus x_j^n = 1$. On the other hand, assuming $x_{i-1}^u = x_{i+1}^u = 1$ and $x_{j-1}^n = x_{j+1}^n = 1$, we get $cr_T(V_i^p) = 1$ and 1 unsatisfied equation.

Dishonest tours: We are going to analyze quasi-tours, which are using one of the forced edges twice. By setting $x_i^u \neq x_j^n$, we are able to find an assignment that always satisfies $x_i^u \oplus x_j^n = 1$ and two other equations out of the five that involve these dishonest variables. The local cost in this case is at least 7. Hence, the credit $cr_T(V_i^p) = 2$ is sufficient to pay for the two unsatisfied equations.

Assignment for Contact Variables

Again, we will distinguish between honest tours (which use forced edges exactly once) and dishonest tours. This time we are interested in seven equations: the size-three equation $x \oplus y \oplus z = 0$ and the six cycle equations containing the three contacts.

Observe that the local cost of $V_j^{3S} := \{x^r, x^l, x, y^r, y^l, y, z^r, z^l, z, e_j^r, e_j^l\}$ is at least 15.5. The local edge cost of any quasi-tour is 9.5 for the forced edges. For each component $\{\gamma, \gamma^l, \gamma^r\}$ with $\gamma \in \{x, y, z\}$, we need to pay at least 2 more because there are two vertices with odd degree (γ^l, γ^r) and we also need to connect the component to the rest of the graph (otherwise the component already costs 2 more). Let us define the credit of V_j^{3S} with respect to E_T by $cr_T = c_T^F(V_j^{3S}) - 15.5$.

Honest tours:
For each $\gamma \in \{x, y, z\}$, we set $\gamma = 1$ if the tour uses both simple edges incident on γ and 0, otherwise. Notice that in the case $(x + y + z = b)$ with $b \in \{0, 2\}$, this satisfies all seven equations and the tour has local cost at least $c_T^F(V_j^{3S}) = 15.5$.
Case $(x = y = z = 1)$**:** The assignment now failed to satisfy the size-three equation, so we need to prove that the quasi-tour has local cost at least 16.5. Since all vertices are balanced with respect to E_T, the quasi-tour has to use at least one edge incident on e_j^r and e_j^l besides $\{s, e_j^r\}_F$ and $\{s, e_j^l\}_F$. If the quasi-tour takes $\{e_j^\alpha, \gamma^\alpha\}$ for a $\gamma \in \{x, y, z\}$ and all $\alpha \in \{r, l\}$, since all simple edges incident on x, y, z are used, we get at total cost of at least 16.5, which gives a credit of 1.
Case $(x + y + z = 1)$**:** Without loss of generality, we assume that $x = y = 0 \neq z$ holds. Again, only the size-three equation is unsatisfied, so we must show that the local cost is at least 16.5. We will discuss two subcases. (i) There is a connected component $\delta -_F \delta^r - \delta^l -_F \delta$ for some $\delta \in \{x, y\}$. We obtain that $c_T^F(\{\delta, \delta^l, \delta^r\}) \geq 6$ and therefore, a lower bound on the total cost of 16.5. (ii) Since we may assume that x^r, x^l, y^r and y^l are balanced with respect to E_T, we have that $\{e_j^\alpha, \gamma^\alpha\} \in E_T$ for all $\alpha \in \{r, l\}$ and $\gamma \in \{x, y\}$. Because e_j^α are also balanced, we obtain $\{e_j^\alpha, z^\alpha\} \in E_T$ for all $\alpha \in \{r, l\}$, which implies a total cost of 16.5.

Dishonest tours:
Let us assume that the quasi-tour uses both of the forced edges $\{\gamma^r, \gamma\}$ and $\{\gamma^l, \gamma\}$ for some $\gamma \in \{x, z, y\}$ twice. We delete both copies and add $\{\gamma^r, \gamma^l\}$ instead which reduces the cost of the quasi-tour. Hence, we may assume that only one of the two incident forced edges is used twice.

First, observe that if all forced edges were used once, then there would be eight vertices in the gadget with odd degree: $x^r, x^l, y^r, y^l, z^r, z^l, e_j^r, e_j^l$. If exactly one forced edge is used twice, then seven of these vertices have odd degree. Thus, it is impossible for the tour to make the degrees of all seven even using only the simple edges that connect them. We can therefore assume that if a forced edge is used twice, there exists another forced edge used twice.

We will now take cases, depending on how many of the vertices x, y, z are incident on forced edges used twice. Note that if one of the forced edges incident on x is used twice, then exactly one of the simple edges incident on x is used once.

First, suppose all three of x, y, z have forced edges used twice. The local cost from forced edges is at least 14. Furthermore, there are three vertices of the form γ^{α}, for $\gamma \in \{x, y, z\}$ and $\alpha \in \{l, r\}$ with odd degree. These have no simple edges connecting them, thus the quasi-tour will use three simple edges to balance their degrees. Finally, the used simple edges incident on x, y, z each contribute 0.5 to the local cost. Thus, the total local cost is at least 18.5, giving us a credit of 3. It is not hard to see that there is always an assignment satisfying four out of the seven affected equations, so this case is done.

Second, suppose exactly two of x, y, z have incident forced edges used twice, say, x, y. For z, we select the honest assignment (1 if the incident simple edges are used, 0 otherwise) and this satisfies the cycle equations for this variable. We can select assignments for x, y that satisfy three of the remaining five equations, so we need to show that the cost in this case is at least 17.5. The cost of forced edges is at least 12.5, and the cost of simple edges incident on x, y adds 1 to the local cost. One of the vertices x^l, x^r and one of y^l, y^r have odd degree, therefore the cost uses two simple edges to balance them. Finally, the vertices z^l, z^r have odd degree. If two simple edges incident to them are used, we have a total local cost of 17.5. If the edge connecting them is used, then the two simple edges incident on z must be used, again pushing the local cost to 17.5.

Finally, suppose only x has an incident forced edge used twice. By the parity argument given above, this means that one of the forced edges incident on s is used twice. We can satisfy the cycle equations for y, z by giving them their honest assignment, and out of the three remaining equations some assignment to x satisfies two. Therefore, we need to show that the cost is at least 16.5. The local cost from forced edges is 11.25 and the simple edge incident on x contributes 0.5. Also, at least one simple edge incident on x^l or x^r is used, since one of them has odd degree. For y^l, y^r, either two simple edges are used, or if the edge connecting them is used the simple edges incident on y contribute 1 more. With similar reasoning for z^l, z^r, we get that the total local cost is at least 16.75.

Let us now conclude our analysis. Consider the following partition of $V(\mathcal{G}_S)$: we have a singleton set $\{s\}$, $9m$ sets of size 2 containing the matching edge gadgets and m sets of size 11 containing the gadgets for size-three equations (except s). The sum of their local costs is at most $c_T^F(V(\mathcal{G}_S)) \leq 61m + k$. But the sum of their local costs is (using the preceding analysis) equal to $61m + \sum cr_T(V_i)$. Thus, the sum of all credits is at most k. Since we have already argued that the sum of all credits is enough to cover all equations unsatisfied by our assignment, this concludes the proof. ∎

We are ready to give the proof of Theorem 10.6.6.

Proof of Theorem 10.6.6.
We are given an instance $\mathscr{L}_1$ of the MAX-E3LIN2 problem with ν variables and m equations. For all $\delta > 0$, there exists a k such that if we repeat each equation k time we get an instance $\mathscr{L}_1^{(k)}$ with $m' = km$ equations and ν variables such that $2(\nu+1)/m' \leq \delta$.

Then, from $\mathscr{L}_1^{(k)}$, we generate an instance $\mathscr{L}_2$ of the MAX-HYBRID-LIN2 problem and the corresponding graph $\mathcal{G}_S$. Due to Lemmata 10.14.1, 10.14.2 and Theorem 10.14.4, we know that for all $\varepsilon > 0$, it is **NP**-hard to tell whether there is a tour with cost at most $61m' + 2\nu + 2 + \varepsilon \cdot m' \leq 61 \cdot m' + (\delta + \varepsilon)m'$ or all tours have cost at least $61m' + (0.5 - \varepsilon)m' - 2 \geq 61.5 \cdot m' - \varepsilon \cdot m' - \delta \cdot m'$. The ratio between these two cases can get arbitrarily close to 123/122 by appropriate choices for ϵ, δ. ∎

10.15 The Asymmetric TSP Problem

In this section, we are going to construct a reduction from the MAX-HYBRID-LIN2 problem to the ATSP problem and prove the following theorem.

Theorem 10.6.7
It is **NP***-hard to approximate the* ATSP *problem to within any constant approximation ratio less than* 75/74.

10.15.1 The Construction of the Graph $\mathcal{G}_A$

Let us describe the construction that encodes an instance $\mathscr{L}_2$ of the MAX-HYBRID-LIN2 problem into an instance of the ATSP problem. Again, it will be useful to have the ability to force some edges to be used, that is, we would like to have bidirected forced edges. A bidirected forced edge of weight w between two vertices x and y will be created in a similar way as undirected forced edges in the previous section: construct $L-1$ new vertices and connect x to y through these new vertices, making a bidirected path with all edges having weight w/L. It is not hard to see that without loss of generality, we may assume that all edges of the path are used in at least one direction, though we should note that the direction is not prescribed. In the remainder, we denote a directed forced edge consisting of vertices x and y by $(x,y)_F$, or $x \to_F y$.

Let $\mathscr{L}_2$ consist of the collection $\{\mathcal{W}_i\}_{i=1}^{\nu}$ of bi-wheels. Recall that the bi-wheel consists of two cycles and a perfect matching between their checkers. Let $\{x_i^u, x_i^n\}_{i=1}^{z}$ be the associated set of variables of $\mathcal{W}_p$. We write $u(i)$ to denote the function which, given the index of a checker variable x_i^u returns the index j of the checker variable x_j^n to which it is matched (that is, the function u is a permutation function encoding the matching). We write $n(i)$ to denote the inverse function $u^{-1}(i)$.

Now, for each bi-wheel $\mathcal{W}_p$, we are going to construct the corresponding directed graph $\mathcal{G}_A^p$ as follows. First, construct a vertex for each checker variable of the wheel. For each matching equation $x_i^u \oplus x_j^n = 1$, we create a bidirected forced edge $\{x_i^u, x_j^n\}_F$ with $w(\{x_i^u, x_j^n\}_F) = 2$. For each contact variable x_k, we create two corresponding vertices x_k^r and x_k^l, which are joined by the bidirected forced edge $\{x_k^r, x_k^l\}_F$ with $w(\{x_k^r, x_k^l\}_F) = 1$.

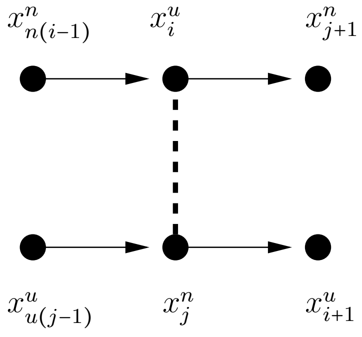


Figure 10.67: Gadget simulating equation with two variables. Dotted and straight lines represent forced and simple edges, respectively.

Next, we will construct two directed cycles $\mathcal{C}^p_u$ and $\mathcal{C}^p_n$. Note that we are doing arithmetic on the cycle indices here, so the index $z+1$ should be read as equal to 1. For $\mathcal{C}^p_u$, for any two consecutive checker vertices x^u_i, x^u_{i+1} on the un-negated side of the bi-wheel, we add a simple directed edge $x^n_{u(i)} \to x^u_{i+1}$. If the checker x^u_i is followed by a contact x^u_{i+1} in the cycle, then we add two simple directed edges $x^n_{u(i)} \to x^{ur}_{i+1}$ and $x^{ul}_{i+1} \to x^u_{i+2}$. Observe that by traversing the simple edges we have just added, the forced matching edges in the direction $x^u_i \to_F x^n_{u(i)}$ and the forced contact edges for the un-negated part in the direction $x^{ur}_i \to_F x^{ul}_i$ we obtain a cycle that covers all checkers and all the contacts of the un-negated part.

We now add simple edges to create a second cycle $\mathcal{C}^p_n$. This cycle will require using the forced matching edges in the opposite direction and, thus, truth assignments will be encoded by the direction of traversal of these edges. First, for any two consecutive checker vertices x^n_i, x^n_{i+1} on the un-negated side of the bi-wheel, we add the simple directed edge $x^u_{n(i)} \to x^n_{i+1}$. Then, if the checker x^n_i is followed by a contact x^n_{i+1} in the cycle then we add the simple directed edges $x^u_{n(i)} \to x^{nr}_{i+1}$ and $x^{nl}_{i+1} \to x^n_{i+2}$. Now by traversing the edges we have just added, the forced matching edges in the direction $x^n_i \to_F x^u_{n(i)}$ and the forced contact edges for the negated part in the direction $x^{nr}_i \to_F x^{nl}_i$, we obtain a cycle that covers all checkers and all the contacts of the negated part, that is, a cycle of direction opposite to $\mathcal{C}^p_u$.

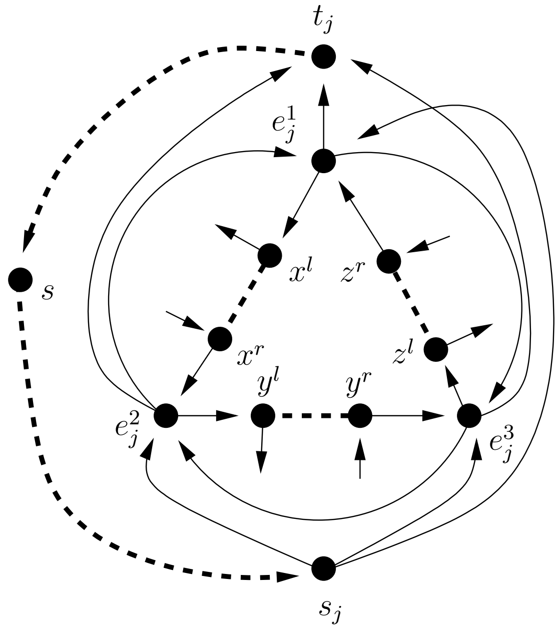


Figure 10.68: Gadgets simulating equations with three variables of the form $x \oplus y \oplus z = 1$. Dotted and straight lines represent forced and simple edges, respectively.

What is left is to encode the equations of size three. Again, we have a central vertex s that is connected to gadgets simulating equations with three variables. For every equation with three variables, we create the gadget displayed in Figure 10.68, which is a variant of the gadget used by Papadimitriou and Vempala [PV06]. Let us assume that the j-th equation with three variables in $\mathscr{L}_2$ is of the form $x \oplus y \oplus z = 1$. This equation is simulated by $\mathcal{G}_j^{3A}$. The vertices used are the contact vertices $\gamma^\alpha, \gamma \in \{x, y, z\}, \alpha \in \{r, l\}$, which we have already introduced, as well as the vertices $\{s_j, t_j, e_j^i \mid i \in [3]\}$. For notational simplicity, we define the set V_j^{3A} of vertices as follows.

$$V_j^{3A} = \left\{s_j, t_j, e_j^i, \gamma^\alpha \mid i \in [3], \gamma \in \{x, y, z\}, \alpha \in \{r, l\}\right\}$$

All directed non-forced edges are simple. The vertices s_j and t_j are connected to s by forced edges with $w((s, s_j)_F) = w((t_j, s)_F) = \lambda$, where $\lambda > 0$ is a small fixed constant. This is the whole description of the graph $\mathcal{G}_A$.

10.15.2 Assignments to Tours in $\mathcal{G}_A$

We are going to construct a tour in $\mathcal{G}_A$ given an assignment to the variables of $\mathscr{L}_2$ and prove the following lemma.

Lemma 10.15.1
Given an instance $\mathscr{L}_2$ of the MAX-HYBRID-LIN2 *problem with ν bi-wheels and an assignment that leaves k equations in $\mathscr{L}_2$ unsatisfied, then, there exists a tour in $\mathcal{G}_A$ with cost at most $37m + 5\nu + 2m\lambda + 2\nu\lambda + k$.*

Before we proceed, let us again give a definition for a local edge cost function. Let $\mathcal{G}$ be an edge-weighted digraph and E_T a multi-set of edges of $E(\mathcal{G})$ that defines a tour. Consider a set $V' \subseteq V(\mathcal{G})$. The local edge cost of the set V' is then defined as follows.

$$c_T(V') = \sum_{u \in V'} \sum_{(u,v) \in E_T} w\big((u,v)\big)$$

In words, for each vertex in V' we count the total weight of its outgoing edges used in the quasi-tour (including multiplicities). Thus, that this sum contains the full weight for edges with their source in V', regardless of where their other endpoint is. Also note that again for two sets V_1, V_2, we have $c_T(V_1 \cup V_2) \leq c_T(V_1) + c_T(V_2)$ (with equality for disjoint sets) and that $c_T(V(\mathcal{G})) = \sum_{e \in E_T} w(e)$.

Proof of Lemma 10.15.1.
Let $\mathcal{W}_p$ be a bi-wheel with variables $\{x_i^u, x_i^n\}_{i=1}^z$. Given an assignment to the variables of $\mathscr{L}_2$, due to Theorem 10.14.4, we may assume that either $x_i^u = 1 \neq x_j^n$ for all $i, j \in [z]$ or $x_i^u = 0 \neq x_j^n$ for all $i, j \in [z]$. We traverse the cycle $\mathcal{C}_u^p$ if $x_1^u = 1$ and the cycle $\mathcal{C}_n^p$ otherwise. This creates ν strongly connected components. Each contains all the checkers of a bi-wheel and the contacts from one side.

For each matching edge gadget, the local edge cost is 3. We pay two for the forced edge and 1 for the outgoing simple edge. We will account for the cost of edges incident on contacts when we analyze the size-three equation gadget below.

Let us describe the part of the tour traversing the graph $\mathcal{G}_j^{3A}$, which simulates $x \oplus y \oplus z = 1$. Recall that if x is set to true in the assignment we have traversed the bi-wheel gadgets in such a way that the forced edge $x^r \rightarrow_F x^l$ is used, and the simple edge coming out of x^l is used. According to the assignment to x, y and z, we traverse $\mathcal{G}_j^{3A}$ as follows:

Case $(x + y + z = 1)$**:**
Let us assume that $z = y = 0 \neq x$ holds. Then, we use $s \rightarrow_F s_j \rightarrow e_j^2 \rightarrow y^l \rightarrow_F y^r \rightarrow e_j^3 \rightarrow z^l \rightarrow_F z^r \rightarrow e_j^1 \rightarrow t_j \rightarrow_F s$. The cost is $3 + \lambda$ for the forced edges, 6 for the simple edges inside the gadget, plus 1 for the simple edge going out of x^l. Total local edge cost cost: $c_T(V_j^{3A}) = 10 + \lambda$.

Case $(x + y + z = 3)$**:**
Then, we use $s \rightarrow_F s_j \rightarrow e_j^2 \rightarrow e_j^1 \rightarrow e_j^3 \rightarrow t_j \rightarrow_F s$. Again we pay $3 + \lambda$ for the forced edges, 4 for the simple edges inside the gadget and 3 for the outgoing edges incident on x^l, y^l, z^l. Total local edge cost: $c_T(V_j^{3A}) = \lambda + 10$.

Case $(x + y + z = 2)$**:**
Let us assume that $x = y = 1 \neq z$ holds. Then, we use $s \rightarrow_F s_j \rightarrow e_j^3 \rightarrow z^l \rightarrow_F z^r \rightarrow e_j^1 \rightarrow e_j^3 \rightarrow e_j^2 \rightarrow t_j \rightarrow_F s$ with total local edge cost $c_T(V_j^{3A}) = \lambda + 11$.

Case $(x + y + z = 0)$**:**
We use $s \rightarrow_F s_j \rightarrow e_j^2 \rightarrow y^l \rightarrow_F y^r \rightarrow e_j^3 \rightarrow z^l \rightarrow_F z^r \rightarrow e_j^1 \rightarrow x^l \rightarrow_F x^r \rightarrow e_j^2 \rightarrow t_j \rightarrow_F s$ with $c_T(V_j^{3A}) = \lambda + 11$.

The total edge cost of the quasi-tour we constructed is $3 \cdot 9m + (10 + 2\lambda)m + k = 37m + 2\lambda m + k$. We have at most $\nu + 1$ strongly connected components: one for each bi-wheel and one containing s. A component representing a bi-wheel can be connected to s as follows: let x^l, x^r be two contact vertices in the component. Add one copy of each edge from the cycle $s \rightarrow_F s_j \rightarrow e_j^1 \rightarrow x^l \rightarrow_F x^r \rightarrow e_j^2 \rightarrow t_j \rightarrow_F s$. This increases the cost by $5 + 2\lambda$ but decreases the number of components by one. ■

10.15.3 Tours in $\mathcal{G}_A$ to Assignments

In this section, we are going to prove the other direction of the reduction.

Lemma 10.15.2

If there is a tour in $\mathcal{G}_A$ with cost $37{\cdot}m+k+2\lambda{\cdot}m$, then, there is an assignment that leaves at most k equations unsatisfied.

Proof. Given a tour E_T in $\mathcal{G}_A$, we are going to define an assignment to checker and contact variables. As in Lemma 10.14.2, we will show that any tour must locally spend on each gadget at least the same amount as the tour we constructed in Lemma 10.15.1. If the tour spends more, we use that credit to satisfy possible unsatisfied equations.

Assignment for Checker Variables

Let us consider the following equations with two variables $x_i^u \oplus x_{i+1}^u = 0$, $x_{i-1}^u \oplus x_i^u = 0$, $x_i^u \oplus x_j^n = 1$, $x_j^n \oplus x_{j+1}^n = 0$, $x_{j-1}^n \oplus x_j^n = 0$ and the corresponding situation displayed in Figure 10.67 (b). Since E_T is a valid tour in $\mathcal{G}_A$, we know that $\{x_i^u, x_j^n\}_F$ is traversed and due to the degree condition, for each $x \in \{x_i^u, x_j^n\}$, the tour uses another incident edge e on x with $w(e) \geq 1$. Therefore, we have that $c_T(\{x_i^u, x_j^n\}) \geq 3$. The credit assigned to a gadget is defined as $cr_T(\{x_i^u, x_j^n\}) = c_T(\{x_i^u, x_j^n\}) - 3$.

Let us define the assignment for x_i^u and x_j^n. A variable x_i^u is honestly traversed if either both the simple edge going into x_i^u is used and the simple edge coming out of x_j^n is used, or neither of these two edges is used. In the first case, we set x_i^u to 1, otherwise to 0. Similarly, x_j^n is honest if both the edge going into x_j^n and the edge out of x_i^u are used, and we set it to 1 in the first case and 0 otherwise.

Honest tours:

First, suppose that both x_i^u and x_j^n are honest. We need to show that the credit is at least as high as the number of unsatisfied equations out of the five equations that contain them. It is not hard to see that if we have set $x_i^u \neq x_j^n$ all equations are satisfied. If we have set both to 1, then the forced edge must be used twice, making the local edge cost at least 6, giving a credit of 3, which is more than sufficient.

Dishonest tours:

If both x_i^u and x_j^n are dishonest the tour must be using the forced edge in both directions. Thus, the local cost is 5 or more, giving a credit of 2. There is always an assignment that satisfies three out of the five equations, so this case is done. If one of them is dishonest, the other must be set to 1 to ensure strong connectivity. Thus, there are two simple edges used leaving the gadget, making the local cost 4 (perhaps the same edge is used twice). We can set the honest variable to 1 (satisfying its two cycle equations), and the other to 0, leaving at most one equation unsatisfied.

Assignment for Contact Variables

First, we note that for any valid tour, we have $c_T(V_j^{3A}) \geq 10 + \lambda$. This is because the two forced edges of weight λ must be used, and there exist 10 vertices in the gadget for which all outgoing edges have weight 1. Let us define the credit $cr_T(V_j^{3A}) = c_T(V_j^{3A}) - (10 + \lambda)$.

Honest Traversals:

We assume that the underlying tour is honest, that is, forced edges are traversed only in one direction. We set x to 1 if the forced edge is used in the direction $x^r \to_F x^l$ and 0 otherwise. In the first case we know that the simple edges going into x^r and out of x^l are used. In the second, the edges $e_j^1 \to x^l$ and $x^r \to e_j^2$ are used. We do similarly for y, z.

We are interested in the equation $x \oplus y \oplus z = 1$ and the six cycle equations involving x, y, z. The assignment we pick for honest variables satisfies the cycle equations, so if it also satisfies the size-three equation we are done. If not, we have to prove that the tour pays at least $11 + \lambda$.

Case $(x = y = z = 0)$**:** Due to our assumption, we know that $e_j^2 \to y^l \to_F y^r \to e_j^3 \to z^l \to_F z^r \to e_j^1 \to x^l \to_F x^r \to e_j^2$ is a part of the tour. Since E_T is a tour, there exists a vertex in $V_j^{3A} \backslash \{s_j, t_j\}$ that is visited twice and we get $c_T(V_j^{3A}) \geq 11 + \lambda$. Thus, we can spend the credit $cr_T(V_j^{3A}) \geq 1$ on the unsatisfied equation $x \oplus y \oplus z = 1$.

Case $(x+y+z = 2)$**:** Without loss of generality, let us assume that $x = y = 1 \neq z$ holds. Then, we know that $e_j^3 \to z^l \to_F z^r \to e_j^1$ is a part of the tour. But, this implies that there is a vertex in $V(\mathcal{G}_j^{3A})$ that is visited twice. Hence, we have that $cr_T(V_j^{3A}) \geq 1$.

Dishonest Traversals:

Consider the situation, in which some forced edges $\{\gamma^r, \gamma^l\}_F$ are traversed in both directions for some variables $\gamma \in \{x, y, z\}$. For the honest variables, we set them to the appropriate value as before, and this satisfies their cycle equations. Observe now that if a forced edge $\gamma^l \to_F \gamma^r$ is also used in the opposite direction, then there must be another edge used to leave the set $\{\gamma^l, \gamma^r\}$. Thus the local edge cost of this set is at least 3. It follows that the credit we have for the gadget is at least as large as the number of dishonest variables. We can give appropriate values to them so each satisfies one cycle equation and the size-three equation is satisfied. Thus, the number of unsatisfied equations is not larger than our credit.

In summary, for every tour E_T in $\mathcal{G}_A$, we can find an assignment to the variables of $\mathscr{L}_2$ such that all unsatisfied equations are paid by the credit induced by E_T. ■

We are ready to give the proof of Theorem 10.6.7.

Proof of Theorem 10.6.7.
We are again given an instance $\mathscr{L}_1$ of the MAX-E3LIN2 problem with ν variables and m equations. For all $\delta > 0$, there exists a k such that if we repeat each equation k time we get an instance $\mathscr{L}_1^{(k)}$ with $m' = km$ equations and ν variables such that $\nu/m' \leq \delta$.

Then, from $\mathscr{L}_1^{(k)}$, we generate an instance $\mathscr{L}_2$ of the MAX-HYBRID-LIN2 problem and the corresponding directed graph $\mathcal{G}_A$. Due to Lemmata 10.15.1, 10.15.2 and Theorem 10.14.4, we know that for all $\varepsilon > 0$, it is **NP**-hard to tell whether there is a tour with cost at most $37m' + 5\nu + 2m(\nu + \lambda) + \varepsilon \cdot m' \leq 37 \cdot m' + \varepsilon' m'$ or all tours have cost at least $37m' + (0.5 - \varepsilon)m' \geq 37.5 \cdot m' - \varepsilon' \cdot m'$, for some ε' depending only on $\varepsilon, \delta, \lambda$. The ratio between these two cases can get arbitrarily close to 75/74 by appropriate choices for $\epsilon, \delta, \lambda$. ■

10.16 Bibliographic Notes

The presented material in this chapter is based on the papers [KS12], [KS13] and [KLS13]. In particular, the proofs of Theorem 10.6.1 and 10.6.2 appeared in [KS12]. The proofs of Theorem 10.6.3, 10.6.4 and 10.6.5 were given in [KS13]. The paper [KLS13] contains the proofs of Theorem 10.6.6 and 10.6.7.

CHAPTER 11

Conclusions and Further Research

In this thesis, we proved that it is hard to approximate the ATSP and the TSP problem within any constant factor less than 75/74 and 123/122, respectively. Since the best known upper bound on the approximability is $O(\log n/\log\log n)$ for the ATSP problem (cf. [AGM$^+$10]) and 3/2 for the TSP problem (cf. [C76]), there is certainly room for improvements. Especially, in the asymmetric version of the TSP problem, there is a large gap between the approximation lower and upper bound, and it remains a major open problem on the existence of an efficient approximation algorithm with constant approximation ratio for that problem.

We gave an improved inapproximability threshold for the (1,2)-STEINER TREE problem of 221/220. The best up to now known approximation upper bound is 5/4 (cf. [BKZ09]). Is the bound 5/4 the best possible under usual complexity theoretic assumptions? Furthermore, it would be nice to investigate if some of the ideas appeared in this thesis, and in particular the bi-wheel amplifiers, can be used to offer improved approximation hardness results for other optimization problems, such as the (1,2)-STEINER TREE and the general STEINER TREE problem.

We provided new explicit inapproximability bounds for general, cubic and subcubic instances of the (1,2)-TSP and GRAPHIC-TSP problem. The important question is to improve the explicit inapproximability bounds on those instances significantly. A bottleneck in our constructions, especially for the cubic case, are the parity gadgets. Using the modularity of the constructions, any improvement of the costs of the parity gadgets will lead to improved inapproximability bounds for the corresponding problems. The current best upper approximation bound for general cubic instances of the GRAPHIC-TSP problem is 4/3 (cf. [BSSS11a]). For the special case of 2-connected cubic graphs, the bound was recently improved to (4/3 - 1/61236) [CLS12]. How about further improving those bounds? How about improving the general upper bound of 8/7 [BK06] for cubic instances of the (1,2)-TSP problem?

An interesting question remains about even tighter lower approximation bounds for the VERTEX COVER problem restricted to dense and subdense k-partite k-hypergraphs, perhaps by resolving Conjecture 6.5.1 affirmatively.

We established an improved explicit approximation lower bound of 333/332 for the SHORTEST SUPERSTRING problem, even when we restrict the length of the given strings to be exactly 4. Recently, this restricted version of the problem was proved to be approximable within 8/5 (cf. [GKM13]). Is it possible to use bi-wheel amplifier methods to obtain improved hardness of approximation results for this problem as well?
Perhaps new PCP constructions or alternatively reductions from the UNIQUE GAMES problem are the natural next step for proving stronger approximation hardness results for the SHORTEST SUPERSTRING problem and some other problems that are related to the TSP problem considered in this thesis.

Bibliography

[AHK96] R. Aharoni, R. Holzman and M. Krivelevich, *On a Theorem of Lovász on Covers in r-Partite Hypergraphs*, Combinatorica 16, pp. 149–174, 1996.

[AK00] P. Alimonti and V. Kann, *Some APX-completeness Results for Cubic Graphs*, Theoretical Computer Science 237, pp. 123–134, 2000.

[AS95] C. Armen and C. Stein, *Improved Length Bounds for the Shortest Superstring Problem*, in Proc. 5th WADS (1995), LNCS 955, pp. 494–505, 1995.

[AS96] C. Armen and C. Stein, *A $2\frac{2}{3}$ Approximation Algorithm for the Shortest Superstring Problem*, in Proc. 7th CPM (1996), LNCS 1075, pp. 87–101, 1996.

[ABSS93] S. Arora, L. Babai, J. Stern and Z. Sweedyk, *The Hardness of Approximate Optima in Lattices, Codes, and Systems of Linear Equations*, in Proc. 34th FOCS (1993), pp. 724–733, 1993; also appeared in J. Comput. Syst. Sci. 54, pp. 317–331, 1997.

[AB09] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, 2009.

[AKK95] S. Arora, D. Karger and M. Karpinski, *Polynomial Time Approximation Schemes for Dense Instances of NP-Hard Problems*, Journal of Computer and System Sciences 58, pp. 193–210, 1999.

[ALM$^+$98] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy, *Proof Verification and the Hardness of Approximation Problems*, Journal of the ACM 45, pp. 501–555, 1998.

[AS98] S. Arora and S. Safra, *Probabilistic Checking of Proofs: A New Characterization of NP*, Journal of the ACM 45, pp. 70–122, 1998.

[AGM$^+$10] A. Asadpour, M. Goemans, A. Madry, S. Oveis Gharan and A. Saberi, *An $O(\log n/\log\log n)$-Approximation Algorithm for the Asymmetric Traveling Salesman Problem*, in Proc. 21st ACM-SIAM SODA (2010), pp. 379–389, 2010.

[ACG⁺99] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela und M. Protasi, *Complexity and Approximation*, Springer, 1999.

[B80] L. Babai, *On the Complexity of Canonical Labeling of Strongly Regular Graphs*, Siam Journal on Computing 9, 212–216, 1980.

[B92] P. Bachmann, *Analytische Zahlentheorie*, Teubner, 1892.

[BC11] R. Bailey and P. Cameron, *Base Size, Metric Dimension and Other Invariants of Groups and Graphs*, Bulletin of the London Mathematical Society 43, pp. 209–242, 2011.

[BGS02] A. Barvinok, E. Gimadi and A. Serdyukov, *The Maximum Traveling Salesman Problem*, in: *The Traveling Salesman Problem and Its Variations*, pp. 585–607, Kluwer, 2002.

[BK04] R. Bar-Yehuda and Z. Kehat, *Approximating the Dense Set-Cover Problem*, Journal of Computer and System Sciences 69, pp. 547–561, 2004.

[BEE⁺05] Z. Beerliova, F. Eberhard, T. Erlebach, A. Hall, M. Hoffmann, M. Mihalák and L. Ram, *Network Discovery and Verification*, IEEE Journal on Selected Areas in Communications 24, pp. 2168–2181, 2006.

[BGS98] M. Bellare, O. Goldreich and M. Sudan, *Free Bits, PCPs, and Nonapproximability-Towards Tight Results*, SIAM Journal on Computing 27, pp. 804–915, 1998.

[BK99] P. Berman and M. Karpinski, *On Some Tighter Inapproximability Results*, in Proc. 26th ICALP (1999), LNCS 1644, pp. 200 –209, 1999.

[BK01] P. Berman and M. Karpinski, *Efficient Amplifiers and Bounded Degree Optimization*, ECCC TR01-053, 2001.

[BK03] P. Berman and M. Karpinski, *Improved Approximation Lower Bounds on Small Occurrence Optimization*, ECCC TR03-008, 2003.

[BK06] P. Berman and M. Karpinski, *8/7-Approximation Algorithm for $(1,2)$-TSP*, in Proc. 17th ACM-SIAM SODA (2006), pp. 641–648, 2006.

[BKZ09] P. Berman, M. Karpinski and A. Zelikovsky, *1.25-Approximation Algorithm for Steiner Tree Problem with Distances 1 and 2*, in Proc. 11th WADS (2009), LNCS 5664, pp. 86–97, 2009.

[BR94] P. Berman and V. Ramaiyer, *Improved Approximations for the Steiner Tree Problem*, Journal of Algorithms 17, pp. 381–408, 1994.

[BP89] M. Bern and P. Plassmann, *The Steiner Problem with Edge Lengths 1 and 2*, Information Processing Letters 32, pp. 171–176, 1989.

[B02] M. Bläser, *An $\frac{8}{13}$-Approximation Algorithm for the Asymmetric Maximum TSP*, in Proc. 13th ACM-SIAM SODA (2002), 64–73, 2002; also appeared in J. of Algorithms 50, pp. 23–48, 2004.

[B04] M. Bläser, *A 3/4-Approximation Algorithm for Maximum ATSP with Weights Zero and One*, in Proc. 7th APPROX (2004), LNCS 3122, pp. 61–71, 2004.

[BJL$^+$94] A. Blum, T. Jiang, M. Li, J. Tromp and M. Yanakakis, *Linear Approximation of Shortest Superstrings*, Journal of the ACM 41, pp. 630–647, 1994.

[BS00] H.-J. Böckenhauer and S. Seibert, *Improved Lower Bounds on the Approximability of the Traveling Salesman Problem*, Informatique Théorique et Applications 34, pp. 213–255, 2000.

[BSSS11a] S. Boyd, R. Sitters, S. van der Ster and L. Stougie, *TSP on Cubic and Subcubic Graphs*, in Proc. 15th IPCO (2011), LNCS 6655, pp. 65–77, 2011.

[BSSS11b] S. Boyd, R. Sitters, S. van der Ster and L. Stougie, *TSP on Cubic and Subcubic Graphs*, CoRR arXiv: abs/1107.1052, 2011.

[BJJ97] D. Breslauer, T. Jiang, and Z. Jiang, *Rotations of Periodic Strings and Short Superstrings*, Journal of Algorithms 24, pp. 340–353, 1997.

[BGRS10] J. Byrka, F. Grandoni, T. Rothvoß and L. Sanità, *An Improved LP-Based Approximation for Steiner Tree*, in Proc. 42nd ACM STOC (2010), pp. 583–592, 2010.

[CHM+07] J. Cáceres, C. Hernando, M. Mora, I. Pelayo, M. Puertas, C. Seara and D. Wood, *On the Metric Dimension of Cartesian Products of Graphs*, SIAM Journal on Discrete Mathematics 21, 423–441, 2007.

[CHM+09] J. Cáceres, C. Hernando, M. Mora, I. Pelayo and M. Puertas, *On the Metric Dimension of Infinite Graphs*, Electronic Notes in Discrete Mathematics 35, pp. 15–20, 2009.

[CKSV11] J. Cardinal, M. Karpinski, R. Schmied and C. Viehmann, *Approximating Subdense Instances of Covering Problems*, Electronic Notes in Discrete Mathematics 37, pp. 297–302, 2011.

[CKSV12] J. Cardinal, M. Karpinski, R. Schmied and C. Viehmann, *Approximating Vertex Cover in Dense Hypergraphs*, Journal of Discrete Algorithms 13, pp. 67–77, 2012.

[CLL+05] J. Cardinal, M. Labbé, S. Langerman, E. Levy and H. Mélot, *A Tight Analysis of the Maximal Matching Heuristic*, in Proc. 11th COCOON 2005, LNCS 3595, pp. 701-709, 2005.

[CLL09] J. Cardinal, S. Langerman and E. Levy, *Improved Approximation Bounds for Edge Dominating Set in Dense Graphs*, Theoretical Computer Science 410, pp. 949–957, 2009.

[CL10] J. Cardinal and E. Levy, *Connected Vertex Covers in Dense Graphs*, Theoretical Computer Science 411, pp. 2581–2590, 2010.

[CGH08] G. Chappell, J. Gimbel and C. Hartman, *Bounds on the Metric and Partition Dimensions of a Graph*, Ars Combinatoria 88, 2008.

[CEJO00] G. Chartrand, L. Eroh, M. Johnson and O. Oellermann, *Resolvability in Graphs and the Metric Dimension of a Graph*, Discrete Applied Mathematics 105, pp. 99–113, 2000.

[CC06] M. Chlebík and J. Chlebíková, *Complexity of Approximating Bounded Variants of Optimization Problems*, Theoretical Computer Science 354, pp. 320–338, 2006.

[CC08] M. Chlebík and J. Chlebíková, *The Steiner Tree Problem on Graphs: Inapproximability Results*, Theoretical Computer Science 406, pp. 207–214, 2008.

[C76] N. Christofides, *Worst-Case Analysis of a New Heuristic for the Traveling Salesman Problem*, Technical Report CS-93-13, Carnegie Mellon University, Pittsburgh, 1976.

[C83] V. Chvátal, *Mastermind*, Combinatorica 3, pp. 325–329, 1983.

[CT99] A. Clementi and L. Trevisan, *Improved Non-Approximability Results for Minimum Vertex Cover with Density Constraints*, Theoretical Computer Science 225, pp. 113–128, 1999.

[C64] A. Cobham, *The Intrinsic Computational Difficulty of Functions*, In Proc. 2nd International Congress for Logic, Methodology, and the Philosophy of Science (1964), pp. 24–30, 1964.

[C71] S. Cook, *The Complexity of Theorem Proving Procedures*, in Proc. 3rd ACM STOC (1971), pp. 151–158, 1971.

[CLS12] J. Correa, O. Larré and J. Soto, *TSP Tours in Cubic Graphs: Beyond* 4/3, in Proc. 20th ESA (2012), LNCS 7501, pp. 790–801, 2012.

[CKK02] B. Csaba, M. Karpinski and P. Krysta, *Approximability of Dense and Sparse Instances of Minimum 2-Connectivity, TSP and Path Problems*, in Proc. 13th ACM-SIAM SODA (2002), pp. 74–83, 2002.

[CGPR97] A. Czumaj, L. Gasieniec, M. Piotrow and W. Rytter, *Sequential and Parallel Approximation of Shortest Superstrings*, Journal of Algorithms 23 , pp. 74–100, 1997.

[DPSL11] J. Díaz, O. Pottonen, M. Serna and E. van Leeuwen, *On the Complexity of Metric Dimension*, in Proc. 20th ESA (2012), LNCS 7501, pp. 419–430, 2012.

[DGK02] I. Dinur, V. Guruswami and S. Khot *Vertex Cover on k-Uniform Hypergraphs is Hard to Approximate within Factor* $(k - 3 - \varepsilon)$, ECCC TR02-027, 2002.

[DGKR05] I. Dinur, V. Guruswami, S. Khot and O. Regev, *A New Multilayered PCP and the Hardness of Hypergraph Vertex Cover*, SIAM Journal on Computing 34, pp. 1129–1146, 2005.

[DS05] I. Dinur and S. Safra, *On the Hardness of Approximating Minimum Vertex Cover*, Annals of Mathematics 162, pp. 439–485, 2005.

[E65] J. Edmonds, *Paths, Trees and Flowers*, Canad. J. Math. 17, pp. 449–467, 1965.

[E03] L. Engebretsen, *An Explicit Lower Bound for TSP with Distances One and Two*, Algorithmica 35, pp. 301–318, 2003.

[EK06] L. Engebretsen and M. Karpinski, *TSP with Bounded Metrics*, Journal of Computer and System Sciences 72, pp. 509–546, 2006.

[E99] A. Eremeev, *On some Approximation Algorithms for Dense Vertex Cover Problem*, in Proc. SOR (1999), Operations Research Proceedings Series, pp. 48-52, Springer, 2000.

[FMO$^+$03] T. Feder, R. Motwani, L. O'Callaghan, R. Panigrahy and D. Thomas, *Online Distributed Predicate Evaluation*, Technical Report 2003-81, Stanford University, 2003.

[FGO06] M. Fehr, S. Gosselin and O. Oellermann, *The Metric Dimension of Cayley digraphs*, Discrete Mathematics 306, pp. 31–41, 2006.

[F03] U. Feige, *Vertex Cover is Hardest to Approximate on Regular Graphs*, Technical Report MCS03-15, Weizmann Institute, 2003.

[FGL$^+$96] U. Feige, S. Goldwasser, L. Lovasz, S. Safra and M. Szegedy, *Interactive Proofs and the Hardness of Approximating Cliques*, Journal of the ACM 43, pp. 268–292, 1996.

[FNW79] M. Fisher, G. Nemhauser and L. Wolsey, *An Analysis of Approximations for Finding a Maximum Weight Hamiltonian Circuit*, Operations Research 27, pp. 799–809, 1979.

[GMS80] J. Gallant, D. Maier and J. Storer, *On Finding Minimal Length Superstrings*, Journal of Computer and System Sciences 20, pp. 50–58, 1980.

[GLS05] D. Gamarnik, M. Lewenstein and M. Sviridenko, *An Improved Upper Bound for the TSP in Cubic 3-Edge-Connected Graphs*, Oper. Res. Lett. 33, pp. 467–474, 2005.

[GJ79] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman and Company, 1979.

[GJT76] M. Garey, D. Johnson and R. Tarjan, *The Planar Hamiltonian Circuit Problem is NP-Complete*, SIAM Journal of Computing 5, 704–714, 1976.

[G77] J. Gill, *Computational Complexity of Probabilistic Turing Machines*, SIAM Journal on Computing 6, pp. 675–695, 1977.

[G11] O. Goldreich, *Using the FGLSS-reduction to Prove Inapproximability Results for Minimum Vertex Cover in Hypergraphs*, ECCC TR01-102, 2001; also appeared in Studies in Complexity and Cryptography, LNCS 6650, pp. 88–97, 2011.

[GKM13] A. Golovnev, A. Kulikov and I. Mihajlin, *Approximating Shortest Superstring Problem Using de Bruijn Graphs*, in Proc. 24th CPM (2013), LNCS 7922, pp. 120–129, 2013.

[GLR08] Z. Gotthilf, M. Lewenstein, and E. Rainshmidt, *A $(2 - c\frac{\log n}{n})$ Approximation Algorithm for the Minimum Maximal Matching Problem*, Proc. 6th WAOA (2008), LNCS 5426, pp. 267–278, 2009.

[GS10a] G. Gottlob and P. Senellart, *Schema Mapping Discovery from Data Instances*, Journal of the ACM 57, Article No. 6, 2010.

[GHS02] V. Guruswami, J. Håstad and M. Sudan, *Hardness of Approximate Hypergraph Coloring*, SIAM Journal on Computing 31, pp. 1663–1686, 2002.

[GS10b] V. Guruswami and R. Saket, *On the Inapproximability of Vertex Cover on k-Partite k-Uniform Hypergraphs*, in Proc. 37th ICALP 2010, LNCS 6198, pp. 360–371, 2010; also appeared in ECCC TR13-071, 2013.

[H02] E. Halperin, *Improved Approximation Algorithms for the Vertex Cover Problem in Graphs and Hypergraphs*, SIAM Journal on Computing 31, pp. 1608–1623, 2002.

[HM76] F. Harary and R. Melter, *On the Metric Dimension of a Graph*, Ars Combinatoria 2, pp. 191–195, 1976.

[HS65] J. Hartmanis and R. Stearns, *On the Computational Complexity of Algorithms*, Transactions of the American Mathematical Society 117, pp. 285–306, 1965.

[HN12] S. Hartung and A. Nichterlein, *On the Parametrized and Approximation Hardness of Metric Dimension*, CoRR arXiv:abs/1211.1636, 2012.

[H01] J. Håstad, *Some Optimal Inapproximability Results*, Journal of the ACM 48, pp. 798–859, 2001.

[H07] M. Hauptmann, *Approximation Hardness of the (1,2)-Steiner Tree Problem*, CS-Report 85283, University of Bonn, 2007.

[HSV12] M. Hauptmann, R. Schmied and C. Viehmann, *Approximation Complexity of Metric Dimension Problem*, in Proc. 21st IWOCA (2010), LNCS 6460, pp. 136–139, 2011; also appeared in Journal of Discrete Algorithms 14, pp. 214–222, 2012.

[HMP+10] C. Hernando, M. Mora, I. Pelayo, C. Seara, and D. Wood, *Extremal Graph Theory for Metric Dimension and Diameter*, The Electronic Journal of Combinatorics 17, 2010.

[H02a] J. Holmerin, *Vertex Cover on* 4*-Regular Hyper-Graphs is Hard to Approximate within* $2 - \varepsilon$, in Proc. 34th ACM STOC (2002), pp. 544–552, 2002.

[H02b] J. Holmerin, *Improved Inapproximability Results for Vertex Cover on k-Uniform Hypergraphs*, in Proc. 29th ICALP (2002), LNCS 2380, pp. 1005–1016, 2002.

[HP99] S. Hougardy and H. Prömel, *A 1.598 Approximation Algorithm for the Steiner Problem in Graphs*, in Proc. 10th ACM-SIAM SODA (1999), pp. 448–453, 1999.

[HRW92] F. Hwang, D. Richards and P. Winter, *The Steiner Tree problem*, North-Holland, 1992.

[ISY05] L. Ilie, R. Solis-Oba and S. Yu, *Reducing the Size of NFAs by Using Equivalences and Preorders*, in Proc. 16th CPM (2005), LNCS 3537, pp. 310–321, 2005.

[II05] T. Imamura and K. Iwama, *Approximating Vertex Cover on Dense Graphs*, in Proc. 16th ACM-SIAM SODA (2005), pp. 582-589, 2005.

[KLSS05] H. Kaplan, M. Lewenstein, N. Shafrir and M. Sviridenko, *Approximation Algorithms for Asymmetric TSP by Decomposing Directed Regular Multigraphs*, Journal of the ACM 52, pp. 602–626, 2005.

[K09] G. Karakostas, *A Better Approximation Ratio for the Vertex Cover Problem*, ACM Transactions on Algorithms 5, 2009.

[K75] R. Karp, *On the Computational Complexity of Combinatorial Problems*, Networks 5, pp 45–68, 1975.

[K01] M. Karpinski, *Polynomial Time Approximation Schemes for Some Dense Instances of NP-Hard Optimization Problems*, Algorithmica 30, pp. 386–397, 2001.

[KLS13] M. Karpinski, M. Lampis and R. Schmied, *New Inapproximability Bounds for TSP*, CoRR arXiv: abs/1303.6437, 2013.

[KS11] M. Karpinski and R. Schmied, *Improved Lower Bounds for the Shortest Superstring and Related Problems*, CoRR arXiv: abs/1111.5442, 2011; also appeared in Proc. 19th CATS (2013), CRPIT 141, pp. 27-36, 2013; submitted to Theoretical Computer Science.

[KS12] M. Karpinski and R. Schmied, *On Approximation Lower Bounds for TSP with Bounded Metrics*, CoRR arXiv: abs/1201.5821, 2012; also appeared in Proc. 19th CATS (2013), CRPIT 141, pp. 27-36, 2013; submitted to Theoretical Computer Science.

[KS13] M. Karpinski and R. Schmied, *Approximation Hardness of Graphic TSP on Cubic Graphs*, CoRR arXiv: abs/1304.6800, 2013.

[KSV11] M. Karpinski, R. Schmied and C. Viehmann, *Tight Approximation Bounds for Vertex Cover on Dense k-Partite Hypergraphs*, CoRR arXiv: abs/1107.2000, 2011; submitted to Journal of Discrete Algorithms.

[KZ97a] M. Karpinski and A. Zelikovsky, *New Approximation Algorithms for the Steiner Tree Problems* Journal of Combinatorial Optimization 1, pp. 47–65, 1997.

[KZ97b] M. Karpinski and A. Zelikovsky, *Approximating Dense Cases of Covering Problems*, ECCC TR97-004, 1997; also appeared in Proc. DIMACS Workshop on Network Design: Connectivity and Facilities Location (1997), pp. 169–178, 1997.

[K02] S. Khot, *On the Power of Unique 2-Prover 1-Round Games*, in Proc. 34th ACM STOC (2002), pp. 767–775, 2002.

[KKMO07] S. Khot, G. Kindler, E. Mossel and R. O'Donnell, *Optimal Inapproximability Results for MAX-CUT and Other 2-Variable CSPs?*, SIAM Journal on Computing 37, pp. 319–357, 2007.

[KR08] S. Khot and O. Regev, *Vertex Cover might be Hard to Approximate to within 2-ϵ*, Journal of Computer and System Sciences 74, pp. 335–349, 2008.

[KRR96] S. Khuller, B. Raghavachari and A. Rosenfeld, *Landmarks in Graphs*, Discrete Applied Mathematics 70, pp. 217–229, 1996.

[KPS94] R. Kosaraju, J. Park and C. Stein, *Long Tours and Short Superstrings*, in Proc. 35th FOCS (1994), pp. 166–177, 1994.

[KMTV11] A. Kumar, R. Manokaran, M. Tulsiani, and N. Vishnoi, *On LP-Based Approximability for Strict CSPs*, in Proc. 22nd ACM-SIAM SODA (2011), pp. 1560–1573, 2011.

[L12] M. Lampis, *Improved Inapproximability for TSP*, in Proc. 15th APPROX (2012), LNCS 7408, pp. 243–253, 2012.

[L09] E. Landau, *Handbuch der Lehre von der Verteilung der Primzahlen*, B. G. Teubner, 1909.

[LMSS55] K. de Leeuw, E. Moore, C. Shannon and N. Shapiro, *Computability by Probabilistic Machines*, in Automata Studies, pp. 183–212, Princeton Unviersity Press, 1955.

[L88] A. Lesk, *Computational Molecular Biology: Sources and Methods for Sequence Analysis*, Oxford University Press, 1988.

[L73] L. Levin, *Universal Sequential Search Problems*, Problems of Information Transmission 9, pp. 265–266, 1973; translated from Problemy Peredachi Informatskii 9, pp. 115–116, 1973.

[LS03] M. Lewenstein and M. Sviridenko, *Approximating Asymmetric Maximum TSP*, in Proc. 14th ACM-SIAM SODA (2003), pp. 646–654, 2003.

[L90] M. Li, *Towards a DNA Sequencing Theory (Learning a String)*, in Proc. 31st FOCS (1990), pp. 125–134, 1990.

[L75] L. Lovász, *On Minimax Theorems of Combinatorics*, Doctoral Thesis, Mathematiki Lapok 26, pp. 209–264, 1975.

[MS77] D. Maier and J. Storer, *A Note on the Complexity of the Superstring Problem*, Report No. 223, Princeton University, 1977.

[M05] B. Manthey, *Approximability of Cycle Covers and Smoothed Analysis of Binary Search Trees*, Doctoral thesis, University of Lübeck, 2005.

[MJ75] A. Mayne and E. James, *Information Compression by Factorising Common Superstrings*, The Computer Journal 18, pp. 157–160, 1975.

[M94] M. Middendorf, *More on the Complexity of Common Superstring and Supersequence Problems*, Theoretical Computer Science 125, pp. 205–228, 1994.

[M98] M. Middendorf, *Shortest Common Superstrings and Scheduling with Coordinated Starting Times*, Theoretical Computer Science 191, pp. 205–214, 1998.

[MU05] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*, Cambridge University Press, 2005.

[MS11] T. Mömke and O. Svensson, *Approximating Graphic TSP by Matchings*, in Proc. IEEE 52nd FOCS (2011), pp. 560–569, 2011.

[M12] M. Mucha, 13/9-*Approximation for Graphic TSP*, in Proc. 29th STACS (2012), Leibniz International Proceedings in Informatics 14, pp. 30–41, 2012.

[M13] M. Mucha, *Lyndon Words and Short Superstrings*, in Proc. 24th ACM-SIAM SODA (2013), pp. 958–972, 2013.

[NI99] H. Nagamochi and T. Ibaraki, *An Approximation of the Minimum Vertex Cover in a Graph*, Japan Journal of Industrial and Applied Mathematics 16, pp. 369–375, 1999.

[O99] S. Ott, *Lower Bounds for Approximating Shortest Superstrings over an Alphabet of Size 2*, in Proc. 25th WG (1999), LNCS 1665, pp. 55–64, 1999.

[OSS11] S. Oveis Gharan, A. Saberi and M. Singh, *A Randomized Rounding Approach to the Traveling Salesman Problem*, in Proc. IEEE 52nd FOCS (2011), pp. 550–559, 2011.

[PEZ12] K. Paluch, K. Elbassioni and A. van Zuylen, *Simpler Approximation of the Maximum Asymmetric Traveling Salesman Problem*, in Proc. 29th STACS (2012), Leibniz International Proceedings in Informatics 14, pp. 501–506, 2012.

[PV06] C. Papadimitriou and S. Vempala, *On the Approximability of the Traveling Salesman Problem*, Combinatorica 26, pp. 101–120, 2006.

[PY91] C. Papadimitriou and M. Yannakakis, *Optimization, Approximation, and Complexity Classes*, Journal of Computer and System Sciences 43, pp. 425–440, 1991.

[PY93] C. Papadimitriou and M. Yannakakis, *The Traveling Salesman Problem with Distances One and Two*, Mathematics of Operations Research 18, pp. 1–11, 1993.

[R98] R. Raz, *A Parallel Repetition Theorem*, SIAM Journal on Computing 27, pp. 763–803, 1998.

[RZ00] G. Robins and A. Zelikovsky, *Tighter Bounds for Graph Steiner Tree Approximation*, SIAM Journal on Discrete Mathematics 19, pp. 122–134, 2005.

[SS11] S. Sachdeva and R. Saket, *Nearly Optimal NP-Hardness of Vertex Cover on k-Uniform k-Partite Hypergraphs*, in Proc. 14th APPROX

(2011), LNCS 6845, pp. 327–338, 2011; also appeared in ECCC TR13-071, 2013.

[SV11] R. Schmied and C. Viehmann, *Approximating Edge Dominating Set in Dense Graphs*, in Proc. 8th TAMC (2011) LNCS 6648, 37–47, 2011; also appeared in Theoretical Computer Science 414, pp. 92–99, 2012.

[ST04] A. Sebö and E. Tannier, *On Metric Generators of Graphs*, Mathematics of Operations Research 29, pp. 383–393, 2004.

[SV12] A. Sebö and J. Vygen, *Shorter Tours by Nicer Ears*, CoRR arXiv: abs/1201.1870, 2012; to appear in Combinatorica.

[SS63] H. Shapiro and S. Söderberg, *A Combinatory Detection Problem*, The American Mathematical Monthly 70, pp. 1066–1070, 1963.

[S75] P. Slater, *Leaves of Trees*, Congressus Numerantium 14, pp. 549-559, 1975.

[S88a] P. Slater, *Dominating and Reference Sets in Graphs*, Journal of Mathematical and Physical Sciences 22, pp. 445–455, 1988.

[S88b] J. Storer, *Data Compression: Methods and Theory*, Computer Science Press, 1988.

[SS82] J. Storer and T. Szymanski, *Data Compression via Textual Substitution*, Journal of the ACM 29, pp. 928–951, 1982.

[S99] Z. Sweedyk, *A $2\frac{1}{2}$-Approximation Algorithm for Shortest Superstring*, SIAM Journal on Computing 29, pp. 954–986, 1999.

[TU88] J. Tarhio and E. Ukkonen, *A Greedy Approximation Algorithm for Constructing Shortest Common Superstrings*, Theoretical Computer Science 57, pp. 131–145, 1988.

[TT93] S. Teng and F. Yao, *Approximating Shortest Superstrings*, SIAM Journal on Computing 26, pp. 410–417, 1997.

[T03] M. Thimm, *On the Approximability of the Steiner Tree Problem*, Theoretical Computer Science 295, pp. 387–402 , 2003.

[T90] V. Timkovskii, *Complexity of Common Subsequence and Supersequence Problems and Related Problems*, Cybernetics and Systems Analysis 25 (1990), pp. 565–580; translated from Kibernetika 25 (1989), pp. 1–13.

[T08] I. Tomescu, *Discrepancies between Metric Dimension and Partition Dimension of a Connected Graph*, Discrete Mathematics 308, pp. 5026–5031, 2008.

[T00] L. Trevisan, *When Hamming Meets Euclid: The Approximability of Geometric TSP and Steiner Tree*, SIAM Journal on Computing 30, pp. 475–485, 2000.

[T01] L. Trevisan, *Non-Approximability Results for Optimization Problems on Bounded Degree Instances*, in Proc. 33rd ACM STOC (2001), pp. 453–461, 2001.

[T36] A. Turing, *On Computable Numbers, with an Application to the Entscheidungsproblem*, in Proc. London Mathematical Society 1936, pp. 230–265.

[T89] J. Turner, *Approximation Algorithms for the Shortest Common Superstring Problem*, Information and Computation 83, pp. 1–20, 1989.

[V05] V. Vassilevska, *Explicit Inapproximability Bounds for the Shortest Superstring Problem*, in Proc. 30th MFCS (2005), LNCS 3618, pp. 793–800, 2005.

[V92] S. Vishwanathan, *An Approximation Algorithm for the Asymmetric Travelling Salesman Problem with Distances One and Two*, Information Processing Letters 44, pp. 297–302, 1992.

[Z93] A. Zelikovsky, *11/6-Approximation Algorithm for the Network Steiner Problem*, Algorithmica 9, pp. 463–470, 1993.

List of Tables

List of Figures

Index

Publikationen

1. J. Cardinal, M. Karpinski, R. Schmied and C. Viehmann, *Approximating Subdense Instances of Covering Problems*, Electronic Notes in Discrete Mathematics 37, pp. 297–302, 2011.

2. J. Cardinal, M. Karpinski, R. Schmied and C. Viehmann, *Approximating Vertex Cover in Dense Hypergraphs*, Journal of Discrete Algorithms 13, pp. 67–77, 2012.

3. M. Hauptmann, S. Kühl, R. Schmied and C. Viehmann *Polynomial Approximation Schemes for Dense and Geometric k-Restricted Steiner Forest Problems*, CS-Report 85288, University of Bonn, 2008.

4. M. Hauptmann, R. Schmied and C. Viehmann, *Approximation Complexity of Metric Dimension Problem*, in Proc. 21st IWOCA (2010), LNCS 6460, pp. 136–139, 2011; also appeared in Journal of Discrete Algorithms 14, pp. 214–222, 2012.

5. M. Karpinski, M. Lampis and R. Schmied, *New Inapproximability Bounds for TSP*, CoRR arXiv: abs/1303.6437, 2013.

6. M. Karpinski and R. Schmied, *Improved Lower Bounds for the Shortest Superstring and Related Problems*, CoRR arXiv: abs/1111.5442, 2011; also appeared in Proc. 19th CATS (2013), CRPIT 141, pp. 27–36, 2013; submitted to Theoretical Computer Science.

7. M. Karpinski and R. Schmied, *On Approximation Lower Bounds for TSP with Bounded Metrics*, CoRR arXiv: abs/1201.5821, 2012; also appeared in Proc. 19th CATS (2013), CRPIT 141, pp. 27–36, 2013; submitted to Theoretical Computer Science.

8. M. Karpinski and R. Schmied, *Approximation Hardness of Graphic TSP on Cubic Graphs*, CoRR arXiv: abs/1304.6800, 2013.

9. M. Karpinski, R. Schmied and C. Viehmann, *Tight Approximation Bounds for Vertex Cover on Dense k-Partite Hypergraphs*, CoRR

arXiv: abs/1107.2000, 2011; submitted to Journal of Discrete Algorithms.

10. R. Schmied and C. Viehmann, *On Approximation Complexity of Edge Dominating Set Problem in Dense Graphs*, in Proc. 7th JCCGG (2009), pp. 135–136, 2009.

11. R. Schmied and C. Viehmann, *Approximating Edge Dominating Set in Dense Graphs*, in Proc. 8th TAMC (2011), LNCS 6648, 37–47, 2011; also appeared in Theoretical Computer Science 414, pp. 92–99, 2012.