UNIVERSITÄT BONN
Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik der Universität Bonn

Beyond Worst-Case Analysis of Max-Cut and Local Search

Michael Etscheid

geboren in Trier

Dissertation

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

Bonn, 2018

1. Gutachter: Prof. Dr. Heiko Röglin
2. Gutachter: Prof. Dr. Stefan Kratsch

Tag der mündlichen Prüfung: 22. Juni 2018
Erscheinungsjahr: 2018

Title page image: Frank Luerweg / Universität Bonn, photography.

Abstract

Since the formal definition of NP-completeness, there has been a huge discrepancy between theory and practice. The theoretical point of view for an NP-complete problem is that we do not know of any efficient, that is, polynomial-time algorithm for this problem. On the other hand, many of these problems are very well tractable in a practical sense. For any such problem there is a family of instances on which every known algorithm takes exponential time and is therefore not suitable for larger instances. But these instance families are often very contrived and artificial. The real-world instances, which we would actually like to solve, often tend to be much easier than what the worst-case running times may suggest. This motivates the research in other settings than the worst-case scenario. We will focus in this work on two popular analysis techniques: smoothed analysis and kernelization.

In a smoothed analysis a small amount of random noise is added to the instances before analyzing the expected running time of an algorithm. We use this framework to analyze local search, a technique that is perhaps the most famous example for which worst-case analysis fails to make reasonable running time predictions. Despite its exponential worst-case running time, we show that local search for the Maximum-Cut problem terminates after a quasi-polynomial number of steps in the smoothed setting. If we consider instances in which the nodes are points in a d-dimensional space and the edge weights are given by the squared Euclidean distances between these points, the smoothed running time is even polynomial in n and 2^d.

Furthermore, we analyze local search for a standard scheduling problem in which jobs with different processing requirements are assigned to related machines. Local search corresponds to the best response dynamics that arises when jobs selfishly try to minimize their costs. We assume that each machine runs a coordination mechanism that determines the order of execution of jobs assigned to it. We obtain various new polynomial and pseudo-polynomial bounds for the convergence time of local search with respect to the coordination mechanisms Makespan, FIFO, and Shortest-Job-First. The pseudo-polynomial bounds can almost all be translated to smoothed polynomial running times.

In the second part of this thesis we will focus on kernelization. This technique is a formalization of efficient preprocessing for NP-hard problems using the framework of parameterized complexity. The first application is again the Maximum-Cut problem and some generalizations of it. Several of these cut problems are known to admit polynomial kernels when parameterized above the tight lower bound measured by the size and order of the graph. We continue this line of research and show how to find kernels with only $O(k)$ vertices in $O(km)$ time, where k is our parameter.

Finally we use an algorithm for compressing numbers due to Frank and Tardos (Cominatorica 1987) to derive polynomial kernels for weighted versions of several well-studied parameterized problems like d-Hitting Set and d-Set Packing. It is also useful to obtain kernels for various formulations of Subset Sum and Knapsack as well as for polynomial integer programs.

This research was supported by ERC Starting Grant 306465 (BeyondWorstCase).

Acknowledgments

First of all I would like to thank my supervisor Heiko Röglin for his endless support. Due to his intuitive understanding especially of smoothed analysis, he always knew to ask the right questions and guide my work. It definitely saved me a lot of time and frustration when he recognized at a glance not only once that an approach would not work. He always made time for me, no matter how much else he had to do. I also thank Stefan Kratsch for his effort as co-referee of this thesis and co-author of one of my papers.

I wish to thank my two other co-authors Tobias Brunsch and Matthias Mnich. They showed me interesting problems to work on and I had fruitful discussions with them. Tobias also introduced me to the work as a PhD student and taught me valuable lessons about work as well as life.

I am grateful to be part of the theory group in Bonn. It was a great time with you. Special thanks go to Carsten Fischer for trying to teach me basics in probability theory and Clemens Rösner for always having an open ear and helpful suggestions for my daily struggles with computer science, English, and LaTeX.

Most importantly, I thank Lisa. You have been supporting me during the last eleven years in a way that I could not imagine before. Thank you!

Contents

1 Introduction **7**
1.1 Smoothed Analysis . . . 8
1.2 Kernelization . . . 10
1.3 Outline and Bibliographical Notes . . . 11

2 Results **13**
2.1 Smoothed Analysis of Local Search for Max-Cut . . . 13
2.1.1 General Graphs . . . 13
2.1.2 Squared Euclidean Distances . . . 14
2.1.3 Max-Cut in the Context of PLS . . . 15
2.1.4 Related Work . . . 15
2.2 Scheduling . . . 16
2.2.1 Terminology . . . 17
2.2.2 Smoothed Analysis . . . 18
2.2.3 Related Work and Results . . . 18
2.3 Signed Max-Cut Above Edwards-Erdős Bound and Linear Vertex Kernels for λ-Extendible Properties . . . 21
2.3.1 Our Contributions . . . 22
2.4 Polynomial Kernels for Weighted Problems . . . 24
2.4.1 Our Contributions . . . 25

3 Max-Cut on General Graphs **27**
3.1 Outline of the Analysis . . . 27
3.2 Analysis . . . 28

4 Max-Cut with Squared Euclidean Distances **34**
4.1 Outline of the Analysis . . . 34
4.2 Preliminaries and Notation . . . 35
4.3 Bounding the Smoothed Number of Steps . . . 37
4.3.1 Approximation for the Improvement of a Flip . . . 38
4.3.2 Bounding the Probability of Insignificant Improvements . . . 40
4.3.3 Calculating the Expected Number of Steps . . . 42
4.4 An Exponential Lower Bound in the Deterministic Setting . . . 43

5 Scheduling **45**
5.1 Smoothed Analysis . . . 45
5.2 Identical Machines . . . 46
5.2.1 FIFO and Makespan Model . . . 46
5.2.2 SJF Model . . . 47
5.3 Unit-Weight Jobs . . . 52
5.4 Related Machines . . . 52
5.4.1 FIFO Model . . . 53
5.4.2 Makespan Model . . . 56
5.4.3 SJF Model . . . 59
5.5 Price of Anarchy for FIFO . . . 59

6 Signed Max-Cut Above Edwards-Erdős Bound **61**
6.1 Preliminaries . . . 61
6.2 A Linear-Time Fixed-Parameter Algorithm for Signed Max-Cut AEE . . . 62
6.3 A Linear Vertex Kernel for Signed Max-Cut AEE . . . 68
6.3.1 Kernelization Rules . . . 69
6.3.2 Bounding the Kernel Size . . . 72

7 Linear Vertex Kernels for λ-Extendible Properties **82**
7.1 Linear Kernel for Properties Diverging on Cliques . . . 82
7.2 Strongly $\frac{1}{2}$-Extendible Properties on Oriented Graphs . . . 84

8 Polynomial Kernel for Weighted Problems **92**
8.1 Settling Open Problems via the Frank-Tardos Theorem . . . 92
8.1.1 Frank and Tardos' theorem . . . 92
8.1.2 Polynomial Kernelization for Knapsack . . . 92
8.2 Small Kernels for Weighted Parameterized Problems . . . 93
8.2.1 Hitting Set and Set Packing . . . 93
8.2.2 Max-Cut . . . 94
8.2.3 Bin Packing with Additive Error . . . 95
8.3 Kernel Bounds for Knapsack Problems . . . 96
8.3.1 Exponential Kernel for Knapsack with Few Item Sizes . . . 96
8.3.2 Polynomial Kernel for Subset Sum with Few Item Sizes . . . 97
8.3.3 A Kernelization Lower Bound for Subset Sum . . . 98
8.4 Integer Polynomial Programming with Bounded Range . . . 99

9 Conclusions and Open Problems **101**
9.1 Smoothed Analysis of Local Search for Max-Cut . . . 101
9.2 Scheduling . . . 102
9.3 Max-Cut Above Edwards-Erdős Bound . . . 102
9.4 Polynomial Kernels for Weighted Problems . . . 102

1 Introduction

Worst-case analysis has been the predominant analysis concept in theoretical computer science since the beginning of this research area. If the worst-case running time of an algorithm is small, we can expect it to run fast in practice. The introduction of NP-completeness has been the most remarkable breakthrough to show that many optimization problems do not admit such an algorithm with polynomial worst-case running time and therefore not every instance can be solved efficiently unless $\mathsf{P} = \mathsf{NP}$. Fortunately, the vast majority of real-world instances for NP-complete problems are much better tractable than the worst case might suggest. As a consequence, we can solve considerably larger instances for many problems than we would be able to by simple brute-force attempts. On the other hand, this means that we need different analysis techniques if we want to predict real-world tractability by theoretical means. In this thesis we will focus on two different popular analysis techniques: smoothed analysis, which is especially successful at explaining running times for local search, and kernelization.

Let us first motivate smoothed analysis for local search. The most successful algorithms for many well-studied optimization problems are based on the principle of local search: Start with an arbitrary feasible solution and perform some kind of local improvements until none is possible anymore. For many important problems like the traveling salesman problem, clustering, and linear programming, local search is the method of choice in practice. Its success, however, lacks a theoretical account. The main reason for the considerable gap between experimental results and our theoretical understanding is that for most problems worst-case analysis predicts that local search is inefficient and can result in very bad solutions. It is not taken into account that worst-case instances are often rather contrived and rarely observed in practice. This indicates that the predominant theoretical measure of worst-case analysis does not suffice to evaluate local search. It suggests to apply more realistic performance measures that help to advance our understanding of this simple, yet powerful algorithmic technique.

To narrow the gap between the worst-case results and experimental findings, we analyze local search algorithms for Max-Cut and some scheduling problems in the framework of smoothed analysis, which has been invented by Spielman and Teng [100] to explain the practical success of the simplex method. In a smoothed analysis, first the adversary creates a bad instance like in a worst-case analysis. Then a small amount of random noise is added to this instance. This weakens of course the adversary. The idea however is that the expected running time of an algorithm on the perturbed instance gets smaller and smaller the more artificial and fragile a worst-case instance is. That means that the smoothed running time of an algorithm should be much nearer to the actual running time in practice than the worst-case is, as long as we do not test our algorithm in specially constructed benchmarks. This approach is by now a well-established alternative to worst-case analysis. It is also considered a very natural approach, as adding random noise to the instance can often be motivated by, e.g., measurement errors, numerical imprecision or rounding errors. It can also model parts of the input that cannot be quantified exactly, but for which there is no reason to believe that they are adversarial. Therefore, the input model of smoothed analysis seems to be well-suited for a vast number of application areas.

The second important technique we use is the concept of kernelization in the framework of parameterized complexity. Kernelization is a formalization of efficient preprocessing for NP-hard problems. Up to now we do not know of any algorithm for an NP-hard problem that runs faster than exponential in the instance size and the widely-believed Exponential Time Hypothesis even rules out such an algorithm. Instance size alone, however, is often not a good indicator for the inherent difficulty of instances of a given NP-hard problem. For

example, the Subset Sum problem, i.e., the problem whether there is a non-empty subset of a set of integers whose sum is zero, becomes easier if most items have the same size, i.e., if the number k of different item sizes is small. This is because we can simply guess how many items of each size we need for a feasible solution. Therefore, the problem is solvable in time $O(n^k)$, which is polynomial for constant k. Hence, we can give more accurate statements about the complexity of Subset Sum if we characterize instances not only by their size, but also by the *parameter* k.

In fact, the running time $n^{O(k)}$ is trivial for many popular parameter choices for NP-complete problems. Therefore, fixed-parameter tractability (and thus also kernelization) even aims at running times that depend only polynomially on the instance sizes, i.e., that are of the form $f(k) \cdot n^{O(1)}$ for some function f. These bounds scale much better when we increase k or n.

The main idea of kernelization is to reduce the size of a given instance in the hope that the remaining part will often be small enough to be tractable with an exponential-time algorithm for the problem (this is where the $f(k)$ part of the running time comes from). As an example, consider the decision variant of the Vertex Cover problem together with the size k of a minimum vertex cover as parameter, i.e., the question: Is there a vertex subset S of size k in a graph G such that every edge of G is incident to S? If such a vertex cover exists, then we know for sure that every vertex v of degree more than k must be contained in it, as the edges incident to v cannot be covered in any other way by at most k vertices. Hence, the question whether G contains a vertex cover of size k is equivalent to the question whether $G - v$ contains a vertex cover of size $k - 1$.

The combination of several such preprocessing steps often leads to equivalent instances where the size of the instance is a function only in k, but independent of the original instance size. This is called a kernel.

Our first application of kernelization is to tackle Max-Cut and some generalizations of it from a different angle. We show that Max-Cut admits a kernel in which the number of vertices is linear in our parameter k. This parameter k is roughly the size of the maximum cut in the graph minus the guaranteed maximum cut size every connected graph of the same order has. Second, we show kernels for various Subset Sum and Knapsack variants. Finally we prove that for several problems, kernels for the unweighted case can be extended to kernels for the weighted counterparts.

1.1 Smoothed Analysis

We will consider two different models for smoothed analysis. The first one is the original model suggested by Spielman and Teng [100], in which Gaussian noise is added to the input. Assume for simplicity that we consider a problem with input set $x_1, \ldots, x_n \in \mathbb{R}$ and that every variable of the input is about to be perturbed. Let $\mathcal{A}$ be an algorithm for the given problem. Let us for simplicity assume that $\mathcal{A}$ is invariant under scaling of the input numbers. Then we can w.l.o.g. scale down the input such that $x_1, \ldots, x_n \in [-1, 1]$. Then we add independent Gaussian random variables $N_1, \ldots, N_n \sim \mathcal{N}(0, \sigma^2)$ with mean value 0 and variance σ^2 to the input. Let f be the running time of an algorithm $\mathcal{A}$ for the considered problem. The worst-case running time of $\mathcal{A}$ can then be expressed as

$$T_{\text{worst-case}} = \sup_{x_1,\ldots,x_n \in [-1,1]} f(x_1, \ldots, x_n).$$

The smoothed running time of $\mathcal{A}$ is in contrast defined as the worst expected running time when adding Gaussian noise to the input:

$$T_{\text{smoothed}} = \sup_{x_1,\ldots,x_n \in [-1,1]} \mathbf{E}_{N_1,\ldots,N_n \sim \mathcal{N}(0,\sigma^2)} \left[f(x_1 + N_1, \ldots, x_n + N_n) \right].$$

We can equivalently write

$$T_{\text{smoothed}} = \sup_{x_1,\ldots,x_n \in [-1,1]} \mathbf{E}_{N_1 \sim \mathcal{N}(x_1,\sigma^2),\ldots,N_n \sim \mathcal{N}(x_n,\sigma^2)} \left[f(N_1, \ldots, N_n) \right].$$

Note that the smoothed running time of the algorithm depends on the choice of the standard deviation σ, that is, the smoothed running time of $\mathcal{A}$ is always expressed as a function depending on σ. The smaller σ is, the more powerful is the adversary and the model converges to a worst-case analysis. On the other hand, if σ tends to infinity, the random noise dominates the input and therefore our model resembles an average-case analysis. Therefore, smoothed analysis can be seen as a natural interpolation between the worst and the average case.

The second smoothed analysis model we consider is the generalization introduced by Beier and Vöcking [9]. Here we are even allowed to choose the probability distribution for the random noise, instead of being fixed to the Gaussian distribution. Let us again assume that our algorithm $\mathcal{A}$ is invariant under scaling such that we can w.l.o.g. scale down the input such that $x_1, \ldots, x_n \in [-1, 1]$. Now every variable x_i is replaced by a random variable $X_i \sim \varphi_i$, where $\varphi_i \colon [-1, 1] \to [0, \phi]$ is an adversarially chosen density function bounded from above by a parameter ϕ. The variables $X_1, \ldots, X_n$ are of course again drawn independently of each other. The smoothed running time is then defined as

$$T_{\text{smoothed}} = \sup_{\varphi_1,\ldots,\varphi_n \colon [-1,1] \to [0,\phi]} \mathbf{E}_{X_1 \sim \varphi_1,\ldots,X_n \sim \varphi_n} \left[f(X_1, \ldots, X_n) \right].$$

The parameter ϕ plays a similar role as $1/\sigma$ in the traditional model: If $\phi \to \infty$, the adversary becomes arbitrarily powerful and the analysis resembles a worst-case analysis. On the other hand, if $\phi = 1/2$, which is the minimum value such that $\varphi_1, \ldots, \varphi_n$ exist, we have reached an average-case analysis.

In order to get an idea what the parameter ϕ means, one can think of the input distributions in the following way: Given ϕ, the adversary can choose for every input variable X_i an interval of length $1/\phi$ from which X_i is chosen uniformly at random. But of course the adversary is not restricted to this kind of density functions.

Technically, the second model is not a proper generalization for the first model because Gaussian random variables have an infinite support, whereas the density functions of the second model are restricted to the interval $[-1, 1]$. Nevertheless, it seems natural to speak of a generalization because we are mainly interested in the algorithmic behavior for $\sigma \to 0$, in which case the tails of the Gaussian density function not contained in an interval of constant length become negligible.

Smoothed analysis is even more general than explained above because we can choose not to perturb all variables of the input, but only a fraction of it. For example, we could say that only the profits, but not the weights in a Knapsack instance are perturbed. This way we can keep hard constraints on parts of the input for which it is harder to legitimate random noise.

After the introduction of smoothed analysis to explain why the simplex method solves linear programs efficiently in practice despite its exponential worst-case running time [100],

it has gained a lot of attention and it has been used to analyze a wide variety of optimization problems and algorithms (see, e.g., the surveys [83, 101]). It seems to be especially well-suited for local search methods, as these are for many optimization problems prime examples of algorithms with exponential worst-case running time that work well and efficiently in practice. Let us give two examples.

Methods based on local search are very successful for the TSP. One commonly used heuristic is k-Opt, which starts with an arbitrary tour and replaces in each local improvement k edges of the current tour by k other edges. Usually the 2-Opt heuristic needs a clearly subquadratic number of improving steps until it reaches a local optimum and the computed solution is within a few percentage points of the global optimum [72]. On the other hand it is known that even on two-dimensional Euclidean instances 2-Opt can take an exponential number of steps to reach a local optimum [38]. Therefore, Englert et al. [38] analyzed the smoothed number of local improvements of 2-Opt and proved that it is polynomially bounded in the number of nodes and the perturbation parameter ϕ. Manthey and Veenstra [85] showed smoothed polynomial bounds in the classical model with Gaussian perturbations.

Another area in which local search methods are predominant is clustering. The k-means method is one of the most widely used clustering algorithms that is very efficient on real-world data (see, e.g., [10]), but exhibits exponential worst-case running time [104]. Arthur and Vassilvitskii [5] initiated the smoothed analysis of the k-means method that culminated in a proof that the smoothed running time of the k-means method is polynomial [4]. The smoothed analysis of the k-means method has also been extended from squared Euclidean distances to general Bregman divergences [84]. Arthur and Vassilvitskii [5] also showed that the smoothed running time of the ICP algorithm for minimizing the difference between two sets of points is polynomial while its worst-case running time is exponential.

1.2 Kernelization

The field of fixed-parameter tractability goes back to the seminal work by Downey and Fellows [32]. Since then it has gained huge popularity, see for example the books by Downey and Fellows [33] and Cygan et al. [28]. This field and kernelization belong to the area of parameterized complexity. Therefore, let us first formally define what a parameterized problem is.

A *parameterized problem* is a language $\Pi \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet; the second component k of instances $(I, k) \in \Sigma^* \times \mathbb{N}$ is called the *parameter*. A problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is *fixed-parameter tractable* if it admits a *fixed-parameter algorithm*, which decides instances (I, k) of Π in time $f(k) \cdot |I|^{\mathcal{O}(1)}$ for some computable function f. The class of fixed-parameter tractable problems is denoted by FPT.

A *kernelization* for a parameterized problem Π is an efficient, i.e., polynomial-time, algorithm that given any instance (I, k) returns an instance (I', k') with $|I'| + k' \leq f(k)$ for some computable function f such that $(I, k) \in \Pi$ if and only if $(I', k') \in \Pi$. We call (I', k') the *kernel* and f the *size* of the kernel, and we have a polynomial kernel if $f(k)$ is polynomially bounded in k.

The size of the kernel, as stated, is defined by the number of bits we need to encode the equivalent instance (I', k'). However, for many graph problems it makes sense to emphasize the kernel size with respect to other measures. For example, we will derive kernels for Max-Cut Above Edwards-Erdős Bound that contain a linear number of vertices, despite needing a quadratic number of bits to encode all the edges in the instance. But the number of vertices in the kernel is arguably more crucial because when applying a brute force algorithm to the kernel it has running time $\mathcal{O}(c^k)$ for some constant c. If we only stated that we have a

kernel with quadratic encoding size, we could not rule out that the brute force-algorithm could have a running time of $\mathcal{O}(c^{k^2})$ for some constant c.

The concepts of fixed-parameter tractability and kernelization are closely related. In fact, a parameterized problem is fixed-parameter tractable if and only if it has a kernelization: If we have a kernelization of size $f(k)$, it is easy to see that we can obtain a fixed-parameter algorithm by first applying the kernelization algorithm and then using a brute-force approach on the resulting kernel. For the other direction, assume there is a fixed-parameter algorithm $\mathcal{A}$ with running time $f(k)\cdot|I|^c$ for some constant $c > 0$. We let algorithm $\mathcal{A}$ run on an input (I, k) for at most $|I|^{c+1}$ steps. If it terminates, we can replace (I, k) by a trivial "yes"- or "no"-instance of constant size. Otherwise, it must hold that $f(k) \geq |I|$. This means that (I, k) itself is a kernel of size at most $f(k) + k$.

Although fixed-parameter tractability and kernelization are equivalent according to the above-mentioned result, the kernels implied by this fact are usually of superpolynomial size. This is because the size nearly matches the $f(k)$ from the running time, which for NP-hard problems is usually exponential as typical parameters are upper bounded by the instance size.

1.3 Outline and Bibliographical Notes

Section 2 gives an overview of all results covered in this work. The following five sections are used to prove these results about local search for the Maximum-Cut problem (Sections 3-4) and scheduling (Section 5) as well as about kernelizations for the Signed Maximum-Cut Problem (Section 6), generalizations of it (Section 7), and several weighted problems (Section 8). In Section 9 we draw conclusions and point out possible future research questions.

The results in Section 3 have been published at a conference and in a journal:

- Michael Etscheid and Heiko Röglin. Smoothed analysis of local search for the maximum-cut problem. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 882–889, 2014.
- Michael Etscheid and Heiko Röglin. Smoothed analysis of local search for the maximum-cut problem. *ACM Trans. Algorithms*, 13(2):Art. 25, 12, 2017. URL: `https://doi.org/10.1145/3011870`.

Preliminary versions of the results in Section 4 have been published at a conference:

- Michael Etscheid and Heiko Röglin. Smoothed analysis of the squared euclidean maximum-cut problem. In *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA)*, pages 509–520, 2015.

The results in Section 5 have been published at a conference:

- Tobias Brunsch, Michael Etscheid, and Heiko Röglin. Bounds for the convergence time of local search in scheduling problems. In *International Conference on Web and Internet Economics*, pages 339–353. Springer, 2016.

The results in Section 6 and Section 7 have been published at a conference and in a journal:

- Michael Etscheid and Matthias Mnich. Linear kernels and linear-time algorithms for finding large cuts. In *Proc. ISAAC 2016*, volume 64 of *Leibniz Int. Proc. Informatics*, pages 31:1–31:13, 2016.
- Michael Etscheid and Matthias Mnich. Linear kernels and linear-time algorithms for finding large cuts. *Algorithmica*, Oct 2017. URL: `https://doi.org/10.1007/s00453-017-0388-z`, `doi:10.1007/s00453-017-0388-z`.

The results in Section 8 have been published at a conference and in a journal:

- Michael Etscheid, Stefan Kratsch, Matthias Mnich, and Heiko Röglin. Polynomial kernels for weighted problems. In *Mathematical foundations of computer science 2015. Part II*, volume 9235 of *Lecture Notes in Comput. Sci.*, pages 287–298. Springer, Heidelberg, 2015. URL: `https://doi.org/10.1007/978-3-662-48054-0_24`.
- Michael Etscheid, Stefan Kratsch, Matthias Mnich, and Heiko Röglin. Polynomial kernels for weighted problems. *J. Comput. System Sci.*, 84:1–10, 2017. URL: `https://doi.org/10.1016/j.jcss.2016.06.004`.

2 Results

In this section we give an overview of all results of this thesis.

2.1 Smoothed Analysis of Local Search for Max-Cut

2.1.1 General Graphs

We analyze the simple local search algorithm FLIP for the Max-Cut Problem. An instance of this problem consists of an undirected graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$. We call a partition (P_1, P_{-1}) of the nodes V a cut and define its weight to be the total weight of the edges between P_1 and P_{-1}. The FLIP algorithm starts with an arbitrary cut (P_1, P_{-1}) and iteratively increases the weight of the cut by moving one vertex from P_1 to P_{-1} or vice versa, as long as such an improvement is possible. It is well-known that any locally optimal cut is a 2-approximation of a maximum cut (see, e.g., [74]). However, it is also known that the problem of finding a locally optimal cut is PLS-complete (for an explanation, see Section 2.1.3) and that there are instances with cuts from which every sequence of local improvements to a local optimum has exponential length [97].

In the smoothed analysis model we consider, an adversary specifies an arbitrary graph $G = (V, E)$ with n nodes. Instead of fixing each edge weight deterministically he can only specify for each edge $e \in E$ a probability density function $f_e : [-1, 1] \to [0, \phi]$ according to which the weight $w(e)$ is chosen independently of the other edge weights. We point out again that the parameter $\phi \geq 1/2$ determines how powerful the adversary is. He can, for example, choose for each edge weight an interval of length $1/\phi$ from which it is chosen uniformly at random. This shows that in the limit for $\phi \to \infty$ the adversary is as powerful as in a classical worst-case analysis, whereas the case $\phi = 1/2$ constitutes an average-case analysis with uniformly chosen edge weights. Note that the restriction to the interval $[-1, 1]$ is merely a scaling issue and no loss of generality.

For a given instance of the Max-Cut Problem we define the *number of steps of the FLIP algorithm* on that instance to be the largest number of local improvements the FLIP algorithm can make for any choice of the initial cut and any pivot rule determining the local improvement that is chosen if multiple improving steps are possible. Formally, this can be described as the longest path in the transition graph of the FLIP algorithm. We are interested in the *smoothed number of steps of the FLIP algorithm*. This quantity depends on the number n of nodes and the perturbation parameter ϕ and it is defined as the largest expected number of steps the adversary can achieve by his choice of the graph G and the density functions f_e. We then obtain the following result.

▶ **Theorem 2.1.** *The smoothed number of steps of the FLIP algorithm is bounded from above by a polynomial in $n^{\log n}$ and ϕ.*

This result significantly improves upon the exponential worst-case running time of the FLIP algorithm. While a polynomial instead of a quasi-polynomial dependence on n would be desirable, let us point out that the theorem is very strong in the sense that it holds for all initial cuts and all pivot rules. The theorem shows that worst-case instances, on which FLIP can take an exponential number of steps, are fragile and unlikely to occur in the presence of a small amount of random noise.

2.1.2 Squared Euclidean Distances

We continue the smoothed analysis of local search for Max-Cut and consider the special case in which the nodes are points in a d-dimensional space and the edge weights are given by the squared Euclidean distances between these points. In this setting, a different encoding for problem instances becomes natural: An instance is now given by a finite set $\{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$ of points that is to be partitioned into two parts P_1 and P_{-1} such that the weight $\sum_{x \in P_1} \sum_{y \in P_{-1}} \|x - y\|^2$ becomes maximal, where $\|x - y\|$ denotes the Euclidean distance between x and y. Squared Euclidean distances are common in many clustering applications.

When an instance is given by some points in $\mathbb{R}^d$ instead of a set of weighted edges, we cannot simply add random variables to the given edge weights and expect that there is a point set in $\mathbb{R}^d$ that complies with the updated edge weights. Therefore we have to change the influence of randomness in the smoothed setting. In the model we consider now, an adversary specifies an arbitrary set $x_1, \ldots, x_n \subseteq [0,1]^d$ of n points. Then each *point* is randomly perturbed by adding a Gaussian vector of standard deviation σ to it. We will denote the Gaussian random vectors with mean values $x_1, \ldots, x_n$ and standard deviation σ by $X_1, \ldots, X_n$. Note again that the restriction to $[0,1]^d$ is merely a scaling issue and entails no loss of generality.

The smoothed number of steps of the FLIP algorithm is in this model of course the largest expected number of steps the adversary can achieve by his choice of the point set $x_1, \ldots, x_n$. This quantity depends on the number n of nodes and the standard deviation σ. We obtain the following result.

▶ **Theorem 2.2.** *For any dimension $d \geq 2$, the smoothed number of steps of the FLIP algorithm for complete graphs with squared Euclidean distances as edge weights is bounded from above by $2^{O(d)} \cdot n^{23} \cdot \max\{\sigma^{-8}, n^4\}$.*

As a contrast, we show that the worst-case number of steps of the FLIP algorithm is exponential even in dimension $d = 2$ on non-complete graphs with squared Euclidean distances.

▶ **Theorem 2.3.** *For every $n \in \mathbb{N}$, there is a weighted graph G on n vertices with the following properties:*

- *There is an embedding of $V(G)$ into $\mathbb{R}^2$ such that the edge weights correspond to the squared Euclidean distance of the incident vertices.*
- *The number of steps of the FLIP algorithm is at least $\Omega(2^{n/6})$.*

The theorems indicate that worst-case instances for squared Euclidean distances are fragile and unlikely to occur in the presence of a small amount of random noise. We view Theorem 2.2 as a further step towards understanding the behavior of local search heuristics on semi-random inputs.

The conference version of this work [46] used a novel approach, which was different to the typical smoothed analysis of local search heuristics in the literature. After $O(d)$ vertices flipped sides of the partition, we constructed a matrix of dimension $d \times d$, the entries of which were independently normally distributed, and a vector of dimension d such that $\sum_{x \in P_1} x - \sum_{x \in P_{-1}} x$ is the solution of the resulting system of linear equations. We then used a result by Sankar, Spielman, and Teng [96] that the condition number of the matrix is bounded with high probability, i.e., that we can compute $\sum_{x \in P_1} x - \sum_{x \in P_{-1}} x$ with only a small error without having to take an exponentially-sized union bound over the configuration

of all points. In this updated version, we take the approach of Angel et al. [3] and simply guess $\sum_{x\in P_1} x - \sum_{x\in P_{-1}} x$ and a few other terms up to a small error. This not only simplifies the analysis significantly, but also improves the resulting bound in the theorem.

2.1.3 Max-Cut in the Context of PLS

We give a short introduction to the class PLS following [1]. The problem of computing a local optimum with respect to some neighborhood structure belongs to PLS if it is possible to compute in polynomial time the objective value of a given solution, an initial feasible solution, and a better solution in the neighborhood of a given locally non-optimal solution. PLS contains many natural local seach problems like linear programming, where the neighborhood are the geometrical neighbor vertices on the polytope (this corresponds to the simplex algorithm), the Traveling Salesman Problem with the k-Opt neighborhood, and Max-2Sat under the FLIP neighborhood.

A PLS-reduction from a local-search problem Π_1 to a local-search problem Π_2 consists of two polynomial-time computable functions h and g: The function h maps instances x of Π_1 to instances of Π_2 and the function g maps solutions of $h(x)$ back to solutions of x such that $g(s, x)$ is a local optimum for x if s is a local optimum for $h(x)$. In other words, we can find local optima for a Π_1 instance by translating it to a Π_2 instance and computing a local optimum for this problem. This means that Π_1 is not "harder" than Π_2, neglecting polynomial-time blow-ups.

A problem Π is PLS-complete if every problem in PLS can be PLS-reduced to it. It was shown that Max-Cut together with the FLIP neighborhood is PLS-complete [97] even for graphs of maximum degree five [37]. This implies that one cannot efficiently compute a locally optimal cut unless PLS $\subseteq$ P.

Furthermore, the reductions used to show PLS-completeness are all *tight*. Intuitively this means the following for a PLS-reduction (h, g) from Π_1 to Π_2: Let x be an instance of Π_1 and let $y = h(x)$. If there is an edge from a solution s_1 to a solution s_2 in the transition graph of y, then either $g(s_1, x) = g(s_2, x)$ or there is an edge from $g(s_1, x)$ to $g(s_2, x)$ in the transition graph of x. In other words, a path in the transition graph of $h(x)$ gets translated to a path in the transition graph of x that is not longer. For a formal definition, we refer again to [1]. As a consequence, local search for $h(x)$ cannot terminate faster than local search for x. Because there is a problem in PLS for which local search takes exponential time (the solution set is the set of n-bit integers and the neighborhood of i is $\{i - 1\}$) and this problem can be tightly reduced to Max-Cut, it follows for the Max-Cut problem that there exist initial cuts from which any sequence of local improvements to a local optimum has exponential length.

The Max-Cut Problem is not only interesting for itself, but it is also interesting because it is structurally one of the easiest PLS-complete problems. It is used as a starting point for many PLS-reductions so that the analysis of the FLIP algorithm might also shed some light on other local search algorithms.

2.1.4 Related Work

The Max-Cut Problem has been considered several times in the model of smoothed analysis. Elsässer and Tscheuschner [37] showed that the smoothed number of steps of the FLIP algorithm is polynomially bounded if the graph G has at most logarithmic degree. After the conference version of our result on squared Euclidean instances [46], Angel et al. [3] improved the quasipolynomial bound from Theorem 2.1 to a polynomial bound for complete graphs.

This result, however, does not imply Theorem 2.2, as the edge weights in the model of Angel et al. are chosen independently from each other, whereas the edge weights in our model for squared Euclidean distances heavily depend on each other.

Schulman [98] studied the min-sum 2-clustering problem for squared Euclidean distances. In this problem, the input also consists of a finite set of points $\mathcal{X} \subseteq \mathbb{R}^d$ and the goal is to find a partition of $\mathcal{X}$ into two classes $\mathcal{X}_1$ and $\mathcal{X}_2$ such that the sum of the edge weights inside the two classes (i.e., $\sum_{x,y\in\mathcal{X}_1} \|x-y\|^2 + \sum_{x,y\in\mathcal{X}_2} \|x-y\|^2$) becomes minimal. This problem is equivalent to the Max-Cut Problem with squared Euclidean distances (not in terms of approximation though) and hence the FLIP algorithm can also be seen as a local search algorithm for min-sum 2-clustering. Schulman gives an algorithm that solves the problem optimally in time $O(n^{d+1})$. His algorithm is based on the observation that in an optimal clustering the classes $\mathcal{X}_1$ and $\mathcal{X}_2$ are separated by a sphere and there are only $O(n^{d+1})$ spheres that one has to consider. He also presented a polynomial time approximation scheme.

In recent years there has been an increased interest in the class PLS due to its connection to algorithmic game theory. For many games the problem of computing pure Nash equilibria belongs to PLS and is often even PLS-complete. This is due to the fact that better- and best-response dynamics followed by agents in a game can be interpreted as variants of local search whose local optima are exactly the pure Nash equilibria. This line of research has been initiated by Fabrikant et al. [49] who showed that for network congestion games the problem of computing a pure Nash equilibrium is PLS-complete. Their proof has been simplified by Ackermann et al. [2] who gave a simple reduction from the Max-Cut Problem.

Even the Max-Cut Problem itself has been formulated as a *party affiliation game* in which agents (nodes) have to choose one of two sides and the edge weights are a measure for how much two agents like or dislike each other [11]. Theorem 2.1 has direct consequences for these games as well and shows that any sequence of better responses has with high probability at most quasi-polynomial length if the edge weights are subject to random noise. Since every local optimum is a 2-approximation, this implies in particular that with high probability after a quasi-polynomial number of better responses the social value is at least half of the optimal value. This is in contrast to the worst-case result of Christodoulou et al. [21] who show that there are instances with exponentially long best response sequences after which the social value is only a $1/n$-fraction of the optimal value.

Christodoulou et al. also show that if agents are allowed to play best responses in a random order then already after one round, in which every agent has been activated at least once, an 8-approximation is achieved. Awerbuch et al. [7] considered α-best response dynamics in which agents only change their strategy if this increases their payoff by a certain factor α. They prove that this α-best response dynamics reaches after a polynomial number of steps a $(2+\varepsilon)$-approximation if every player gets the chance to move at least once every polynomial number of steps. While these positive results hold only for certain sequences of best responses, our positive result holds for any quasi-polynomial sequence of better responses.

2.2 Scheduling

Additionally to the FLIP algorithm for Max-Cut, we analyze another local search approach for the following scheduling problem: Given m machines and n jobs, find an assignment of the jobs to the machines minimizing the maximum costs of a job, which are defined according to a coordination mechanism. The jobs may have different job sizes and the machines may have different machine speeds. A typical definition of the costs of a job is the sum of the job sizes assigned to the same machine divided by the machine speed, which is a natural

choice when the makespan is to be minimized. In other contexts it might be more realistic to assume an order in which the jobs on a machine are executed and that a job only pays for the execution time of itself and all previous jobs.

Even in the case of identical machine speeds, the problem is known to be strongly NP-hard [55] and local search is a popular tool to approximate good solutions. Here, a job unilaterally changes its assignment and moves to another machine if it can reduce its costs this way. Throughout this work, we assume a best response policy, i.e., a moving job selects a machine that minimizes its costs. If there is no job left that can improve its costs, we have attained a local optimum, which is guaranteed to be reached after a finite number of steps. Although the quality of the worst local optimum has been thoroughly analyzed [16, 30, 40, 53, 99], there is not much work about the convergence time needed to find one via local search.

2.2.1 Terminology

Let us first describe the studied problem in detail. Consider an instance with m machines and n jobs. Each machine i has a *speed* $s_i \in \mathbb{Q}_{>0}$ and each job j has a *job size* $p_j \in \mathbb{Q}_{>0}$. Let $s_{\min}, s_{\max}, p_{\min}$, and $p_{\max}$ be the minimal and maximal speeds and job sizes. Let $W = \sum_{j=1}^{n} p_j$ be the sum of the job sizes. For *identical machines*, $s_{\max} = s_{\min} = 1$, and for *unit-weight jobs*, $p_{\max} = p_{\min} = 1$.

For an assignment $\sigma\colon \{1,\dots,n\} \to \{1,\dots,m\}$ that maps the jobs to the machines, let $L_i = \sum_{j \in \sigma^{-1}(i)} p_j/s_i$ be the *load* of machine i. The maximum load is called *makespan*. The *costs* of a job j are defined according to a *coordination mechanism*, which assigns costs to every job depending only on the set of jobs that have chosen the same machine, but not on the residual schedule.

1. In the *Makespan* model, all jobs assigned to the same machine are executed simultaneously such that the costs $c_j^\sigma = L_{\sigma(j)}$ of a job j correspond to the load of its machine. This is the most common coordination mechanism and it corresponds to linear weighted congestion games on parallel links.

2. In the *FIFO* model, the jobs on each machine are executed one after another. Therefore, we need a permutation π on the jobs that determines the order in which the jobs on a machine get processed. The costs of a job j are then $c_j^{(\sigma,\pi)} = \sum_{j' \in J_\sigma^\pi(j)} \frac{p_{j'}}{s_{\sigma(j)}}$, where $J_\sigma^\pi(j)$ is the set of jobs j' on the same machine with $\pi(j') \le \pi(j)$. If a job j jumps to another machine, it is inserted as the last job, i.e., $\pi(j) = n$.

3. In the *SJF* (shortest job first) model, the jobs are executed one after another, but the permutation of the jobs is at any time implicitly given by their job sizes where the smallest job on a machine is executed first. Ties for jobs of equal size are broken arbitrarily. This means that the costs of a job are defined as $c_j^\sigma = \sum_{j'\colon \sigma(j')=\sigma(j) \wedge \pi(j') \le \pi(j)} \frac{p_{j'}}{s_{\sigma(j)}}$, where π is a permutation of the jobs assigned to machine $\sigma(j)$ such that $\pi(j') < \pi(j)$ if $p_{j'} < p_j$ and $\pi(j') > \pi(j)$ if $p_{j'} > p_j$.

The FIFO model is not a coordination mechanism in the classical sense as the order in which the jobs are executed depends on previous iterations. Nevertheless, we believe that this model can easily be motivated by many real-world applications where the first-come, first-served principle is ubiquitous.

In the case of the Makespan and SJF models, we call σ a *schedule*. In the FIFO model, we call the tuple (σ, π) a *schedule*. Often, we omit the parameters σ and π if they are clear

from the context, or we replace them by an iteration number t. Then we mean the schedule before the move of iteration t gets executed.

We say that a job is *unsatisfied* if it could improve its costs by jumping to a different machine. When an unsatisfied job jumps, it always jumps to a machine minimizing its costs, i.e., we consider best response dynamics. If there is no unsatisfied job, we call the current schedule a *local optimum*. The *convergence time* for an instance is the maximum number of jumps it can take starting from an arbitrary schedule until a local optimum is reached. The *price of anarchy* is the ratio of the makespans of the worst local optimum and the global optimum.

If there are several unsatisfied jobs, we choose the next job to jump according to a pivot rule:

- *Best Improvement*: Select a job for which the largest improvement of its costs is possible.
- *Random*: Select a job uniformly at random from the set of unsatisfied jobs.
- *Min Weight*: Select a smallest unsatisfied job.
- *Max Weight*: Select a largest unsatisfied job.
- *Fixed Priority*: Select the unsatisfied job with the largest priority according to a given order on the jobs. This pivot rule includes Min Weight and Max Weight as special cases.

2.2.2 Smoothed Analysis

We will again use the more general model of smoothed analysis introduced by Beier and Vöcking [9], in which the adversary is even allowed to specify the probability distribution of the random noise. The influence he can exert is described by a parameter $\phi \geq 1$ denoting the maximum density of the noise. The model is formally defined as follows. The adversary chooses the following input data:

- the number m of machines;
- arbitrary machine speeds $s_1, \ldots, s_m$ in the case of non-identical machines;
- the number n of jobs;
- for each p_j, a probability density $f_j : [0,1] \to [0,\phi]$ according to which p_j is chosen independently of the sizes of the other jobs.

The *smoothed convergence time* is the worst expected convergence time and the *smoothed price of anarchy* is the worst expected price of anarchy the adversary can achieve by his choices. Note that the only perturbed part of the instance are the job sizes. These perturbed job sizes are easily justifiable in many practical applications.

2.2.3 Related Work and Results

The notion of coordination mechanisms has been introduced by Christodoulou et al. [20] in the context of congestion games. There has been extensive research about the price of anarchy for the different coordination mechanisms. In the Makespan model it is constant for identical machines [53, 99] and $\Theta\left(\min\left\{\frac{\log m}{\log\log m}, \log\frac{s_{\max}}{s_{\min}}\right\}\right)$ for related machines [30]. The smoothed price of anarchy for related machines is $\Theta(\log\phi)$ regardless of whether the job sizes [16] or the machine speeds [40] are perturbed.

Immorlica et al. [68] showed a price of anarchy of $2 - 1/m$ for identical and $\Theta(\log m)$ for related machines for the SJF model, which is the same as for list schedules, i.e., schedules that are generated by a greedy assignment.

The FIFO model has been introduced implicitly by Brunsch et al. [16] through the equivalent concept of near list schedules which was used as a generalization of local optima w.r.t. the Makespan model and list schedules. They showed that the smoothed price of anarchy is $\Theta(\log \phi)$. We complement this by the corresponding worst-case results for identical and related machines to obtain the same tight bounds as in the SJF model.

There is less known about the convergence times in the different models. As we are up to our knowledge the first ones who consider the FIFO model, there are no previous results about convergence times. We show tight results for special cases like identical machines and several upper bounds depending on $W/p_{\min}$ for different pivot rules in the general case. Although we conjecture polynomial bounds for all cases, we give the first non-trivial proofs for this natural problem. Immorlica et al. [68] showed for the SJF model that if the jobs are asked on a rotational basis if they want to jump, the convergence time is in $O(n^2)$. This is in sharp contrast to our result that for the Min Weight pivot rule it can take an exponential number of iterations even in the case of two identical machines.

Brucker et al. [15] considered the Makespan model with the difference that only jobs from a machine with maximum load—a so-called *critical* machine—are allowed to jump, i.e., a local optimum is reached as soon as every job on a critical machine is satisfied. They gave an algorithm that finds a local optimum after $O(n^2)$ improving steps for identical machines. From this, one can easily derive an algorithm for identical machines in the Makespan model: Run Brucker's algorithm exhaustively until every job on a critical machine is satisfied. As on identical machines the minimum load of a machine is monotonically increasing, these jobs cannot become unsatisfied again by any sequence of improving steps. Hence, the jobs on the critical machine are fixed and therefore we can remove the critical machine together with its assigned jobs from the instance. Repeating this argument yields a running time of $O(n^2m)$ improving steps. As the monotonicity argument does not hold anymore in the case of related machines, we are not aware of a way to use similar results by Schuurman and Vredeveld [99] and Hurkens and Vredeveld [67] for Brucker's model on related machines.

For the Makespan model and identical machines, Goldberg [56] considers randomized local search, where in each step a job and a machine are selected uniformly at random, and the job moves to that machine if it is an improving step . He shows that random local search converges in expected $(m + n + \frac{p_{\max}}{p_{\min}})^{O(1)}$ time.

In the Makespan model, Feldmann et al. [50] provided an $O(nm^2)$-time Nashification algorithm, which, given an arbitrary schedule, computes a local optimum without increasing the social cost, i.e., the makespan in our case. They further showed that the convergence time on identical machines is bounded by $\Omega(2^{\sqrt{n}})$ and $O(2^n)$. To be more precise, Even-Dar et al. [48] showed (again for identical machines) that the Max Weight and the Random pivot rule converge in n and $O(n^2)$ steps, respectively, while the Min Weight pivot rule can take an exponential number of steps. We extend this result by showing that every pivot rule converges in $O(n \cdot W/p_{\min})$ steps, which can be seen as a generalization of their result that every pivot rule converges in $O(W + n)$ steps in the case of integer weights. For related machines and unit-weight jobs, Even-Dar et al. [48] showed that there is a pivot rule that converges in mn steps. We improve this by showing that the convergence time for any pivot rule with best response policy is exactly n. For the case of related machines and integral job sizes and machine speeds, they showed that any pivot rule converges in $O(W^2 \cdot s_{\max}^2/s_{\min})$ steps. We prove a similar bound for the Best Improvement pivot rule on arbitrary weights. An overview of our results on convergence times is given in Table 1, Table 2, and Table 3.

identical machines	$n-1$	(Thm. 5.3)
unit-weight jobs	n	(Thm. 5.12)
two machines	$\Theta(n)$	(Thm. 5.15)
Best Improvement	$O(m^2 n \cdot W/p_{\min})$	(Thm. 5.16)
Random	$O(m^2 n^2 \cdot W/p_{\min})$	(Thm. 5.17)
Fixed Priority	$O(n^2 \cdot W/p_{\min})$	(Thm. 5.18)
lower bounds	$\Omega(mn)$, $\Omega(m^2)$ for Min Weight	(Thm. 5.19)

Table 1 FIFO convergence times

identical machines	$O(n \cdot W/p_{\min})$	(Thm. 5.4)
unit-weight jobs	n	(Thm. 5.12)
Best Improvement	$O(m^2 n \cdot W^2/p_{\min}^2)$	(Thm. 5.26)

Table 2 Makespan convergence times

Max Weight on two identical machines	$2^{\Omega(n)}$	(Thm. 5.5)
Max Weight on two identical machines with random weights	$2^{\Omega(\sqrt{n})}$	(Thm. 5.5)
Min Weight	n	(Thm. 5.27)
Random	$O(n^2)$	(Thm. 5.27)

Table 3 SJF convergence times

2.3 Signed Max-Cut Above Edwards-Erdős Bound and Linear Vertex Kernels for λ-Extendible Properties

We now turn to the parameterized complexity part of this thesis. A recent paradigm in parameterized complexity is to not only show a problem to be fixed-parameter tractable, but indeed to give algorithms with *optimal* running times in *both* the parameter and the input size. Ideally, we strive for algorithms that are *linear* in the input size, and optimal in the dependence on the parameter k assuming a standard hypothesis such as the Exponential Time Hypothesis [69]. New results in this direction include linear-time fixed-parameter algorithms for Graph Bipartization [70], Planar Subgraph Isomorphism [31], DAG Partitioning [103], Planar Independent Set [34] and Subset Feedback Vertex Set [79].

Here, we consider the Max-Cut problem and some generalizations from the view-point of linear-time fixed-parameter algorithms. We refer to the survey [91] for an overview of the research area.

We focus on Max-Cut *parameterized above Edwards-Erdős bound*. This parameterization is motivated by the classical result of Edwards [35, 36] that any connected graph on n vertices and m edges admits a cut of size at least

$$m/2 + (n-1)/4 \ . \tag{1}$$

This lower bound is known as the *Edwards-Erdős bound*, and it is tight for cliques of every odd order n. Ngd*o*c and Tuza [89] gave a linear-time algorithm that finds a cut of size at least (1).

Parameterizing Max-Cut above Edwards-Erdős bound means, for a given connected graph G and integer k, to determine if G admits a cut that exceeds (1) by an amount of k: formally, the problem Max-Cut Above Edwards-Erdős Bound (Max-Cut AEE) is to determine if $\mathsf{mc}(G) \geq |E(G)|/2 + (|V(G)| - 1 + k)/4$ for a given pair (G, k), where $\mathsf{mc}(G)$ is the maximum number of edges of a bipartite subgraph of G. It was asked in a sequence of papers [22, 60, 81, 82] whether Max-Cut AEE is fixed-parameter tractable, before Crowston et al. [25] gave an algorithm that solves instances of this problem in time $8^k \cdot O(n^4)$, as well as a kernel of size $O(k^5)$. Their result inspired a lot of further research on this problem, leading to smaller kernels of size $O(k^3)$ [23] and fixed-parameter algorithms for generalizations [87] and variants [26].

In the Signed Max-Cut problem, we are given a graph G whose edges are labelled by $(+)$ or $(-)$, and we seek a maximum balanced subgraph H of G, where *balanced* means that each cycle has an even number of negative edges. Max-Cut is the special case where all edges are negative. Signed Max-Cut finds applications in, e.g., modelling social networks [63], statistical physics [8], portfolio risk analysis [64], and VLSI design [18]. The dual parameterization of Signed Max-Cut by the number of edge deletions was also shown to be fixed-parameter tractable [66].

Poljak and Turzík [90] showed that the property of having a large cut (i.e., a large bipartite subgraph) can be generalized to many other classical graph properties, including properties of oriented and edge-labelled graphs. They defined the notion of "λ-extendible" properties Π and generalized the lower bound (1) to tight lower bounds for all such properties; we refer to these lower bounds as the *Poljak-Turzík bound* for Π. Well-known examples of such properties include the bipartite subgraphs, q-colorable subgraphs for fixed q, or acyclic subgraphs of oriented graphs. We will use the slightly different notion of strongly λ-extendible properties [87]. Every strongly λ-extendible property is λ-extendible and every λ-extendible

property we are aware of is also strongly λ-extendible, but it is unclear whether the two concepts are really equivalent.

Let us give now a formal definition. A *graph property* Π is simply a set of graphs. For a graph G, a Π-subgraph is a subgraph of G that belongs to Π. A graph property Π is *hereditary* if for any $G \in \Pi$ also all vertex-induced subgraphs of G belong to Π.

▶ **Definition 2.4.** Let $\mathcal{G}$ be a class of (possibly labelled and/or oriented) graphs and let $\lambda \in (0, 1)$. A (graph) property Π is *strongly λ-extendible* on $\mathcal{G}$ if it satisfies the following properties:

(i) *inclusiveness*: $\{G \in \mathcal{G} \mid \langle G \rangle \in K_1, K_2\} \subseteq \Pi$.
(ii) *block additivity*: $G \in \mathcal{G}$ belongs to Π if and only if each 2-connected component of G belongs to Π.
(iii) *extendibility*: For any $G \in \mathcal{G}$ and any partition $U \uplus W$ of $V(G)$ for which $G[U], G[W] \in \Pi$ there is a set $F \subseteq E(U, W)$ of size $|F| \geq \lambda |E(U, W)|$ for which $G - (E(U, W) \setminus F) \in \Pi$.

The set of all bipartite graphs $\Pi_{\text{bipartite}}$ is a strongly $\frac{1}{2}$-extendible property. Thus, Max-Cut AEE is equivalent to Above Poljak-Turzík Bound($\Pi_{\text{bipartite}}$).

Poljak and Turzík[90] showed that, given a (strongly) λ-extendible property Π, any connected graph G contains a subgraph $H \in \Pi$ with at least $\lambda |E(G)| + \frac{1-\lambda}{2}(|V(G)| - 1)$ edges. We denote this lower bound by $\mathsf{pt}(G)$.

Mnich et al. [87] considered the problem Above Poljak-Turzík(Π) of finding subgraphs in Π with k edges above the *Poljak-Turzík bound* $\mathsf{pt}(G)$; they gave fixed-parameter algorithms for this problem on all strongly λ-extendible properties Π, thereby generalizing the algorithm for Max-Cut. A subclass of these properties, requiring certain technical conditions, was later shown to admit polynomial kernels [26].

2.3.1 Our Contributions

Linear-Time FPT. Our first result shows that the fixed-parameter algorithm by Crowston et al. [23] for the Signed Max-Cut AEE problem can be implemented so as to run in linear time:

▶ **Theorem 2.5.** (Signed) Max-Cut AEE *can be solved in time* $8^k \cdot O(m)$.

Theorem 2.5 considerably improves the earlier running time analysis [23, 25], which shows a running time of $8^k \cdot O(n^4)$. At the same time, our algorithm improves the very involved algorithm by Bollobás and Scott [14] that considers the weaker lower bound $m/2+(\sqrt{8m+1}-1)/8$ instead of (1). Third, Theorem 2.5 generalizes the linear-time algorithm by Ngd*o*c and Tuza [89] for the special case of Max-Cut with $k = 0$. Note that Max-Cut AEE cannot be solved in time $2^{o(k)} \cdot n^{O(1)}$ assuming the Exponential Time Hypothesis [25].

Linear Vertex Kernels. Our second contribution is a kernel with a linear number $O(k)$ of vertices for Max-Cut AEE and its generalization Signed Max-Cut AEE.

▶ **Theorem 2.6.** *The* (Signed) Max-Cut AEE *problem admits a kernel with $O(k)$ vertices, which can be computed in time $O(km)$.*

These results considerably improve the previous best kernel bound of $O(k^3)$ vertices by Crowston et al. [23]. Moreover, the presented kernel completely resolves the asymptotic kernelization complexity of (Signed) Max-Cut AEE, since a kernel with $o(k)$ vertices would again contradict the Exponential-Time Hypothesis, as the Max-Cut problem can

be solved by checking all vertex bipartitions. On top of that, our kernelization is also *fast*. In fact, we only need to compute $O(k)$ DFS/BFS trees. The rest of the algorithm runs in time $O(m)$.

Extensions to Strongly λ-Extendible Properties. As mentioned, the property of graphs having large bipartite subgraphs can be generalized to λ-extendible properties as defined by Poljak and Turzík [90]. For a given λ-extendible property Π, we consider the following problem:

Above Poljak-Turzík(Π)	
Input:	A connected graph G and an integer k.
Question:	Does G have a spanning subgraph $H \in \Pi$ s.t. $\lvert E(H)\rvert \geq \lambda \cdot \lvert E(G)\rvert + \frac{1-\lambda}{2} \cdot (\lvert V(G)\rvert - 1) + k$?

Max-Cut AEE is a special case of this problem with $\lambda = \frac{1}{2}$. Note the slight change in the definition of k compared to (Signed) Max-Cut AEE, where k was divided by $4 = \frac{2}{1-\lambda}$ for $\lambda = \frac{1}{2}$.

Crowston et al. [23] gave polynomial kernels for Above Poljak-Turzík(Π), for all strongly λ-extendible properties Π on possibly oriented and/or labelled graphs satisfying at least one of the following properties:

(P1) $\lambda \neq \frac{1}{2}$; or
(P2) $G \in \Pi$ for all graphs G whose underlying simple graph is K_3; or
(P3) Π is a hereditary property of simple or oriented graphs.

Their kernels have $O(k^3)$ or $O(k^2)$ vertices, depending on the exact problem.

Our third result improves *all* these kernels for strongly λ-extendible properties to asymptotically optimal $O(k)$ vertices:

▶ **Theorem 2.7.** *Let Π be any strongly λ-extendible property of (possibly oriented and/or labelled) graphs satisfying (P1), or (P2), or (P3). Then* Above Poljak-Turzík(Π) *admits a kernel with $O(k)$ vertices, which is computable in time $O(km)$.*

Consequences for Acyclic Subdigraphs. Theorem 2.7 has several applications. For instance, Raman and Saurabh [92] asked for the parameterized complexity of the Max Acyclic Subdigraph problem above the Poljak-Turzík bound: Given a weakly connected oriented graph G on n vertices and m arcs, does it have an acyclic sub-digraph of at least $m/2 + (n-1)/4 + k$ arcs? For this problem, Crowston et al. [24] gave an algorithm with running time $2^{O(k \log k)} \cdot n^{O(1)}$ and showed a kernel with $O(k^2)$ vertices. They explicitly asked whether the kernel size can be improved to $O(k)$ vertices, and whether the running time can be improved to $2^{O(k)} \cdot n^{O(1)}$. Here, we answer their questions in the affirmative by using Theorem 2.7 and then applying an $O^*(2^n)$-time algorithm by Raman and Saurabh [93, Thm. 2] to our kernel with $O(k)$ vertices.

▶ **Corollary 2.8.** Max Acyclic Subdigraph *parameterized above Poljak-Turzík bound admits a kernel with $O(k)$ vertices and can be solved in time $2^{O(k)} \cdot n^{O(1)}$.*

Again, assuming the Exponential Time Hypothesis, the running time of this algorithm is asymptotically optimal.

2.4 Polynomial Kernels for Weighted Problems

The question of handling numerical values is of fundamental importance in computer science. Typical issues are precision, numerical stability, and representation of numbers. We study the effect that the presence of (possibly very large) numbers has on weighted versions of well-studied NP-hard problems in the context of kernelization. In other words, we are interested in the effect of large numbers on the computational complexity of solving hard combinatorial problems.

The issue of handling large weights in kernelization has been brought up again and again as an important open problem [12, 52, 29, 27]. For example, it is well-known that for the task of finding a vertex cover of at most k vertices for a given unweighted graph G one can efficiently compute an equivalent instance (G', k') such that G' has at most $2k$ vertices. Unfortunately, when the vertices of G are additionally equipped with positive rational weights and the chosen vertex cover needs to obey some specified maximum weight $W \in \mathbb{Q}$ then it was long unknown how to encode (and shrink) the vertex weights to bitsize polynomial in k. In this direction, Cheblík and Cheblíková [19] showed that an equivalent graph G' with total vertex weight at most $2w^*$ can be obtained in polynomial time, whereby w^* denotes the minimum weight of a vertex cover of G. This, however, does not mean that the size of G' is bounded, unless one makes the additional assumption that the vertex weights are bounded from below; consequently, their method only yields a kernel with that extra requirement of vertex weights being bounded away from zero. In contrast, we do not make such an assumption.

Let us attempt to clarify the issue some more. The task of finding a polynomial kernelization for a weighted problem usually comes down to two parts: (1) Deriving reduction rules that work correctly in the presence of weights. The goal, as for unweighted problems, is to reduce the number of relevant objects, e.g., vertices, edges, sets, etc., to polynomial in the parameter. (2) Shrinking or replacing the weights of remaining objects such that their encoding size becomes (at worst) polynomial in the parameter. The former part usually benefits from existing literature on kernels of unweighted problems, but regarding the latter only little progress was made.

For a pure weight reduction question let us consider the Subset Sum problem. Therein we are given n numbers $a_1, \ldots, a_n \in \mathbb{N}$ and a target value $b \in \mathbb{N}$ and we have to determine whether some subset of the n numbers has sum exactly b. Clearly, reducing such an instance to size polynomial in n hinges on the ability of handling large numbers a_i and b. Let us recall that a straightforward dynamic program solves Subset Sum in time $\mathcal{O}(nb)$, implying that large weights are to be expected in hard instances. Harnik and Naor [65] showed that taking all numbers modulo a sufficiently large random prime p of magnitude about 2^{2n} produces an equivalent instance with error probability exponentially small in n. (Note that the obtained instance is with respect to arithmetic modulo p.) The total bitsize then becomes $\mathcal{O}(n^2)$. Unfortunately, this elegant approach fails for more complicated problems than Subset Sum.

Consider the Subset Range Sum variant of Subset Sum where we are given not a single target value b but instead a lower bound L and an upper bound U with the task of finding a subset with sum in the interval $\{L, \ldots, U\}$. Observe that taking the values a_i modulo a large random prime faces the problem of specifying the new target value(s), in particular if $U - L > p$ because then every remainder modulo p is possible for the solution. Nederlof et al. [88] circumvented this issue by creating not one but in fact a polynomial number of small instances. Intuitively, if a solution has value close to either L or U then the randomized approach will work well (possibly making a separate instance for target values close to L or U). For solutions sufficiently far from L or U there is no harm in losing a little

precision and dividing all numbers by 2; then the argument iterates. Overall, because the number of iterations is bounded by the logarithm of the numbers (i.e., their encoding size), this creates a number of instances that is polynomial in the input size, with each instance having size $\mathcal{O}(n^2)$; if the initial input is "yes" then at least one of the created instances is "yes" (this is usually called a (disjunctive) Turing kernelization).

To our knowledge, the mentioned results are the only positive results that are aimed directly at the issue of handling large numbers in the context of kernelization. Apart from these, there are of course results where the chosen parameter bounds the variety of feasible weights and values, but this only applies to integer domains; e.g., it is easy to find a kernel for Weighted Vertex Cover when all weights are positive integers and the parameter is the maximum total weight k. On the negative side, there are a couple of lower bounds that rule out polynomial kernelizations for various weighted and ILP problems [13, 77]. Note, however, that the lower bounds appear to "abuse" large weights in order to build gadgets for lower bound proofs that also include a super-polynomial number of objects as opposed to having just few objects with weights of super-polynomial encoding size. In other words, the known lower bounds pertain rather to the first step, i.e. finding reduction rules that work correctly in the presence of weights, than to the inherent complexity of the numbers themselves. Accordingly, since 2010 the question for a deterministic polynomial kernelization for Subset Sum or Knapsack with respect to the number of items can be found among open problems in kernelization [12, 52, 29, 27].

Recently, Marx and Végh [86] gave a polynomial kernelization for a weighted connectivity augmentation problem. As a crucial step, they use a technique of Frank and Tardos [54], originally aimed at obtaining strongly polynomial-time algorithms, to replace rational weights by sufficiently small and equivalent integer weights. They observe and point out that this might be a useful tool to handle in general the question of getting kernelizations for weighted versions of parameterized problems. It turns out that, more strongly, Frank and Tardos' result can also be used to settle the mentioned open problems regarding Knapsack and Subset Sum. We point out that this is a somewhat circular statement since Frank and Tardos had set out to, amongst others, improve existing algorithms for ILPs, which could be seen as *very general* weighted problems.

2.4.1 Our Contributions

We use the theorem of Frank and Tardos [54] to formally settle the open problems, i.e., we obtain deterministic kernelizations for Subset Sum(n) and Knapsack(n). Generally, in the spirit of Marx and Végh's observation, this allows to get polynomial kernelizations whenever one is able to first reduce the *number of objects*, e.g., vertices or edges, to polynomial in the parameter. The theorem can then be used to sufficiently shrink the weights of all objects such that the *total size* becomes polynomial in the parameter.

Motivated by this, we consider weighted versions of several well-studied parameterized problems, e.g., d-Hitting Set, d-Set Packing, and Max Cut, and show how to reduce the number of relevant structures to polynomial in the parameter. An application of Frank and Tardos' result then implies polynomial kernelizations.

Next, we consider the Knapsack problem and its special case Subset Sum. For Subset Sum instances with only k item sizes, we derive a kernel of size polynomial in k. This way, we are improving the exponential-size kernel for this problem due to Fellows et al. [51]. We also extend the work of Fellows et al. in another direction by showing that the more general Knapsack problem is fixed-parameter tractable (i.e., has an exponential kernel) when parameterized by the number k of item sizes, even for unbounded number of item

values. On the other hand, we provide quadratic kernel size lower bounds for general Subset Sum instances assuming the Exponential Time Hypothesis [69].

Finally, as a possible tool for future kernelization results we show that the weight reduction approach also carries over to polynomial ILPs so long as the maximum degree and the domains of variables are sufficiently small.

3 Max-Cut on General Graphs

We start the main part of this thesis with the analysis of local search for the Max-Cut problem on general graphs.

3.1 Outline of the Analysis

As mentioned in the introduction, there are already a few results on smoothed analysis of local search. There is, however, one fundamental difference between our analysis and the previous ones. The previous analyses for the 2-Opt heuristic, the ICP algorithm, the k-means method, and the Max-Cut Problem for graphs of logarithmic degree all rely on the observation that on smoothed inputs with high probability for every locally non-optimal solution every available local improvement results in a significant change of the potential function. Together with bounds on the minimal and maximal potential value, this implies that in expectation there cannot be too many local improvements.[1]

For the Max-Cut Problem the situation is different because even on smoothed inputs it is very likely that there exists a locally non-optimal cut that allows local improvements that cause an exponentially small change of the potential. It is even likely that there exist longer sequences of consecutive local improvements that all cause only a small change of the potential. The reason for this is the large number of possible cuts. While for any fixed cut it is unlikely that there exists an exponentially small local improvement, it is very likely that for one of the exponentially many cuts there exists such an improvement.

On first glance the situation for the other problems is no different. There is, for example, also an exponential number of different TSP tours. However, for determining the amount by which the length of the tour decreases by a particular 2-Opt step, only the lengths of the four involved edges are important and there is only a polynomial number of choices for these four edges. If, on the other hand, one node changes its side in the Max-Cut Problem, to determine the improvement one needs to know the configuration of all nodes in its neighborhood. If the degree of the graph is at most logarithmic, there is only a polynomial number of such configurations. In fact, this is a crucial ingredient of the smoothed polynomial bound by [37]. However, in general there is an exponential number of configurations.

As it is not sufficient anymore to analyze the potential change of a single step, our analysis is based on considering longer sequences of ℓ consecutive steps for an appropriately chosen ℓ. Let us denote by Δ the smallest improvement made by any sequence of ℓ consecutive local improvements for any initial cut. In order to analyze Δ, one could try to use a union bound over all choices for the initial cut and the sequence of ℓ steps. There are at most $2^n n^\ell$ such choices. Let us assume that an initial cut and a sequence of ℓ consecutive steps are given. Then we get a system of ℓ linear combinations of edge weights that describe the potential increases that are caused by the ℓ steps. Each such linear combination has the form $\sum_{e \in E} \lambda_e w(e)$ for some $\lambda_e \in \{-1, 0, 1\}$, where λ_e is 1 for edges joining the cut, -1 for edges leaving the cut, and 0 for the other edges. For $\varepsilon > 0$, we would like to bound the probability that all these linear combinations simultaneously take values in the interval $(0, \varepsilon]$, that is, they are all improvements by at most ε. A result from Röglin [94] implies that this probability can be bounded from above by $(\varepsilon\phi)^r$ where r denotes the rank of the set of linear combinations.

[1] To be more precise, in one case of the analysis of the k-means method, one needs to consider three consecutive local improvements to gain a significant change of the potential function. Also in the analysis of the 2-Opt algorithm two steps are considered in order to improve the degree of the polynomial.

If we could argue that for any initial cut and any sequence of length ℓ the rank is at least $\alpha\ell$ for some constant $\alpha > 0$, then a union bound would yield that $\mathbf{Pr}[\Delta \leq \varepsilon] \leq 2^n(n\phi^\alpha\varepsilon^\alpha)^\ell$. Choosing $\ell = n$, this would even be sufficient to prove an improved version of Theorem 2.1 with only polynomial dependence on n. The problem is, however, that we cannot guarantee that for any sequence of length n the rank is $\Omega(n)$. Indeed there are sequences of length n in which only few different nodes move multiple times such that the rank is only polylogarithmic in n. Hence with this approach Theorem 2.1 cannot be proved.

In order to reduce the factor 2^n in the union bound, we make use of the following observation: Consider a node v that moves at least twice and take the linear combination L obtained by adding up the linear combinations belonging to two consecutive moves of node v. As after these two moves node v is in its original partition again, L contains only weights belonging to edges between v and other nodes that have moved an odd number of times between the two moves of node v. Therefore we only need to fix the configuration of these active nodes, which reduces the factor 2^n in the union bound to 2^ℓ. While this is no improvement for $\ell \geq n$, it proves valuable when we consider subsequences of smaller length. If both moves of node v yield an improvement in $(0, \varepsilon]$, then L takes a value in $(0, 2\varepsilon]$.

We call a sequence of length ℓ a *k-repeating sequence* if at least $\lceil \ell/k \rceil$ different nodes move at least twice. Given a k-repeating sequence of length ℓ, we argue that the rank of the set of linear combinations constructed in the above way is at least $\lceil \ell/(2k) \rceil$. One can then show that with high probability any k-repeating sequence yields an improvement in the order of $1/(\phi n^{\Theta(k)})$. Then we argue that any sequence of $5n$ consecutive improvements must contain a subsequence that is $\Theta(\log n)$-repeating. Together this implies Theorem 2.1.

Let us make one remark about the number 2^n of initial cuts that we have to consider. One might be tempted to conjecture that the factor 2^n in the union bound can be avoided if only the cut the FLIP algorithm starts with is considered instead of every possible cut. However, then our analysis would not be possible anymore. We break the sequence of steps of the FLIP algorithm into subsequences of length $5n$ each and argue that each such subsequence yields a significant improvement. Hence, not only the initial cut the FLIP algorithm starts with needs to be considered but also the initial cut of each of these subsequences.

3.2 Analysis

Our goal is to show that each sequence of $5n$ consecutive steps yields a big improvement with high probability. Throughout the analysis, we need a parameter k. For simplicity, we directly set it to $k = \lceil 5 \log_2 n \rceil$, which is the value needed in Lemma 3.5. We would like to point out that Lemma 3.2 and Lemma 3.4 also hold for general k.

▶ **Definition 3.1.** We call a sequence of $\ell \in \mathbb{N}$ consecutive steps *k-repeating* if at least $\lceil \ell/k \rceil$ different nodes move at least twice in that sequence.

As already explained in subsection 3.1, for each two consecutive moves of a node we can obtain a linear combination that only contains edges to active nodes. To be more precise, consider an arbitrary initial cut and an arbitrary sequence of moves in which a node v moves at least twice. Now consider two moves of v that are consecutive in the sense that v does not move in between these two moves. The improvements of these two moves can be expressed as linear combinations

$$\sum_{e=\{v,w\}\in E} \lambda_e \cdot w(e) \quad \text{and} \quad \sum_{e=\{v,w\}\in E} \mu_e \cdot w(e)$$

with $\lambda_e, \mu_e \in \{-1, 1\}$. In the sum of these linear combinations, the weight $w(e)$ of an edge $e = \{v, w\}$ cancels out if and only if node w moves not at all or an even number of times

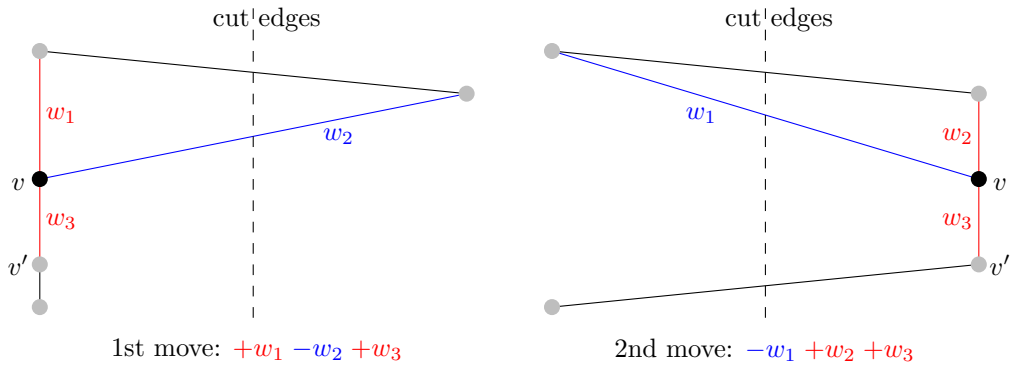


Figure 1 Example for the linear combination L_v for a node v: The left hand side and right hand side depict the partition of the vertices before the first and the second move of v, respectively. Through the first move of v, the edges with weights w_1 and w_3 enter the cut, while the edge with weight w_2 leaves the cut. Through the second move of node v, the edges with weights w_2 and w_3 enter the cut, while the edge with weight w_1 leaves the cut. Adding the two corresponding linear combinations results in $L_v = (w_1 - w_2 + w_3) + (-w_1 + w_2 + w_3) = 2w_3$, where w_3 is the weight of the edge $\{v, v'\}$ and v' is the only neighbor of v which moved an odd number of times in the meantime.

between the two considered moves of node v. Hence, the sum L_v of these linear combinations contains only weights of edges $e = \{v, w\}$ for which node w moves at least once between the moves of node v. Figure 1 illustrates this observation in an easy example.

We first show a lower bound for the rank of the set of these linear combinations.

▶ **Lemma 3.2.** *Let S be a k-repeating sequence of length ℓ with an arbitrary starting configuration and let $D \subseteq V$ be the set of vertices that move at least twice in the sequence S. For each node $v \in D$, let L_v denote the sum of the linear combinations corresponding to the first two moves of v. The rank of the set $\{L_v : v \in D\}$ of linear combinations is at least $\lceil \ell/(2k) \rceil$.*

Proof. We construct an auxiliary directed graph $G' = (V, E')$ in the following way: For any L_v, let W_v be the set of vertices that move an odd number of times between the first two moves of v, i.e., L_v contains exactly the weights of edges between v and the nodes from W_v. This set cannot be empty for any $v \in D$ because otherwise the configuration before the first move of v would be equal to the configuration after the second move, contradicting the fact that configurations cannot repeat in any sequence of improving steps. For any v that moves at most once, set $W_v = \emptyset$. Let $E' = \{(w, v) : v \in V, w \in W_v\}$.

We want to find a set $I \subseteq D$ with $|I| \geq |D|/2$ such that the linear combinations $\{L_v : v \in I\}$ are linearly independent. We call a vertex $w \in W_v$ a *witness* for $v \in I$ if $w \notin I$ or $v \notin W_w$. If w is a witness for $v \in I$, then the edge $\{v, w\}$ only occurs in L_v but not in another $L_{v'}, v' \in I \setminus \{v\}$ (it could only occur in L_w, but this is forbidden by the definition of witnesses). Hence, if every vertex in I has a witness, then the linear combinations $\{L_v : v \in I\}$ must be linearly independent.

We start with $I = \emptyset$. Compute a BFS arborescence B in G' rooted at an arbitrary vertex $r \in V(G')$, and let V' be the vertex set of B. Define the bipartition $V^0 \dot\cup V^1$ of V' as the sets of vertices whose (unique) path from r in B contains an even or odd number of edges, respectively. Add the bigger of the two sets $V^0 \cap D$ and $V^1 \cap D$ to I. Remove V' from G' and repeat until G' is the empty graph.

In every iteration, every vertex v added to I has a witness: If v is not the root r of the arborescence, its predecessor w on the path from r to v is a witness for v as it is in W_v but

not in I. If $v = r$, then r belongs to D, i.e., W_r is non-empty. No edge $(w, r), w \in V$, has been deleted in a former iteration as otherwise r would have belonged to the same former arborescence as w. Hence, there is an edge (w, r) in G'. Then w is a witness: If $r \in W_w$ then $w \in V^1$ and hence w will deleted and not added to I. Otherwise $r \notin W_w$ and w is a witness for r by definition.

As S is k-repeating, there are at least $\lceil \ell/k \rceil$ nodes in D. Therefore we obtain a set I with size $|I| \geq \lceil \ell/k \rceil/2$, i.e., $|I| \geq \lceil \ell/(2k) \rceil$ because $|I|$ is integral. This yields the lemma. ◀

For the next lemma we need a probability result by Röglin [94]. For completeness, we state a simpler, sufficient version of this result and prove it.

▶ **Lemma 3.3** (Röglin [94]). *Let $X_1, \ldots, X_n$ be independent real random variables with density bounded by ϕ. Let $\lambda^1, \ldots, \lambda^k \in \mathbb{Z}^n$ be linearly independent vectors. For $i \in \{1, \ldots, k\}$ and fixed $\varepsilon \geq 0$, we denote by $\mathcal{A}_i$ the event that $\lambda^i \cdot X$ takes a value in $[0, \varepsilon]$, where X denotes the vector $X = (X_1, \ldots, X_n)$. Then*

$$\mathbf{Pr}\left[\bigcap_{i=1}^{k} \mathcal{A}_i\right] \leq (\varepsilon\phi)^k.$$

Proof. For $1 \leq i \leq n$, let $f_i \colon \mathbb{R} \to [0, \phi]$ denote the density of X_i. The main tool for proving the lemma is a change of variables. Instead of using the canonical basis of the n-dimensional vector space $\mathbb{R}^n$, we use the given linear combinations as basis vectors. To be more precise, the basis $\mathcal{B}$ that we use consists of two parts: it contains the vectors $\lambda^1, \ldots, \lambda^k$ and it is completed by some vectors from the canonical basis $\{e^1, \ldots, e^n\}$, where e^i denotes the i-th canonical row vector, i.e., $e^i_i = 1$ and $e^i_j = 0$ for $j \neq i$. Without loss of generality, we assume that $\mathcal{B} = \{\lambda^1, \ldots, \lambda^k, e^{k+1}, \ldots, e^n\}$.

Let $A = (\lambda^1, \ldots, \lambda^k, e^{k+1}, \ldots, e^n)^T$ and let $\Phi \colon \mathbb{R}^n \to \mathbb{R}^n$ be defined by $\Phi(x) = Ax$. Since $\mathcal{B}$ is a basis, the function Φ is a diffeomorphism. We define the vector $Y = (Y_1, \ldots, Y_n)$ as $Y = \Phi(X)$. Let $f \colon \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ denote the joint density of the entries of X, and let $g \colon \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ denote the joint density of the entries of Y. We can express the joint density g as

$$g(y_1, \ldots, y_n) = |\det_{\partial} \Phi^{-1}| \cdot f(\Phi^{-1}(y_1, \ldots, y_n)),$$

where $\det_{\partial} \Phi^{-1}$ denotes the determinant of the Jacobian matrix of Φ^{-1}.

The matrix A is invertible as $\mathcal{B}$ is a basis of $\mathbb{R}^n$. Hence, for $y \in \mathbb{R}^n$, $\Phi^{-1}(y) = A^{-1}y$ and the Jacobian matrix of Φ^{-1} equals A^{-1}. Thus, $\det_{\partial} \Phi^{-1} = \det A^{-1} = (\det A)^{-1}$. Since all entries of A are integers, also its determinant must be an integer, and since it is invertible, we know that $\det A \neq 0$. Hence, $|\det A| \geq 1$ and $|\det A^{-1}| \leq 1$. For $y \in \mathbb{R}^n$, we decompose $\Phi^{-1}(y) \in \mathbb{R}^n$ into $\Phi^{-1}(y) = (\Phi_1^{-1}(y), \ldots, \Phi_n^{-1}(y))$. Due to the independence of the random vectors $X_1, \ldots, X_n$, we have $f(x_1, \ldots, x_n) = f_1(x_1) \cdot \ldots \cdot f_n(x_n)$. This yields

$$\begin{aligned} g(y) &\leq f(\Phi^{-1}(y)) = f_1(\Phi_1^{-1}(y)) \cdot \ldots \cdot f_n(\Phi_n^{-1}(y)) \leq \phi^k \cdot f_{k+1}(\Phi_{k+1}^{-1}(y)) \cdot \ldots \cdot f_n(\Phi_n^{-1}(y)) \\ &\leq \phi^k \cdot f_{k+1}(y_{k+1}) \cdot \ldots \cdot f_n(y_n) \end{aligned}$$

as $f_1, \ldots, f_k$ are bounded from above by ϕ and the i-th row of A is e^i for $k < i \leq n$. Hence,

the probability we want to estimate can be upper bounded by

$$\begin{aligned}\mathbf{Pr}\left[\bigcap_{i=1}^{k}\mathcal{A}_i\right] &= \int_{(y_1,\dots,y_n)\in[0,\varepsilon]^k\times\mathbb{R}^{n-k}} g(y_1,\dots,y_n)\,d(y_1,\dots,y_n)\\ &\leq (\varepsilon\phi)^k\cdot\prod_{i=k+1}^{n}\int_{\mathbb{R}} f_i(y_i)\,dy_i = (\varepsilon\phi)^k,\end{aligned}$$

where the last equation follows because $f_{k+1},\dots,f_n$ are density functions. ◀

▶ **Lemma 3.4.** *Denote by $\Delta(\ell)$ the smallest improvement made, for any starting configuration, by any k-repeating sequence of length ℓ in which every step increases the potential (and set $\Delta(\ell)=\infty$ if not such sequence exists). Then for any $\varepsilon>0$,*

$$\mathbf{Pr}\left[\Delta(\ell)\leq\varepsilon\right]\leq(2n)^{\ell}(2\phi\varepsilon)^{\lceil\ell/(2k)\rceil}.$$

Proof. We first fix a k-repeating sequence of length ℓ. As there are ℓ steps in this sequence, there are at most n^ℓ choices for the sequence. We will use a union bound over all these n^ℓ choices and over all possible starting configurations of the nodes that are active in the sequence. This gives the additional factor of 2^ℓ because in the considered sequence at most ℓ nodes can move.

For a fixed starting configuration and a fixed sequence, we consider a node v that moves at least twice and the linear combinations L_1 and L_2 that correspond to two consecutive moves of node v. As after these two moves node v is in its original partition again, the sum $L=L_1+L_2$ contains only weights belonging to edges between v and other nodes that have moved an odd number of times between the two moves of node v. In particular, L contains only weights belonging to edges between active nodes, for which we fixed the starting configuration.

Only if $L\in(0,2\varepsilon]$, both L_1 and L_2 can take values in $(0,\varepsilon]$. Hence it suffices to bound the probability that $L\in(0,2\varepsilon]$. Due to Lemma 3.2, the rank of the set of all linear combinations constructed like L is at least $\lceil\ell/(2k)\rceil$. We can apply Lemma 3.3 to obtain a bound of $(2\varepsilon\phi)^{\lceil\ell/(2k)\rceil}$ for the probability that all these linear combinations take values in $(0,2\varepsilon]$. Together with the union bound this proves the claimed bound on $\Delta(\ell)$. ◀

▶ **Lemma 3.5.** *Denote by $\Delta:=\min_{1\leq\ell\leq5n}\Delta(\ell)$ the minimum improvement made, for any starting configuration, by any k-repeating sequence of length at most $5n$ in which every step increases the potential. Then Δ is a lower bound for the improvement made by any sequence of $5n$ steps.*

This holds due to the fact that every sequence of $5n$ steps contains a k-repeating subsequence, which we will prove later. Based on this lemma, we can prove Theorem 2.1 by showing that, with high probability, Δ is significantly bounded away from zero.

Proof of Theorem 2.1. Let T be the number of steps of the FLIP algorithm divided by $5n$. As every cut contains fewer than n^2 edges and every edge weight is in the interval $[-1,1]$, the weight of every cut is in $[-n^2,n^2]$. If the minimum improvement of any sequence of length $5n$ is at least Δ, then the number of steps is bounded by $5n\cdot T\leq 5n\cdot 2n^2/\Delta$, i.e., $\mathbf{Pr}\left[T\geq t\right]\leq\mathbf{Pr}\left[\Delta\leq 2n^2/t\right]$ for every $t>0$.

To simplify the notation, let $\zeta = 4\phi n^2(2n)^{2k} = \phi \cdot n^{O(\log n)}$. For $i \geq 2$, let $t_i = \zeta i$. By Lemma 3.4 we know that for every $i \geq 2$,

$$\begin{aligned}\mathbf{Pr}\left[T \geq t_i\right] &\leq \mathbf{Pr}\left[\Delta \leq \frac{2n^2}{t_i}\right] \leq \sum_{\ell=1}^{5n} \mathbf{Pr}\left[\Delta(\ell) \leq \frac{2n^2}{t_i}\right] \leq \sum_{\ell=1}^{5n} (2n)^\ell \left(2\phi \frac{2n^2}{\zeta i}\right)^{\lceil \ell/(2k)\rceil} \\ &\leq \sum_{\ell=1}^{5n} \left((2n)^{2k} \cdot \frac{4\phi n^2}{\zeta i}\right)^{\lceil \ell/(2k)\rceil} = \sum_{\ell=1}^{5n} \left(\frac{1}{i}\right)^{\lceil \ell/(2k)\rceil} \leq \sum_{\ell'=0}^{\infty} 2k \left(\frac{1}{i}\right)^{\ell'} - 2k \\ &= 2k\left(\frac{1}{1-1/i} - 1\right) = \frac{2k}{i-1}.\end{aligned}$$

The bound $T \leq 2^n$ is trivial as no configuration of the nodes can occur twice. Together with $t_{i+1} - t_i = \zeta$, we obtain

$$\begin{aligned}\mathbf{E}\left[T\right] &= \sum_{t=1}^{2^n} \mathbf{Pr}\left[T \geq t\right] \leq 2\zeta + \sum_{i=2}^{2^n} \sum_{t=t_i}^{t_{i+1}-1} \mathbf{Pr}\left[T \geq t\right] \leq 2\zeta + \sum_{i=2}^{2^n} \sum_{t=t_i}^{t_{i+1}-1} \mathbf{Pr}\left[T \geq t_i\right] \\ &\leq 2\zeta + \sum_{i=2}^{2^n} \zeta \cdot \frac{2k}{i-1} \leq \zeta(2 + 2k(\log(2^n)+1)) = \zeta \cdot O(n \log n) = \phi \cdot n^{O(\log n)}. \qquad \blacktriangleleft\end{aligned}$$

What remains to show is Lemma 3.5, i.e., Δ is a lower bound for the minimum improvement of any sequence of length $5n$. Assume that this is not the case. Then there is a sequence of length $5n$ that does not contain any k-repeating subsequence. We show that this is not possible because then more than n nodes would have to move in that sequence.

▶ **Definition 3.6.** We call a sequence $A_1, \ldots, A_q$ of sets a *non-k-repeating block sequence of length ℓ* if the following conditions hold.

(i) For every $1 \leq i < q$, $|A_i| = k$.
(ii) $1 \leq |A_q| \leq k$.
(iii) $\sum_{i=1}^{q} |A_i| = \ell$.
(iv) For every i and j with $i \leq j$, the number of elements that are contained in at least two sets from $A_i, \ldots, A_j$ is at most $j - i$.

We denote by $n_k(\ell)$ the cardinality of $A_1 \cup \ldots \cup A_q$ minimized over all non-k-repeating block sequences of length ℓ.

Proof of Lemma 3.5. Any sequence S of length ℓ that does not contain a k-repeating subsequence corresponds to a non-k-repeating block sequence $A_1, \ldots, A_q$ of length ℓ with $q = \lceil \ell/k \rceil$ blocks: Subdivide S into subsequences $S_1, \ldots, S_q$ of length k (except for S_q, which can be shorter) and put the nodes occurring in S_i into A_i. Because S does not contain a k-repeating subsequence of length k, no node can occur twice in a subsequence S_i. For $1 \leq i < j \leq q$, the subsequence $S_i, \ldots, S_j$ contains at most $j - i$ nodes which occur multiple times because S does not contain a k-repeating subsequence of length $(j - i + 1) \cdot k$.

Hence, it suffices to show that there is no non-k-repeating block sequence of length $5n$ with at most n distinct elements. In other words, $n_k(5n) > n$. If $n \leq 3$, then there are at least two blocks as $5n > k$, but $k = \lceil 5 \log_2 n \rceil > n$ such that the first condition of Definition 3.6 cannot be satisfied under the assumption $n_k(\ell) \leq n$. Therefore we can assume $n \geq 4$.

Let $q = \lceil 5n/k \rceil$ and let $A_1, \ldots, A_q$ be a non-k-repeating block sequence of length $5n$ with exactly $n_k(5n)$ different elements $x_1, \ldots, x_{n_k(5n)}$ contained in $A_1 \cup \ldots \cup A_q$. Construct an auxiliary graph H as follows: Introduce a vertex $i \in V(H)$ for each set A_i. For an

element x_i, let $\rho(i)$ be the number of different sets that contain x_i and let $A_{i_1}, \ldots, A_{i_{\rho(i)}}$ ($i_1 < \ldots < i_{\rho(i)}$) be these sets. Define $Q_i = \{\{i_j, i_{j+1}\} \colon 1 \leq j < \rho(i)\}$. The edges in Q_i connect neighbored occurrences of the element x_i. Define $E(H) = \dot{\bigcup}_{i=1}^{n_k(5n)} Q_i$ as the disjoint union of these edge sets. Note that we allow parallel edges. With every edge in H, the number of different elements needed in the sequence decreases by exactly 1. Therefore, $n_k(5n) = \sum_{i=1}^{q} |A_i| - |E(H)| = 5n - |E(H)|$, i.e., it suffices to show that $|E(H)|$ is strictly less than $4n$.

Define the *length* of an edge $\{v, w\}$ as $|w - v|$. Now we group the edges by their lengths: For $1 \leq i \leq \lceil \log q \rceil$, let $E_i = \{\{v, w\} \in E(H) \colon 2^{i-1} \leq |w - v| \leq 2^i\}$. Furthermore, we define *cuts* $S_j = \{\{v, w\} \in E(H) \colon v \leq j < w\}$ for every $1 \leq j < q$.

For a cut S_j and some E_i, consider an arbitrary edge $\{v, w\} \in S_j \cap E_i$: As this edge has a length of at most 2^i, we know that $j - 2^i < v, w \leq j + 2^i$. Because $A_1, \ldots, A_q$ is a non-k-repeating block sequence, there can only be at most $j + 2^i - (j - 2^i) \leq 2^{i+1}$ elements that occur multiple times in $A_{\max\{j-2^i,1\}}, \ldots, A_{\min\{j+2^i,q\}}$. By construction, every element can generate at most one edge in S_j. Hence $|S_j \cap E_i| \leq 2^{i+1}$.

On the other hand, every edge in E_i has a length of at least 2^{i-1}. Therefore every edge in E_i occurs in at least 2^{i-1} cuts. Thus we can bound the cardinality of E_i by

$$\begin{aligned} |E_i| &\leq \frac{1}{2^{i-1}} \sum_{j=1}^{q-1} |S_j \cap E_i| \leq \frac{1}{2^{i-1}} \sum_{j=1}^{q-1} 2^{i+1} \leq 4(q-1) \\ &= 4 \left(\left\lceil \frac{5n}{\lceil 5 \log n \rceil} \right\rceil - 1 \right) < \frac{4n}{\log n}. \end{aligned}$$

As the union of the E_i is a covering of $E(H)$, we can bound the total number of edges by

$$|E(H)| \leq \sum_{i=1}^{\lceil \log q \rceil} |E_i| < \frac{4n}{\log n} \lceil \log q \rceil = \frac{4n}{\log n} \left\lceil \log \left\lceil \frac{5n}{k} \right\rceil \right\rceil \leq 4n,$$

where the last inequality stems from

$$\left\lceil \log \left\lceil \frac{5n}{k} \right\rceil \right\rceil \leq \log \left(\frac{5n}{k} \right) + 1 \leq \log \left(\frac{5n}{\lceil 5 \log n \rceil} \right) + \log 2 \leq \log \left(\frac{2n}{\log n} \right) \leq \log n$$

for $n \geq 4$. This concludes the proof. ◀

4 Max-Cut with Squared Euclidean Distances

We continue the analysis of local search for Max-Cut with the special case of complete graphs for which the edge weights are given by squared Euclidean distances between the graph vertices. In Section 4.1 we will present the main idea of our proof of Theorem 2.2. Section 4.2 contains preliminary observations and simple probability lemmas. We then use Section 4.3 and Section 4.4 to prove Theorem 2.2 and Theorem 2.3, respectively.

4.1 Outline of the Analysis

We will now give an outline of the main ideas necessary to prove Theorem 2.2. For simplicity, assume for now that the dimension d is constant so we can speak about polynomial bounds despite having factors $2^{O(d)}$. Theorem 2.2 is as well as all other results on smoothed analysis of local search algorithms based on finding a lower bound for the improvement made by any local improvement or any sequence of consecutive local improvements of a certain length. Since the mean values of our points are in $[0,1]^d$, the value of any cut is bounded polynomially in n with high probability. Hence, proving that in any local improvement or in any sequence of $\text{poly}(n)$ consecutive local improvements the value of the cut increases by at least $\varepsilon := 1/\text{poly}(n)$ with high probability suffices for proving that the expected number of local improvements is polynomially bounded. We will call an improvement of at least ε *significant* in the following. On the other hand, we call a pair of a configuration C of V (i.e., a partition of V into two parts) and a point $x \in V$ *bad* if flipping x in the configuration C yields an insignificant improvement.

In the proof of Theorem 2.1 the union bound fixes only the configuration of the active vertices of the considered sequence. The configuration of the passive vertices is not fixed in the union bound. By adding two flips of an active point, it sufficed to know their configuration to compute the probability that the considered sequence of steps was bad, because the influence of the passive points canceled out.

In our analysis we also fix only the configuration of the active vertices. The difference is that we do not combine two improvements of the same vertex anymore and thus the passive vertices are not irrelevant because their configuration has a very essential impact on the improvements made by the flips in the considered sequences. The crucial idea is that we can rewrite the improvements in such a way that we do not need to know the exact configuration and placement of the passive points, and that certain partial information is enough to bound the probability that all steps in the sequence are bad. We can guess this partial information closely enough such that we can approximately compute the improvements while having few enough possibilities for our guesses such that the union bound stays small enough.

Let us go into more detail. Remember that we consider complete graphs in which each vertex is a point in $\mathbb{R}^d$ and the weights of the edges are given by squared Euclidean distances. Our goal is to show that in this setting with high probability there is no sequence in which $\ell = O(d)$ different vertices flip making only insignificant local improvements. Observe that the length of such a sequence is at most 2^ℓ as otherwise one configuration would repeat, which is not possible since the value of the cut increases with every flip. We apply a union bound over all such sequences and over all configurations of the active points.

With only the information about the sequence and the configuration of the active points, it is not possible to determine linear combinations of the edge weights that describe the improvements made in the sequence because the configuration of the passive points is unknown. Assume that the point set $\mathcal{X}$ is partitioned into the sets P_1 and P_{-1}. We can rewrite the improvement made by flipping a point such that we only need to know the value

$|P_1|$ as well as $c := \sum_{x\in P_1} x - \sum_{x\in P_{-1}} x$ and $\xi := \sum_{x\in P_1} \|x\|^2 - \sum_{x\in P_{-1}} \|x\|^2$. These values are unknown if the configuration of the passive points is unknown and we have to assume that they are chosen adversarially.

Let us assume that the standard deviation σ is small. Then the whole point set $\mathcal{X}$ is contained in $[-2,2]^d$ with high probability. This gives us bounding boxes for c and ξ. Hence, we can guess $|P_1|$, c, and ξ up to an ε error, which causes a factor of roughly $(nd/\varepsilon)^d$ in the union bound. We then substitute c and ξ by our guesses in the formulas describing the improvements of steps. This results in formulas which do not depend on the passive points anymore and are good approximations for the improvements of the active points. We use these formulas to argue that it is unlikely that the first step of every active point yields an improvement of at most ε without having to use a union bound over the configuration of the passive points. (This approach is remotely inspired by the analysis of the k-means method where approximate centers of clusters are used [4].)

In our analysis we crucially rely on the fact that the edge weights are given by squared Euclidean distances because for other distance measures the necessary information about the configuration of the passive points is not captured solely by $|P_1|, c$, and ξ.

4.2 Preliminaries and Notation

In this subsection we state some lemmas that we will use later to prove Theorem 2.2 and we introduce some notation. Throughout the section, ε denotes the threshold value between an insignificant and a significant step. We use the notation $X \sim \mathcal{N}(\mu, \sigma^2)$ to indicate that X is a Gaussian random variable with mean μ and standard deviation σ. Up to our proof of the main result, we assume without further mention that $\sigma \le 1/\sqrt{2n}$. Furthermore, we may assume $n \ge d$ because every instance with n points in dimension $d \ge n$ can simply be projected to an n-dimensional subspace of $\mathbb{R}^d$. We will often identify the points with their indices $\{1,\dots,n\} =: N$. A *configuration* of the points is a map $\pi\colon N \to \{-1,1\}$. We often use the shortcut π_v to denote $\pi(v)$.

▶ **Lemma 4.1.** *Let $D_{\max} := \sigma\sqrt{2n}+1$ and let $\mathcal{X}$ be a set of n Gaussian random vectors in $\mathbb{R}^d$ with mean values in $[0,1]^n$ and standard deviation σ. Let $\mathcal{F}_1$ be the event $\mathcal{X} \not\subseteq [-D_{\max}, D_{\max}]^d$. Then $\mathbf{Pr}[\mathcal{F}_1] \le d/2^n$.*

For the proof of Lemma 4.1, we use the following well-known fact.

▶ Claim 4.2. Let $X \sim \mathcal{N}(0,1)$ be a one-dimensional Gaussian random variable. Then $\mathbf{Pr}[|X| > x] \le \frac{\sqrt{2}\cdot\exp(-x^2/2)}{\sqrt{\pi}\cdot x}$ for all $x \ge 0$.

Proof. The density function of X is symmetric around 0. As $\frac{t}{x} \ge 1$ for $t \ge x$, it follows

$$\begin{aligned}\mathbf{Pr}[|X| > x] &= 2\int_x^\infty \frac{1}{\sqrt{2\pi}}\cdot e^{-t^2/2}\,\mathrm{d}t \le \frac{2}{\sqrt{2\pi}x}\int_x^\infty t\cdot e^{-t^2/2}\,\mathrm{d}t\\ &= \frac{\sqrt{2}}{\sqrt{\pi}\cdot x}\cdot\left[-e^{-t^2/2}\right]_x^\infty = \frac{\sqrt{2}\cdot e^{-x^2/2}}{\sqrt{\pi}\cdot x}.\end{aligned}$$

◀

Proof of Lemma 4.1. We use a union bound over all nd coordinates of the points in $\mathcal{X}$. Let $x \sim \mathcal{N}(\mu,\sigma^2)$ be such a coordinate. We apply the claim to the random variable $y = (x-\mu)/\sigma \sim \mathcal{N}(0,1)$. Because we assume $\sigma \le 1/\sqrt{2n}$, it holds

$$\begin{aligned}\mathbf{Pr}[|x| > D_{\max}] &\le \mathbf{Pr}\left[|x-\mu| > \sigma\sqrt{2n}\right] = \mathbf{Pr}\left[|y| > \sqrt{2n}\right]\\ &\le \frac{\sqrt{2}\cdot e^{-2n/2}}{\sqrt{\pi}\cdot\sqrt{2n}} < \frac{1}{\sqrt{n}\cdot e^n} = \frac{1}{\sqrt{n}\cdot(e/2)^n\cdot 2^n} \le \frac{1}{n\cdot 2^n},\end{aligned}$$

where the last inequality follows from $(e/2)^n \geq \sqrt{n}$. Now the lemma follows from the union bound over the nd coordinates. ◀

▶ **Lemma 4.3.** *If $\mathcal{F}_1$ does not occur, then the weight of any cut is between 0 and $\phi_{\max} := 16dn^2$.*

Proof. The lower bound trivially holds because all distances are non-negative. For the upper bound it suffices to observe that fewer than n^2 edges cross the cut in any partition and the Euclidean distance between any two points in the hypercube $[-D_{\max}, D_{\max}]^d$ is at most $2\sqrt{d} \cdot D_{\max} \leq 4\sqrt{d}$, where the inequality follows from the assumption $\sigma \leq 1/\sqrt{2n}$. ◀

The following lemma follows from elementary probability theory.

▶ **Lemma 4.4.** *Let $k \in \mathbb{N}$ and $\lambda_1, \ldots, \lambda_k \in \mathbb{Z}$ with $\sum_{i=1}^k \lambda_i \neq 0$. Let $u, v_1, \ldots, v_k \in \mathbb{R}^d$ and let z denote a d-dimensional Gaussian random vector with mean $\mu \in \mathbb{R}^d$ and standard deviation σ. Then for every $\tau \in \mathbb{R}$ and $\delta > 0$,*

$$\mathbf{Pr}\left[u \cdot z + \textstyle\sum_{i=1}^k \lambda_i \cdot \left\|z - v_i\right\|^2 \in [\tau, \tau + \delta]\right] \leq \tfrac{\sqrt{\delta}}{\sigma}.$$

Proof. Let $\lambda := \sum_{i=1}^k \lambda_i$. Then

$$\begin{aligned} u \cdot z + \sum_{i=1}^k \lambda_i \cdot \left\|z - v_i\right\|^2 &= u \cdot z + \sum_{i=1}^k \lambda_i \cdot (\left\|z\right\|^2 - 2z \cdot v_i + \left\|v_i\right\|^2) \\ &= \lambda \cdot \left\|z\right\|^2 - 2z \cdot \left(\sum_{i=1}^k \lambda_i v_i - \frac{u}{2}\right) + \sum_{i=1}^k \lambda_i \cdot \left\|v_i\right\|^2. \end{aligned}$$

Note that every term except for z is constant. Hence, by completing the square we obtain a constant $\tau_1 \in \mathbb{R}$ such that the previous term simplifies to

$$\begin{aligned} &\lambda \cdot \left\|z\right\|^2 - 2z \cdot \left(\sum_{i=1}^k \lambda_i v_i - \frac{u}{2}\right) + \sum_{i=1}^k \lambda_i \cdot \left\|v_i\right\|^2 \\ =&\lambda \cdot \left\|z - \frac{\sum_{i=1}^k \lambda_i v_i - \frac{u}{2}}{\lambda}\right\|^2 + \tau_1. \\ =&\lambda \cdot \left\|z - x\right\|^2 + \tau_1 \text{ for } x = \frac{\sum_{i=1}^k \lambda_i v_i - \frac{u}{2}}{\lambda}. \end{aligned}$$

We may assume w.l.o.g. that $\lambda \geq 1$. Then

$$\begin{aligned} &\mathbf{Pr}\left[u \cdot z + \sum_{i=1}^k \lambda_i \cdot \left\|z - v_i\right\|^2 \in [\tau, \tau + \delta]\right] \\ =&\mathbf{Pr}\left[\lambda \cdot \left\|z - x\right\|^2 + \tau_1 \in [\tau, \tau + \delta]\right] \\ =&\mathbf{Pr}\left[\left\|z - x\right\|^2 \in \left[\tau_2, \tau_2 + \frac{\delta}{\lambda}\right]\right] \text{ for } \tau_2 = \frac{\tau - \tau_1}{\lambda}. \end{aligned}$$

We use the principle of deferred decisions and assume that the last $d-1$ coordinates $z_2, \ldots, z_d$

of z are already uncovered. Then

$$\begin{aligned}
&\mathbf{Pr}_z\left[\left\|z-x\right\|^2 \in \left[\tau_2, \tau_2+\frac{\delta}{\lambda}\right]\right] \\
\leq &\max_{\tau_3\in\mathbb{R}} \mathbf{Pr}_{z_1}\left[(z_1-x_1)^2 \in \left[\tau_3, \tau_3+\frac{\delta}{\lambda}\right]\right] \\
\leq &\max_{\tau_3\in\mathbb{R}} \mathbf{Pr}_{z_1}\left[z_1 \in \left[x_1+\sqrt{\tau_3}, x_1+\sqrt{\tau_3+\frac{\delta}{\lambda}}\right]\right] + \mathbf{Pr}_{z_1}\left[z_1 \in \left[x_1-\sqrt{\tau_3+\frac{\delta}{\lambda}}, x_1-\sqrt{\tau_3}\right]\right].
\end{aligned}$$

Since both intervals have length $\sqrt{\tau_3+\frac{\delta}{\lambda}}-\sqrt{\tau_3} \leq \sqrt{\frac{\delta}{\lambda}}$, we can bound the probability further by

$$\begin{aligned}
&\mathbf{Pr}_z\left[\left\|z-x\right\|^2 \in \left[\tau_2, \tau_2+\frac{\delta}{\lambda}\right]\right] \\
\leq 2\cdot &\max_{\tau_4\in\mathbb{R}} \mathbf{Pr}_{z_1}\left[z_1 \in \left[\tau_4, \tau_4+\sqrt{\frac{\delta}{\lambda}}\right]\right] \\
\leq &\frac{2}{\sqrt{2\pi}\sigma}\cdot\sqrt{\frac{\delta}{\lambda}} \leq \frac{\sqrt{\delta}}{\sqrt{\lambda}\sigma} \leq \frac{\sqrt{\delta}}{\sigma},
\end{aligned}$$

where the first inequality in the last line follows because the density of z_1 is bounded from above by $1/(\sqrt{2\pi}\sigma)$ and the last inequality follows from $\lambda \geq 1$. ◀

▶ **Lemma 4.5.** *Let $u \in \mathbb{R}^d$ with $\left\|u\right\| \neq 0$, and let X be a d-dimensional Gaussian random vector with mean $\mu \in \mathbb{R}^d$ and standard deviation σ. Then for every $\tau \in \mathbb{R}$ and $\delta > 0$,*

$$\mathbf{Pr}\left[u\cdot X \in [\tau, \tau+\delta]\right] \leq \frac{\delta}{2\cdot\left\|u\right\|\cdot\sigma} \ .$$

Proof. Since every X_i, $i \leq d$, is a normally distributed random variable with standard deviation σ, the product $u_i \cdot X_i$ is normally distributed with standard deviation $|u_i|\cdot\sigma$. Therefore, $u\cdot X$ has variance $\sum_{i=1}^d (|u_i|\cdot\sigma)^2 = \sigma^2\cdot\sum_{i=1}^d |u_i|^2 = \sigma^2\cdot\left\|u\right\|^2$. This means that the density function of $u\cdot X$ is bounded from above by $(\sqrt{2\pi}\sigma\cdot\left\|u\right\|)^{-1}$, from which the lemma follows. ◀

▶ **Corollary 4.6.** *Let $\lambda \in \mathbb{Z}$ be an integer, let $\delta > 0$, let $u \in \mathbb{R}^d$ with $\left\|u\right\| \geq \frac{\sqrt{\delta}}{4}$, and let X denote a d-dimensional Gaussian random vector with mean $\mu \in \mathbb{R}^d$ and standard deviation σ. Then for every $\tau \in \mathbb{R}$,*

$$\mathbf{Pr}\left[\lambda\cdot\left\|X\right\|^2 - 2u\cdot X \in [\tau, \tau+\delta]\right] \leq \frac{\sqrt{\delta}}{\sigma} \ . \tag{2}$$

Proof. If $\lambda \neq 0$, use Lemma 4.4. Otherwise use Lemma 4.5 with $u' = -2u$. ◀

4.3 Bounding the Smoothed Number of Steps

Let L be a fixed sequence of steps with $\ell = \Theta(d)$ active points. We may assume without loss of generality that the active points are the points $1, \ldots, \ell$ and that the first move of point i is before the first move of point j if and only if $i < j$. Furthermore, let $\pi^i \in \{-1, 1\}^n$ be the configuration of all points before the first flip of point i. Note that π^1 is the initial configuration of the points.

4.3.1 Approximation for the Improvement of a Flip

We start by rewriting the improvement of the objective function that a flip makes.

▶ **Lemma 4.7.** *Let π be a configuration of all points and let $v \in N$ flip its state. Then the improvement made by the flip of v is given by*

$$\pi_v \cdot \left(\sum_{w\in N} \pi_w\right) \cdot \left\|X_v\right\|^2 - 2\pi_v \cdot \left(\sum_{w\in N} \pi_w \cdot X_w\right) \cdot X_v + \pi_v \cdot \left(\sum_{w\in N} \pi_w \cdot \left\|X_w\right\|^2\right). \tag{3}$$

Proof. As v switches its state, an edge vw enters the cut if $\pi_v = \pi_w$, and leaves the cut otherwise. Hence, the improvement can be written as

$$\begin{aligned}\sum_{w\neq v} \pi_v\pi_w \cdot \left\|X_v - X_w\right\|^2 &= \sum_{w\in N} \pi_v\pi_w \cdot \left\|X_v - X_w\right\|^2 \\ &= \sum_{w\in N} \pi_v\pi_w \left(\left\|X_v\right\|^2 - 2X_vX_w + \left\|X_w\right\|^2\right) \\ &= \pi_v \sum_{w\in N} \pi_w \cdot \left\|X_v\right\|^2 - 2\pi_vX_v \sum_{w\in N} \pi_wX_w + \pi_v \sum_{w\in N} \pi_w\left\|X_w\right\|^2. \text{◀}\end{aligned}$$

An improvement is bad if the term (3) lies in the interval $(0, \varepsilon]$. Note the structural resemblance to (2), which also contains a quadratic and a linear dependence on the random vector $X = X_v$. Our goal is now to apply Corollary 4.6 to (3) once for every active point, yielding a combined bound of roughly $(\sqrt{\varepsilon}/\sigma)^\ell$ (assuming that the applications can be made independently of each other). In order to achieve this, we have to overcome three obstacles:

- Corollary 4.6 requires that the norm of u is not too small. We will show later that this happens often with high probability.
- The prefactors in (3) are not known without determining the exact configuration of all points. This would yield a factor 2^n in the union bound, which is too large for our purposes.
- An application of Corollary 4.6 and Lemma 4.7 would be done in the following way: Let v be the point that is about to flip its state. Uncover all other points to fix the prefactors of (3) (to be precise, the prefactors also depend on X_v, but this dependency can easily be avoided, see the proof of Lemma 4.7). Then Corollary 4.6 is applicable.
  This approach would mean that we could not use Corollary 4.6 and Lemma 4.7 independently for every active vertex because after the first application every random point X_i would already be uncovered, i.e., exactly determined.

Therefore we alter the approach by guessing the prefactors of (3) up to a small error. This idea is inspired by Angel et al. [3].

- Guess $\lambda = \sum_{w\in N} \pi_w^1$. There are $2n + 1$ possible values for λ.
- Guess a point $c \in (\frac{2\varepsilon}{d}\mathbb{Z})^d$ with $\left\|c - \sum_{w\in N} \pi_w^1 \cdot X_w\right\|_\infty \leq \frac{\varepsilon}{d}$. If we assume that $\mathcal{F}_1$ does not occur, the point $\sum_{w\in N} \pi_w^1 \cdot X_w$ lies in $[-2n, 2n]^d$, i.e., there are $O((\frac{nd}{\varepsilon})^d)$ possible choices for c.
- Guess $\xi \in 2\varepsilon\mathbb{Z}$ such that $\left|\xi - \sum_{w\in N} \pi_w^1 \cdot \left\|X_w\right\|^2\right| \leq \varepsilon$. Again if we assume that $\mathcal{F}_1$ does not occur, every $\left\|X_w\right\|^2$ is bounded from above by $2d$, which means that there are $O(\frac{nd}{\varepsilon})$ possible choices for ξ.

This way we get a good approximation of the actual improvement by a vertex flip using a union bound of size only $O(n \cdot (\frac{nd}{\varepsilon})^{d+1})$ instead of 2^n. The improvement made in the first step of the sequence L (in which point 1 flips) is approximated by $\pi_1^1 \cdot \lambda \cdot \left\|X_1\right\|^2 - 2\pi_1^1 \cdot c \cdot X_1 + \pi_1^1 \cdot \xi$ with an error of $O(\varepsilon)$. Note that we can use Corollary 4.6 uncovering only X_1, i.e., the other points $X_2, \dots, X_n$ are still covered.

We now want to use the same tools for the first flip of every active vertex $2, \dots, \ell$. Instead of guessing all the parameters again, we use that the prefactors in (3) are only deterministically changed by points that previously have already been active. As an example, consider the factor $c^* := \sum_{w \in N} \pi_w^1 \cdot X_w$, for which c is an approximation, i.e., $\left\|c - c^*\right\|_\infty \leq \frac{\varepsilon}{d}$. If a point i flips from side 1 to side -1, the vector c^* as well as c are translated by the exact same vector $-2 \cdot X_i$. Because we have already uncovered X_i if this is not the first flip by point i, this translation of c is deterministic. Hence, if we want to use Corollary 4.6 for the first flip of a point j, we can guess the prefactor $\sum_{w \in N} \pi_w^j \cdot X_w$ by simply translating our guess c by $\sum_{w \in N} \pi_w^j \cdot X_w - c^*$, which is a linear combination of $X_1, \dots, X_{j-1}$.

Let us make this observation more formal. The configurations π^i and π^1 can only differ in the first $i-1$ coordinates. To be precise, $\pi_v^i - \pi_v^1 = 0$ for every point v that flipped an even number of times (including the case that v did not flip at all, i.e., $v \geq i$), and $\pi_v^i - \pi_v^1 = -2\pi_v^1$ for all points $v < i$ that flipped an odd number of times. Therefore we define B as the set of all vectors $\zeta \in \mathbb{R}^n$ such that

- $\zeta_i \in \{0, -2\pi_i^1\}$ for $1 \leq i \leq \ell$,
- $\zeta_i = 0$ for $\ell + 1 \leq i \leq n$.

Furthermore we define $C := \{c + \sum_{v \leq \ell} \zeta_v \cdot X_v \mid \zeta \in B\}$. This means that C contains every possible approximation for the prefactor $\sum_{w \in N} \pi_w^i \cdot X_w$. Now because for every $i \in N$ the difference $\pi^i - \pi^1$ lies in B, we can write $\pi^i = \pi^1 + \zeta$ for some $\zeta \in B$. This gives rise to the following lemma.

▶ **Lemma 4.8.** *For every $v \leq \ell$ there are constants $\alpha_v, \gamma_v \in \mathbb{R}$ and a vector $\beta_v \in C$, each of which only depends on $\lambda, c, \xi, \pi^v - \pi^1$, and $X_1, \dots, X_{v-1}$, but not on $X_v, \dots, X_n$, such that the following holds:*

If $\mathcal{F}_1$ does not occur and the improvement made by the first flip of point v is at most ε, then

$$\left|\alpha_v \cdot \left\|X_v\right\|^2 - 2\beta_v \cdot X_v + \gamma_v\right| \leq 6\varepsilon.$$

Proof. For notational simplicity, let $(X_v)^2 = \left\|X_v\right\|^2$ and $(X_v)^0 = 1$. According to Lemma 4.7, the improvement made by flipping v is given by

$$\pi_v^v \cdot \left(\sum_{w \in N} \pi_w^v\right) \cdot \left\|X_v\right\|^2 - 2\pi_v^v \left(\sum_{w \in N} \pi_w^v X_w\right) \cdot X_v + \pi_v^v \left(\sum_{w \in N} \pi_w^v \cdot \left\|X_w\right\|^2\right).$$

Let $\lambda^* = \sum_{w \in N} \pi_w^1 = \lambda$, $c^* = \sum_{w \in N} \pi_w^1 \cdot X_w$, and $\xi^* = \sum_{w \in N} \pi_w^1 \cdot \left\|X_w\right\|^2$. Then the improvement made by flipping point 1 for the first time is given by

$$\pi_1^1 \cdot \left(\lambda^* \cdot \left\|X_v\right\|^2 - 2c^* \cdot X_v + \xi^*\right).$$

Now we want to rewrite the improvement made by flipping a point v for the first time. Note that $\pi_v^v = \pi_v^1$, i.e., we can replace π_v^v by π_v^1. Let $\zeta := \pi^v - \pi^1 \in B$. Then for

every $i \in \{0,1,2\}$, the difference between $\sum_{w \in N} \pi_w^v \cdot (X_w)^i$ and $\sum_{w \in N} \pi_w^1 \cdot (X_w)^i$ is given by $\eta_i = \sum_{w<v} \zeta_w \cdot (X_w)^i$. This means that the improvement made by flipping v for the first time can be rewritten as

$$\pi_v^1 \cdot \left((\lambda^* + \eta_0) \cdot \left\|X_v\right\|^2 - 2(c^* + \eta_1) \cdot X_v + (\xi^* + \eta_2) \right).$$

We can now set

$$\begin{aligned} \alpha_v &:= (\lambda + \eta_0), \\ \beta_v &:= (c + \eta_1), \text{ and} \\ \gamma_v &:= (\xi + \eta_2). \end{aligned}$$

Note that $\beta_v \in C$ and $\alpha_v, \beta_v, \gamma_v$ in fact do not depend on $X_v, \dots, X_n$. Then we can again rewrite the improvement made by flipping v by

$$\pi_v^1 \cdot \left(\alpha_v \cdot \left\|X_v\right\|^2 - 2\beta_v \cdot X_v + \gamma_v - 2(c^* - c) \cdot X_v + \xi^* - \xi \right).$$

Because c and ξ are approximate guesses for c^* and ξ^*, it holds that $|\xi^* - \xi| \leq \varepsilon$ and $|2(c^* - c) \cdot X_v| \leq 2 \cdot \frac{\varepsilon}{d} \cdot \left\|X_v\right\|_1 \leq 4\varepsilon$, where the last inequality holds because we assume that $\mathcal{F}_1$ does not occur, i.e., every coordinate of X_v lies in $[-D_{\max}, D_{\max}] \subseteq [-2,2]$.

This means that $\left| \alpha_v \cdot \left\|X_v\right\|^2 - 2\beta_v \cdot X_v + \gamma_v \right|$ is by at most 5ε larger than the improvement made by the first flip of v. This yields the lemma. ◀

4.3.2 Bounding the Probability of Insignificant Improvements

The idea developed so far is to apply Corollary 4.6 successively for every active point $1, \cdots, \ell$ to the approximation term in Lemma 4.8. The only problem left is that the point β_v can have small norm, in which case Corollary 4.6 is not applicable. The following lemma shows that this does not occur too often with high probability.

▶ **Lemma 4.9.** *Let $\mathcal{F}_2^\varepsilon$ be the event that there are at least nine different vectors in C with norm at most $\sqrt{\varepsilon}$. Then $\mathbf{Pr}[\mathcal{F}_2^\varepsilon] \leq 2^{4\ell} \cdot \left(\frac{\sqrt{\varepsilon}}{\sigma}\right)^{4d}$.*

Proof. Let C_ε be the subset of vectors of C with norm at most $\sqrt{\varepsilon}$. Consider first the case that there are four vectors x_1, x_2, x_3, x_4 in C such that $x_1 - c, x_2 - c, x_3 - c, x_4 - c$ are linearly independent. Every coordinate of a vector in C_ε must be contained in the interval $[-\sqrt{\varepsilon}, \sqrt{\varepsilon}]$, i.e., its absolute value must be in the interval $[0, \sqrt{\varepsilon}]$. Furthermore, every coordinate is normally distributed with variance at least σ^2 and thus the density of its absolute value is bounded by $1/\sigma$. Hence, Lemma 3.3 bounds the probability that $x_1, \dots, x_4$ are all contained in C_ε by $\left(\frac{\sqrt{\varepsilon}}{\sigma}\right)^{4d}$. Together with a union bound of size $|C|^4 = 2^{4\ell}$ for the different choices of x_1, x_2, x_3, x_4, this yields the desired bound.

It remains to show that $|C_\varepsilon| \leq 8$ if C_ε contains only at most three linearly independent vectors. To simplify notation, let D be the set of vectors $z \in \{0,1\}^\ell$ such that $c + \sum_{v \leq \ell} z_v \cdot (-2\pi_i^1) \cdot X_v \in C_\varepsilon$. That means, a vector in D indicates for every coordinate i whether the according point i has flipped an odd number of times. It clearly holds $|C_\varepsilon| = |D|$ and if a vector set in C_ε is linearly independent when shifted by $-c$, then the corresponding vector set in D must also be linearly independent. Hence, we can assume that there do not exist four linearly independent vectors in D. That means that D is contained in a linear subspace of $\mathbb{R}^\ell$ with dimension 3. The following claim then implies $|D| \leq 8$ and hence also $|C_\varepsilon| \leq 8$.

▶ Claim 4.10. Let U be a linear subspace of $\mathbb{R}^\ell$. Then $|U \cap \{0,1\}^\ell| \leq 2^{\dim(U)}$.

Let $x_1, \ldots, x_k$ be a maximum set of linearly independent vectors in $U \cap \{0,1\}^\ell$. We may assume that $x_1, \ldots, x_k$ is a basis of U by simply restricting U to the linear span of $x_1, \ldots, x_k$. Then every vector $v \in U$ can be written as $\sum_{i=1}^{k} \mu_i \cdot x_i$ for some $\mu_1, \ldots, \mu_k \in \mathbb{R}$. This means that we can write U as

$$U = \left\{ \left(\mu \cdot \lambda^1, \ldots, \mu \cdot \lambda^\ell\right) \mid \mu \in \mathbb{R}^k \right\} \text{ for } (\lambda^1 \ldots \lambda^\ell) = (x_1 \ldots x_k)^\intercal.$$

Let $i_1, \ldots, i_k$ be an index subset such that $\{\lambda^1, \ldots, \lambda^\ell\}$ lies in the span of $\lambda^{i_1}, \ldots, \lambda^{i_k}$. This subset exists because $\operatorname{rank}(\lambda^1 \ldots \lambda^\ell) = \operatorname{rank}(x_1 \ldots x_k)^\intercal = k$. If $\mu \cdot \lambda^{i_1}, \ldots, \mu \cdot \lambda^{i_k}$ are already fixed, then so is $\mu \cdot \lambda^j$ for every $1 \leq j \leq \ell$, since λ^j can be expressed as a linear combination of $\lambda^{i_1}, \ldots, \lambda^{i_k}$. Therefore, two vectors $v, w \in U$ are equal if and only if $(v_{i_1}, \ldots, v_{i_k}) = (w_{i_1}, \ldots, w_{i_k})$. Hence, there can only be at most $|\{0,1\}|^k \leq 2^{\dim(U)}$ different vectors in $U \cap \{0,1\}^\ell$. This completes the proof of both the claim and the lemma. ◀

▶ **Lemma 4.11.** *Let Δ be the smallest improvement made by any sequence L with exactly $\ell := 4d+8$ active vertices, beginning in an arbitrary starting configuration π, and let $0 < \varepsilon \leq 1$. Then $\mathbf{Pr}[\Delta \leq \varepsilon] \leq \mathbf{Pr}[\mathcal{F}_1] + 2^{17d^2+O(d)} n^{5d+10} \sigma^{-4d} \varepsilon^{d-1}$.*

Proof. Let us first derive a union bound over all the ingredients we need:

- There are n^ℓ choices for the active vertices of L.
- A union bound over all possible sequences L with ℓ active vertices would be too large. But in order to use Lemma 4.8 we only need to know the configuration of the active points before an application of that lemma. We use the lemma at most ℓ times. Hence, by introducing a factor 2^{ℓ^2} in the union bound we gather enough information about the sequence L.
- As discussed before, there are $O(n \cdot (\frac{nd}{\varepsilon})^{d+1})$ choices for λ, c, and ξ.
- If $\mathcal{F}_2^\varepsilon$ does not occur, then there are at most eight vertices v such that $\left\|\beta_v\right\| \leq \sqrt{\varepsilon}$. Let L' be a subset of $\{1, \ldots, \ell\}$ of size $\ell - 8 = 4d$ that contains none of these vertices. There are less than ℓ^8 possible choices for L'.

In total this yields a union bound of size $O(2^{\ell^2} n^{\ell+d+2} d^{d+1} \ell^8 \varepsilon^{-(d+1)}) \leq 2^{17d^2+O(d)} n^{5d+10} \varepsilon^{-(d+1)}$.

For $1 \leq v \leq \ell$, let $\alpha_v, \beta_v, \gamma_v$ be defined like in Lemma 4.8. Let $\mathcal{F}_3$ be the event

$$\forall v \leq \ell\colon \left| \alpha_v \cdot \left\|X_v\right\|^2 - 2\beta_v \cdot X_v + \gamma_v \right| \leq 6\varepsilon.$$

Because the improvement Δ can be bounded from below by the smallest single improvement during the whole sequence L, we can according to Lemma 4.8 bound the probability for the event $\Delta \leq \varepsilon$ by

$$\mathbf{Pr}[\Delta \leq \varepsilon] = \mathbf{Pr}[(\Delta \leq \varepsilon) \cap \mathcal{F}_1] + \mathbf{Pr}[(\Delta \leq \varepsilon) \cap (\neg \mathcal{F}_1)] \leq \mathbf{Pr}[\mathcal{F}_1] + \mathbf{Pr}[\mathcal{F}_3^\varepsilon \cap (\neg \mathcal{F}_1)].$$

The last probability can be bounded in the following way:

$$\mathbf{Pr}[\mathcal{F}_3^\varepsilon \cap (\neg \mathcal{F}_1)] \leq \mathbf{Pr}[\mathcal{F}_3^\varepsilon] \leq \mathbf{Pr}[\mathcal{F}_2^\varepsilon] + \mathbf{Pr}[\mathcal{F}_3^\varepsilon \cap (\neg \mathcal{F}_2^\varepsilon)].$$

Because α_v, β_v, and γ_v do not depend on the random variables $X_v, \ldots, X_n$, we can apply Corollary 4.6 successively and independent of the former applications for the first flip of every vertex $1, \ldots, \ell$. If $v \in L'$, then Corollary 4.6 yields

$$\mathbf{Pr}\left[\left| \alpha_v \cdot \left\|X_v\right\|^2 - 2\beta_v \cdot X_v + \gamma_v \right| \leq 6\varepsilon \right] \leq \frac{\sqrt{12\varepsilon}}{\sigma}.$$

Hence, if the event $\mathcal{F}_2^\varepsilon$ does not occur, the application of Corollary 4.6 to all vertices in L' yields

$$\mathbf{Pr}\left[\mathcal{F}_3^\varepsilon \cap (\neg\mathcal{F}_2^\varepsilon)\right] \leq \left(\frac{\sqrt{12\varepsilon}}{\sigma}\right)^{4d} = \frac{(12\varepsilon)^{2d}}{\sigma^{4d}}.$$

Therefore, the probability that the whole sequence yields an improvement of at most ε can be bounded by

$$\begin{aligned}\mathbf{Pr}\left[\Delta \leq \varepsilon\right] &\leq \mathbf{Pr}\left[\mathcal{F}_1\right] + \mathbf{Pr}\left[\mathcal{F}_2^\varepsilon\right] + \frac{2^{17d^2+O(d)}n^{5d+10}}{\varepsilon^{d+1}} \cdot \frac{(12\varepsilon)^{2d}}{\sigma^{4d}} \\ &\leq \mathbf{Pr}\left[\mathcal{F}_1\right] + 2^{4\ell} \cdot \frac{\varepsilon^{2d}}{\sigma^{4d}} + \frac{2^{17d^2+O(d)}n^{5d+10}}{\varepsilon^{d+1}} \cdot \frac{\varepsilon^{2d}}{\sigma^{4d}} \\ &\leq \mathbf{Pr}\left[\mathcal{F}_1\right] + \frac{2^{17d^2+O(d)}n^{5d+10}}{\varepsilon^{d+1}} \cdot \frac{\varepsilon^{2d}}{\sigma^{4d}} = \mathbf{Pr}\left[\mathcal{F}_1\right] + 2^{17d^2+O(d)}n^{5d+10}\sigma^{-4d}\varepsilon^{d-1}. \blacktriangleleft\end{aligned}$$

4.3.3 Calculating the Expected Number of Steps

The rest of the proof is a simple calculation. For this, let $\kappa' := 2^{17d}n^5\sigma^{-4}$ and $\kappa := 2^{\ell+5}n^2d\cdot\kappa' = 2^{21d+13}n^7d\sigma^{-4}$.

▶ **Lemma 4.12.** *For any dimension $d \geq 2$, it holds*

$$\int_{\kappa'}^{2^n} \left(\frac{2^{17d}n^5}{\sigma^4 t}\right)^{d-1} \mathrm{d}t \leq 2^{17d}n^6\sigma^{-4}.$$

Proof. Substitute $x = \frac{t\sigma^4}{2^{17d}n^5}$. Then $\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{\sigma^4}{2^{17d}n^5} = \frac{1}{\kappa'}$ and thus

$$\int_{\kappa'}^{2^n} \left(\frac{2^{17d}n^5}{\sigma^4 t}\right)^{d-1} \mathrm{d}t = \int_1^{2^n/\kappa'} \left(\frac{1}{x}\right)^{d-1} \cdot \frac{2^{17d}n^5}{\sigma^4}\,\mathrm{d}x \leq \frac{2^{17d}n^5}{\sigma^4} \cdot \int_1^{2^n} \frac{1}{x}\,\mathrm{d}x \leq 2^{17d}n^6\sigma^{-4}. \blacktriangleleft$$

▶ **Lemma 4.13.** *For any dimension $d \geq 2$, it holds*

$$\int_{\kappa'}^{2^n} \mathbf{Pr}\left[\Delta \leq \frac{1}{t}\right] \mathrm{d}t \leq 2^{O(d)}n^{21}\sigma^{-8}.$$

Proof. First note that $\kappa' \geq 1$ because we assume that $\sigma \leq 1/\sqrt{2n}$. Therefore, we can use Lemma 4.11 together with Lemma 4.1 to obtain the following calculation:

$$\begin{aligned}&\int_{\kappa'}^{2^n} \mathbf{Pr}\left[\Delta \leq \frac{1}{t}\right] \mathrm{d}t \leq \int_{\kappa'}^{2^n} \mathbf{Pr}\left[\mathcal{F}_1\right] + \mathbf{Pr}\left[\Delta \leq \frac{1}{t}\right] \mathrm{d}t \\ \leq& d + \int_{\kappa'}^{2^n} 2^{17d^2+O(d)}n^{5d+10}\sigma^{-4d}t^{-(d-1)}\,\mathrm{d}t \\ =& d + 2^{O(d)}n^{15}\sigma^{-4} \cdot \int_{\kappa'}^{2^n} \left(\frac{2^{17d}n^5}{\sigma^4 t}\right)^{d-1} \mathrm{d}t.\end{aligned}$$

Plugging in Lemma 4.12 yields

$$\int_{\kappa'}^{2^n} \mathbf{Pr}\left[\Delta \leq \frac{1}{t}\right] \mathrm{d}t \leq d + 2^{O(d)}n^{15}\sigma^{-4} \cdot 2^{17d}n^6\sigma^{-4} \leq 2^{O(d)}n^{21}\sigma^{-8}. \blacktriangleleft$$

We are now ready to prove the main theorem.

Proof of Theorem 2.2. We first stick with our assumption $\sigma \leq 1/\sqrt{2n}$. Let T be the number of steps that FLIP takes starting from an arbitrary starting configuration.

Following the definition of Δ, every 2^ℓ steps ϕ increases by at least Δ. Because the size of the maximum cut is bounded from above by $16dn^2$ if $\mathcal{F}_1$ does not occur, FLIP must then terminate after at most $2^\ell \cdot \lceil \frac{16dn^2}{\Delta} \rceil \leq 2^\ell \cdot \frac{32dn^2}{\Delta} = 2^{\ell+5} \cdot \frac{dn^2}{\Delta}$ steps, where the inequality follows from the fact that the minimum increase Δ cannot be larger than $16dn^2$.

Because no configuration of the vertices can occur twice in a sequence of strictly improving steps, T cannot be larger than 2^n. This means that the smoothed number of steps $\mathbf{E}[T]$ can be bounded in the following way.

$$\begin{aligned}\mathbf{E}[T] &= \int_0^\infty \mathbf{Pr}[T \geq t]\,\mathrm{d}t = \int_0^{2^n} \mathbf{Pr}[T \geq t]\,\mathrm{d}t \leq \kappa + \int_\kappa^{2^n} \mathbf{Pr}[\mathcal{F}_1] + \mathbf{Pr}\left[2^{\ell+5} \cdot \frac{dn^2}{\Delta} \geq t\right] \mathrm{d}t \\ &\leq d + \kappa + \int_\kappa^{2^n} \mathbf{Pr}\left[\Delta \leq \frac{2^{\ell+5}dn^2}{t}\right] \mathrm{d}t\end{aligned}$$

Substitute $x = \frac{t}{2^{\ell+5}dn^2} = \frac{t \cdot \kappa'}{\kappa}$. Then $\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{\kappa'}{\kappa}$ and thus

$$\mathbf{E}[T] \leq d + \kappa + \frac{\kappa}{\kappa'} \cdot \int_{\kappa'}^{2^n \cdot \kappa'/\kappa} \mathbf{Pr}\left[\Delta \leq \frac{1}{x}\right] \mathrm{d}x \leq d + \kappa + \frac{\kappa}{\kappa'} \cdot \int_{\kappa'}^{2^n} \mathbf{Pr}\left[\Delta \leq \frac{1}{x}\right] \mathrm{d}x.$$

Using $\frac{\kappa}{\kappa'} = 2^{O(d)} \cdot n^2 d$ together with Lemma 4.13 yields

$$\begin{aligned}\mathbf{E}[T] &\leq d + 2^{O(d)} \cdot n^7 d\sigma^{-4} + 2^{O(d)} \cdot n^2 d \cdot 2^{O(d)} \cdot n^{21}\sigma^{-8} \\ &\leq 2^{O(d)} \cdot n^7\sigma^{-4} + 2^{O(d)} \cdot n^{23}\sigma^{-8} \leq 2^{O(d)} \cdot n^{23}\sigma^{-8}.\end{aligned}$$

If $\sigma > 1/\sqrt{2n}$, we create an equivalent instance by scaling down the mean values by the factor $1/(\sqrt{2n}\sigma)$ (i.e., the mean values remain in $[0,1]^n$) and setting the standard deviation to $\sigma' = 1/\sqrt{2n}$. As these instances are equivalent, we obtain the same expected number of iterations and thus a bound of $2^{O(d)} \cdot n^{23} \cdot (\sqrt{2n})^8 \leq 2^{O(d)} \cdot n^{27}$. ◀

4.4 An Exponential Lower Bound in the Deterministic Setting

In this subsection we prove Theorem 2.3.

Proof of Theorem 2.3. For this proof we name the sides of a partition 0 and 1 instead of -1 and 1. We use XOR gadgets as depicted in Figure 2. These gadgets follow the idea by Haken and Luby [61]. Every gadget contains four vertices a, b, c, d and two additional "anchor" vertices (the ones in squares) that are permanently assigned to side 1, i.e., they never flip sides at all. Given such a gadget, every time vertex a or vertex b flips sides, the vertices c and d are checked in the given order whether they can also be flipped. If neither c nor d can be flipped, their logical meaning is exactly a NOR b and a XOR b, respectively. Hence, vertex d can be flipped as often as vertices a and b combined.

We can now concatenate $n/6$ gadgets in such a way that the edge weights are scaled down by the factor 10 for adjacent gadgets. Then the vertices a' and b' of the next gadget can be flipped each whenever vertex d is flipped. This means that the vertices a' and b' can flip twice as often as the vertices a and b. Add now an additional vertex and two very expensive edges to the vertices a and b of the top-most gadget such that a and b can flip sides. This completes the construction.

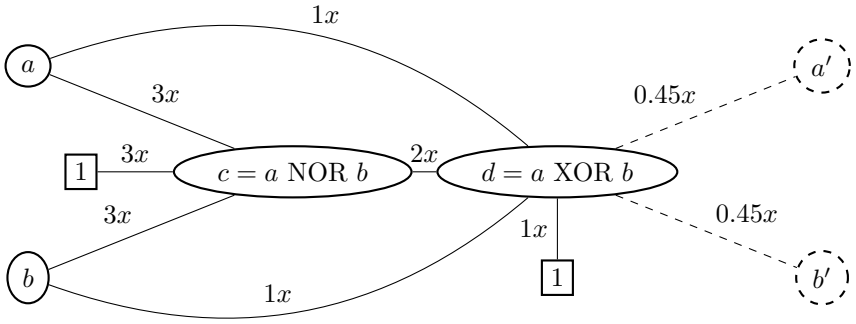
It remains to show that the generated instance can be embedded into the plane with squared Euclidean distances. Given a gadget with scaling factor x, we place the vertices at the following coordinates:

- Vertices a and b are both located at $(0, \sqrt{x})$.
- Vertex c is located at $(\sqrt{2x}, 0)$.
- Vertex d is located at $(0, 0)$.
- The anchors are located at $(\sqrt{2x}, \sqrt{3x})$ and $(0, \sqrt{x})$.

We now verify that all edge weights inside a XOR gadget are correct:

- For the edge between c and the first anchor: $\left\|(\sqrt{2x}, \sqrt{3x}) - (\sqrt{2x}, 0)\right\|^2 = 3x$.
- For the edge between d and the second anchor: $\left\|(0, \sqrt{x}) - (0, 0)\right\|^2 = x$.
- For the edges $\{a, c\}$ and $\{b, c\}$: $\left\|(\sqrt{2x}, 0) - (0, \sqrt{x})\right\|^2 = 2x + x = 3x$.
- For the edges $\{a, d\}$ and $\{b, d\}$: $\left\|(0, 0) - (0, \sqrt{x})\right\|^2 = x$.
- For the edge $\{c, d\}$: $\left\|(0, 0) - (\sqrt{2x}, 0)\right\|^2 = 2x$.

To make the edge weights of the edges $\{d, a'\}$ and $\{d, b'\}$ correct, we simply shift the gadgets accordingly. Note that this does not impose a problem, as a' and b' have the same position in the plane. This completes the proof. ◀



a	b	$c = a$ NOR b	$d = a$ XOR b
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	0

Figure 2 A XOR gadget. Every flip of a or b induces a flip of d. The table shows the preferred side for the vertices c and d, given a configuration for a and b.

5 Scheduling

This part is devoted to the convergence time of local search in different scheduling settings. First we show in Section 5.1 how to convert superpolynomial deterministic convergence times to smoothed polynomial convergence times. In Section 5.2 and Section 5.3 we deal with the special cases of identical machines and unit-weight jobs, respectively, before we turn to the more general case of related machines in Section 5.4. We conclude with the analysis of the price of anarchy in the FIFO model in Section 5.5.

5.1 Smoothed Analysis

Some of our shown convergence times include the factor $W/p_{\min}$. While in the worst case this fraction can be exponentially large, in the smoothed setting they turn into expected polynomial convergence times.

▶ **Lemma 5.1.** *If the convergence time is bounded by $f(m,n)\cdot W/p_{\min}$ for some polynomial f, then the smoothed convergence time is bounded by $O(f(m,n)\cdot n^3\log(m)\cdot\phi)$.*

Proof. Let T be a random variable for the convergence time. Then T is trivially bounded by m^n as there are only m^n different schedules and no schedule can appear twice in a sequence of monotonically improving steps. As the job sizes are each drawn from the interval $[0,1]$ and thus the total weight W is at most n, we can bound the expected value of T in the following way.

$$\begin{aligned}\mathbf{E}[T] &= \int_0^\infty \mathbf{Pr}[T \geq \alpha]\,\mathrm{d}\alpha = \int_0^{m^n} \mathbf{Pr}[T \geq \alpha]\,\mathrm{d}\alpha \leq \int_0^{m^n} \mathbf{Pr}\left[f(m,n)\cdot\frac{n}{p_{\min}} \geq \alpha\right]\mathrm{d}\alpha \\ &= f(m,n)\cdot n\cdot\int_0^{\frac{m^n}{f(m,n)\cdot n}} \mathbf{Pr}[p_{\min}\leq 1/\alpha]\,\mathrm{d}\alpha \leq f(m,n)\cdot n\cdot\int_0^{m^n}\mathbf{Pr}[p_{\min}\leq 1/\alpha]\,\mathrm{d}\alpha.\end{aligned}$$

The random variable $p_{\min}$ is at least $1/\alpha$ exactly if all job sizes are at least $1/\alpha$. For every job size p_j this happens independently of the other jobs sizes with probability at least $1-\phi/\alpha$ as the density function of p_j is bounded by ϕ. Hence, with a Union Bound it follows

$$\mathbf{Pr}[p_{\min}\leq 1/\alpha] = \mathbf{Pr}[\exists j\colon p_j\leq 1/\alpha] \leq \sum_{j=1}^{n}\mathbf{Pr}[p_j\leq 1/\alpha] \leq \frac{n\phi}{\alpha},$$

and thus,

$$\int_0^{m^n}\mathbf{Pr}[p_{\min}\leq 1/\alpha]\,\mathrm{d}\alpha \leq 1+\int_1^{m^n}\frac{n\phi}{\alpha}\,\mathrm{d}\alpha = 1+n\phi\cdot\ln(m^n) = 1+n^2\phi\cdot\ln m. \qquad ◀$$

Unfortunately, our result about the convergence time of the Best Improvement pivot rule in the Makespan model depends quadratically on $p_{\min}$. This does not allow us to derive an expected polynomial convergence time, but instead we can show that with high probability the convergence time is polynomially bounded.

▶ **Lemma 5.2.** *The smoothed convergence time of the Best Improvement pivot rule in the Makespan model is in $m^2n^7\phi^2$ with probability at least $1-1/n$.*

Proof. Analogously to the proof of Lemma 5.1, we can bound the probability of $1/p_{\min}^2 \geq \alpha$ for some α using a Union Bound.

$$\mathbf{Pr}[1/p_{\min}^2\geq\alpha] \leq \mathbf{Pr}[p_{\min}\leq 1/\sqrt{\alpha}] \leq \frac{n\phi}{\sqrt{\alpha}}.$$

This bound is at most $1/n$ for $\alpha \geq n^4\phi^2$. Theorem 5.26 shows that the Best Improvement pivot rule converges in the Makespan model in $O(m^2n \cdot W^2/p_{\min}^2)$ steps. As $W^2 \leq n^2$, the probability that the Best Improvement pivot rule does not converge in $m^2n^7\phi^2$ steps is bounded from above by

$$\mathbf{Pr}[m^2n^3/p_{\min}^2 \geq m^2n^7\phi^2] = \mathbf{Pr}[1/p_{\min}^2 \geq n^4\phi^2] \leq \frac{1}{n}. \quad \blacktriangleleft$$

5.2 Identical Machines

5.2.1 FIFO and Makespan Model

In the FIFO model, the costs of a job decrease monotonically while the minimum load of a machine increases monotonically when considering identical machines. As a moving job always jumps to a machine with minimum load, every job can jump at most once. This leads to the following result.

▶ **Theorem 5.3.** *In the FIFO model, for any pivot rule the worst-case running time is exactly $n-1$.*

Proof. First consider an instance with $m = n$ machines and every job has size 1. In the initial schedule all jobs are assigned to the first machine. Then in every step one job moves from the first machine to an empty machine. This happens $n-1$ times.

For the upper bound, first observe that the running time of the least loaded machine is monotonically increasing (this was also observed in [48]). As a job always strictly decreases its running time by moving and moves to a least loaded machine, each job can only move once. The first job on each machine does not move. Hence, the number of iterations is bounded from above by $n-1$. ◀

For the Makespan model, Even-Dar et al. [48] proved that the Min Weight pivot rule can take as many as $\Omega((n/m^2)^{m-1})$ steps. They also showed that this is near to the worst case as every pivot rule terminates after $O((\frac{n}{K}+1)^K)$ steps, where K is the number of different job weights. We derive the bound $O(n \cdot \frac{W}{p_{\min}})$ for arbitrary pivot rules, which is a significant improvement if $\frac{W}{p_{\min}}$ is small. This bound is almost optimal as it is easy to see that the worst case instance used in [48] has $\frac{p_{\max}}{p_{\min}} = (n/(m-1))^{m-2}$. It is also a generalization of the result that every pivot rule converges in $O(W+n)$ steps in the case of integer weights.

▶ **Theorem 5.4.** *In the Makespan model, every pivot rule terminates after $O(n \cdot \frac{W}{p_{\min}})$ steps.*

Proof. As Even-Dar et al. [48] pointed out (without proof), after a job j moved to machine i, it can only be unsatisfied again after a strictly greater job moved to machine i in the meantime: A job always jumps to a machine with minimum load and the minimum load increases monotonically. Consider the last job j' entering machine i in iteration t' before job j jumps away from machine i in iteration t. Then machine i must be a machine with minimum load before iteration t'. Now if $p_j > p_{j'}$, then L_i^{t+1} would be stricly smaller than $L_i^{t'}$, which is a minimum load in a former iteration. If $p_j = p_{j'}$, then job j cannot be unsatisfied because job j' is not unsatisfied.

Based on their idea of push-out potentials, we define the potential $\phi := \sum_{i=1}^{m} u_i^t \leq W$, where u_i^t is the maximum total weight of jobs on machine i that could consecutively move away from i, starting in the schedule before iteration t. When a job j jumps from machine i to machine i', then $u_{i'}^t$ was 0 beforehand. As mentioned above, no job from any other machine than machine i' can become unsatisfied by the move of job j and thus the potential ϕ decreases by at least $u_i^t - u_i^{t+1} - u_{i'}^{t+1}$.

If $u_{i'}^{t+1}$ was larger than p_j, then there would be a sequence of moves from jobs away from machine i' such that the load of machine i' after these moves would be less than $L_{i'}^t$. But machine i' was a machine with minimum load before iteration t, a contradiction. Note also that $u_i^t - u_i^{t+1} \geq p_j$: Let J' be the jobs on machine i with total weight u_i^{t+1} which could consecutively jump away from machine i after iteration t. Then $J' \cup \{j\}$ could consecutively jump away before iteration t and thus $u_i^t \geq \sum_{j' \in J' \cup \{j\}} p_{j'} = u_i^{t+1} + p_{j'}$. We can conclude that $u_i^t - u_i^{t+1} - u_{i'}^{t+1} \geq 0$ and thus that ϕ is actually a potential.

We call a jump of job j to machine i in iteration t *stable* if after that jump, another job moves to i before a job leaves i. As discussed above, through the stable jump the total potential of all machines except machine i decreases by at least $p_j \geq p_{\min}$ and $u_i^{t'} = 0$ at time t' when the next job enters or leaves machine i. Hence, every stable jump induces a potential drop of at least $p_{\min}$. We maintain a set of indices: In the initial schedule, every job has an index attached to it. When a job j moves away from machine i, then the indices attached to j get transferred to the job j' that moved last to machine i. If no such job exists, the indices get deleted. Afterwards, a new index gets attached to job j on its new machine if it was a stable jump.

When a job j moves to machine i, then no job on machine i was unsatisfied beforehand as j jumps to a machine with minimum load. Thus, when an index gets reattached from job j' to job j, then j made j' unsatisfied and thus p_j is strictly greater than $p_{j'}$ because only larger jobs can make smaller jobs unsatisfied. Therefore, every index can be reattached at most n times. Furthermore, every time a job j jumps away from a machine i, it has at least one index attached to it: Assume to the contrary that it is the first jump without attached indices. If it is the first jump by job j or its last jump was stable, then there is by definition an attached index. Otherwise, there is a job j' that left machine i such that job j is the last job entering machine i beforehand and thus job j' transferred its indices to job j. Hence, the number of indices is at least one nth of the total number of jumps. There can only be $W/p_{\min}$ many stable jumps as otherwise ϕ would be negative. This yields the desired bound. ◀

5.2.2 SJF Model

Finally let us consider the SJF model. The Max Weight pivot rule in the SJF model can take an exponential number of steps even on two identical machines. Also an average-case analysis yields a superpolynomial convergence time. We do not consider the Random pivot rule and the Min Weight pivot rule in this subsection because for these rules we prove in subsection 5.4.3 polynomial upper bounds even for the more general setting of related machines. We leave it as an open question whether the convergence time of the Best Improvement pivot rule is polynomial for identical machines.

▶ **Theorem 5.5.** *In the SJF model, the convergence time of the Max Weight pivot rule is $2^{\Omega(n)}$ even for two identical machines. The smoothed convergence time of the Max Weight pivot rule is $2^{\Omega(\sqrt{n})}$ even for two identical machines and $\phi = 1$.*

The proof idea for Theorem 5.5 is the following. We consider $2k + 1$ jobs with job sizes $p_0 \ll p_1 < \ldots < p_{2k} < (1 + 1/k) \cdot p_1$, which are to be assigned to $m = 2$ identical machines. Initially, job 1 and all even-numbered jobs $j \geq 4$ are assigned to machine 1, whereas job 2 and all odd-numbered jobs $j \geq 3$ are assigned to machine 2. As the job sizes are all almost equal, only the number of jobs smaller than job j on each machine and not their exact weights determine on which machine job j wants to be. One can then show that in every second iteration either job $2k - 1$ or job $2k$ jumps. Then by induction, the number of iterations grows exponentially in k.

One can use the same idea for the second part of the theorem to prove an expected superpolynomial convergence time when all job sizes are drawn uniformly at random from the interval $[0,1]$. Let $k = \Theta(\sqrt{n})$ and let $p_1 < \ldots < p_{2k}$ be the $2k$ largest job sizes. Then with constant probability $p_{2k} < (1 + 1/k) \cdot p_1$ and the other $n - 2k$ smaller jobs can be distributed among the two machines in such a way that they generate nearly the same load on both machines and thus do not affect the jumps of the $2k$ largest jobs.

Now let us go into detail. Let $k \geq 2$. We construct a scheduling instance I_k with $m = 2$ identical machines and $n = 2k + 1$ jobs and an initial schedule for this instance from which the Max Weight pivot rule needs 2^{k-1} steps to a local optimum. The jobs $1, \ldots, 2k$ have processing requirements $p_1 < p_2 < \ldots < p_{2k-1} < p_{2k} < (1 + 1/k) \cdot p_1$. Additionally there is an extremely small job 0 with processing requirement $p_0 \leq (k + 1) \cdot p_1 - k \cdot p_{2k} \leq p_1$. Note that $(k + 1) \cdot p_1 - k \cdot p_{2k} = k \cdot ((1 + 1/k) \cdot p_1 - p_{2k})$ is strictly positive. The additional job 0 is not needed for the proof of the first part of Theorem 5.5. However, we will see later that its existence makes it possible to extend the theorem (with a slightly weaker bound for the convergence time) to ϕ-perturbed scheduling instances.

▶ **Lemma 5.6.** *Consider an arbitrary schedule σ on instance I_k. For a job j and a machine $i \in \{1, 2\}$, let $J_i(j)$ denote the set of jobs on machine i that have a smaller size than job j ignoring job 0, i.e.,*

$$J_i(j) = \{j' \in \{1, \ldots, j-1\} : \textit{job } j' \textit{ is assigned to machine } i\}.$$

- *If $|J_1(j)| < |J_2(j)|$, then job j prefers to jump onto (stay on) machine 1.*
- *If $|J_1(j)| > |J_2(j)|$, then job j prefers to jump onto (stay on) machine 2.*

Lemma 5.6 states that in the case $|J_1(j)| \neq |J_2(j)|$, the values $p_0, p_1, \ldots, p_{2k}$ are not of interest when we have to decide which of the machines 1 and 2 is favored by job j in the current schedule. We only have to count on each machine the jobs whose indices are smaller than j, ignoring job 0. Only in the case $|J_1(j)| = |J_2(j)|$, the values $p_0, p_1, \ldots, p_{2k}$ matter.

Proof of Lemma 5.6. Due to symmetry, we only have to consider the first claim. For this, observe that $|J_1(j)| < k$ since $|J_1(j)| < |J_2(j)|$, $J_1(j) \cap J_2(j) = \emptyset$, and $J_1(j) \cup J_2(j) \subseteq \{1, \ldots, 2k\}$. We obtain

$$\begin{aligned}\sum_{j' \in J_2(j)} p_{j'} - \sum_{j' \in J_1(j)} p_{j'} &\geq |J_2(j)| \cdot p_1 - |J_1(j)| \cdot p_{2k} \geq p_1 + |J_1(j)| \cdot (p_1 - p_{2k}) \\ &> p_1 + k \cdot (p_1 - p_{2k}) = (k+1) \cdot p_1 - k \cdot p_{2k} \geq p_0 .\end{aligned}$$

Hence, the total processing requirement of all jobs on machine 1 that are smaller than job j is at most

$$p_0 + \sum_{j' \in J_1(j)} p_{j'} < \sum_{j' \in J_2(j)} p_{j'} .$$

The latter sum is a lower bound for the total processing requirement of all jobs on machine 2 that are smaller than job j. ◀

▶ **Definition 5.7.** Let σ be an arbitrary schedule on instance I_k. By $\chi(\sigma) \in \mathbb{Z}$ we denote the difference between the number of jobs from $\{1, \ldots, 2k\}$ on machine 1 and the number of jobs from $\{1, \ldots, 2k\}$ on machine 2. We call schedule σ *balanced*, if $\chi(\sigma) = 0$. Otherwise, σ is called *imbalanced*. In an imbalanced schedule, we call the machine with more jobs from $\{1, \ldots, 2k\}$ the *critical machine*.

Now consider the following schedule σ_k: Job 0 is assigned arbitrarily and job $j \in \{1, \dots, 2k\}$ is assigned to

$$\begin{cases} \text{machine 1} & \text{if } j = 1 \text{ or } (j \geq 3 \text{ and } j \text{ is even}), \\ \text{machine 2} & \text{if } j = 2 \text{ or } (j \geq 3 \text{ and } j \text{ is odd}). \end{cases}$$

In the remainder of this subsection, let S denote the sequence of schedules that we obtain when we consider the Max Weight pivot rule, starting with schedule σ_k.

▶ **Lemma 5.8.** *For any schedule σ from the sequence S, $\chi(\sigma) \in \{-2, 0, 2\}$.*

Proof. We prove the claim by induction. For the initial schedule, the claim is true since $\chi(\sigma_k) = 0$. Now consider an arbitrary schedule σ and its predecessor σ' in S. For σ', the induction hypothesis states that $\chi(\sigma') \in \{-2, 0, 2\}$. If $\chi(\sigma') = 0$, then $|\chi(\sigma)| = 2$ since the value χ changes by 2 in each iteration. Let us consider the case $|\chi(\sigma')| = 2$. We show that the largest job j on the critical machine i of σ' will jump in the next iteration yielding $\chi(\sigma) = 0$. Due to $|\chi(\sigma')| = 2$, we obtain $|J_i(j)| = k$ and $|J_{i'}(j)| \leq k - 1$ in the schedule σ', where $i' = 3 - i$ denotes the other machine. Hence, in accordance with Lemma 5.6, job j desires to jump. Due to the Max Weight pivot rule, it prevents all jobs $j' < j$ from jumping. On the other hand, all other jobs $j' > j$ must be on machine i' due to the choice of j. As $|J_i(j')| = |J_i(j)| + 1 = k + 1 > |J_{i'}(j')|$ for the schedule σ', these jobs do not desire to jump. This again follows from Lemma 5.6. Consequently, job j will jump from machine i to machine i' in the next iteration. This concludes the proof. ◀

▶ **Lemma 5.9.** *Let σ be an arbitrary imbalanced schedule from the sequence S. Then at least one of the two largest jobs $2k - 1$ or $2k$ is assigned to the critical machine.*

Proof. Assume, to the contrary, that there is an imbalanced schedule σ in S that assigns both jobs $2k-1$ and $2k$ to the non-critical machine. We consider the first such schedule in the sequence and let i and i' denote the critical and the non-critical machine, respectively. Since the first schedule σ_k in the sequence S is balanced and, thus, is not equal to σ, schedule σ must have a predecessor σ' in the sequence S. Due to Lemma 5.8 and the observation, that the value χ changes by 2 in each iteration, schedule σ' must be balanced. Furthermore, since machine i becomes the critical machine after the following iteration, a job must jump from machine i' to machine i. Consequently, as both, job $2k - 1$ and job $2k$, are not assigned to machine i in schedule σ, they must be assigned to machine i' in schedule σ'. Now let us consider the largest job j that is assigned to machine i in schedule σ'. We will show that this job is the next to jump, contradicting the fact that the next jump is from machine i' to machine i.

As all jobs j' that are larger than j (including job $2k - 1$ and $2k$) are assigned to machine i', none of these can improve by jumping to machine i as $|J_i(j')| = k > |J_{i'}(j')|$ because schedule σ' is balanced and j is the largest job on machine i. On the other hand, we know that $|J_{i'}(j)| \leq |J_{i'}(2k-1)| = k-2 < |J_i(j)|$ since both jobs $2k-1$ and $2k$ are assigned to machine i'. This implies that job j will be the next job to move. ◀

▶ **Corollary 5.10.** *In the sequence S, the number of jumps involving one of the jobs $2k - 1$ or $2k$ is at least half the number of all jumps.*

Proof. Consider an arbitrary imbalanced schedule σ from the sequence S. According to Lemma 5.9, at least one job from $\{2k-1, 2k\}$ is assigned to the critical machine i. Let $j \geq 2k - 1$ be the largest job on the critical machine. Then, $|J_i(j)| = k > |J_{2-i}(j)|$, i.e.,

job j desires to change its machine. Hence, the next jump will be performed by job $2k-1$ or job $2k$.

Since every second schedule from the sequence S is imbalanced and the last schedule of S cannot be imbalanced due to the previous argument, at least half of the jumps involves job $2k-1$ or job $2k$. ◀

The first part of Theorem 5.5 follows directly from the following lemma.

▶ **Lemma 5.11.** *The number of iterations in the sequence S is at least 2^{k-1}.*

Proof. We prove the lemma by induction on k. For the base case $k=2$, job 1 and job 4 are assigned to machine 1 and job 2 and job 3 are assigned to machine 2 in schedule σ_2. Hence, job 3 will jump to machine 1 in the next iteration because $p_1 < p_2$. Afterwards, job 4 will jump to machine 2. Hence, the number of jumps is at least $2 = 2^{2-1}$.

For the inductive step let us consider an arbitrary value $k \geq 3$. First of all observe that

$$p_1 < p_2 < \ldots < p_{2k} < (1+1/k) \cdot p_1 < (1+1/(k-1)) \cdot p_1$$

and

$$\begin{aligned} p_0 &\leq (k+1) \cdot p_1 - k \cdot p_{2k} \\ &< ((k-1)+1) \cdot p_1 - (k-1) \cdot p_{2k} \\ &< ((k-1)+1) \cdot p_1 - (k-1) \cdot p_{2(k-1)}\,. \end{aligned}$$

Particularly, the jobs $0, 1, \ldots, 2(k-1)$, together with the machines 1 and 2, form a scheduling instance I_{k-1} with the properties required to apply the inductive hypothesis. Moreover, when we remove the jobs $2k-1$ and $2k$ from schedule σ_k, then we obtain σ_{k-1} for which we can apply the inductive hypothesis. We classify the iterations that we obtain starting from schedule σ_k as follows: if job $2k-1$ or job $2k$ changes the machine, then we call this iteration a Type 2 iteration. Otherwise it is a Type 1 iteration. Observe that the subsequence of all Type 1 iterations is exactly the sequence of iterations that we get when we start with schedule σ_{k-1}. This is due to the fact that the behavior of the jobs $0, 1, \ldots, 2(k-1)$ is not affected by the larger jobs $2k-1$ and $2k$. Hence, the number of Type 1 iterations is at least 2^{k-2} in accordance with the inductive hypothesis. Finally, Corollary 5.10 states that there are at least as many Type 2 iterations as Type 1 iterations, yielding the lower bound of 2^{k-1} for the total number of iterations. ◀

Proof of Theorem 5.5. Only the second part of the theorem is left to show. Let n denote the number of jobs and let $r_1, \ldots, r_n$ denote the random processing requirements. For $\phi = 1$, every processing requirement r_i is chosen uniformly at random from $[0,1]$. Let $x = \frac{1}{3\sqrt{n}} \leq \frac{1}{3}$ and $k = \lfloor \frac{x}{4} \cdot n \rfloor = \lfloor \frac{\sqrt{n}}{12} \rfloor$. We show that the convergence time is $2^{\Omega(\sqrt{n})}$, unless at least one of two failure events $\mathcal{F}_1$ or $\mathcal{F}_2$ occurs.

We define $\mathcal{F}_1$ to be the event that fewer than $2k$ values r_i lie in the interval $[1-x, 1]$ or that fewer than $n/3$ values r_i lie in the interval $[0, 1-x]$. In expectation, $nx = \frac{\sqrt{n}}{3}$ values r_i lie in the interval $[1-x, 1]$. Due to the Chernoff bound, the probability that fewer than $2k \leq \frac{nx}{2}$ values r_i lie in the interval $[1-x, 1]$, is bounded from above by $\exp(-xn/8) = \exp(-\sqrt{n}/24)$, which tends to 0 for $n \to \infty$. Additionally, the probability that at least $n \cdot 2x \leq 2n/3$ values r_i fall into the interval $[1-x, 1]$ is at most $1/2$ due to Markov's inequality. Hence, with constant probability the failure event $\mathcal{F}_1$ does not occur. From now on assume that this is the case. Then at least $2k$ values lie in $[1-x, 1]$ and at least $n/3$ values lie in $[0, 1-x]$.

By $p_1 < \ldots < p_{2k}$ we denote the $2k$ largest values r_i. Observe that, under the assumption $\neg\mathcal{F}_1$, $p_1 \geq 1 - x$ and hence

$$\begin{aligned}\bar{p_0} &:= (k+1) \cdot p_1 - k \cdot p_{2k} \\ &= p_1 - k \cdot (p_{2k} - p_1) \\ &\geq 1 - x - k \cdot x \\ &= 1 - (k+1) \cdot x\end{aligned}$$

and

$$(k+1) \cdot x = kx + 2x - x \leq \frac{x^2 n}{4} + 2x - x \leq \frac{1}{36} + \frac{2}{3} - x \leq 1 - x\,.$$

Consequently, $\bar{p}_0 \geq x > 0$ and

$$p_{2k} = \frac{1}{k} \cdot ((k+1) \cdot p_1 - \bar{p_0}) < \left(1 + \frac{1}{k}\right) \cdot p_1\,.$$

Hence, the jobs with processing requirements $p_1, \ldots, p_{2k}$ together with the two machines form an instance I_k as used in the proof of the first part of Theorem 5.5 (except for the missing job 0).

This alone does not suffice to prove the theorem because we cannot simply eliminate the other jobs from the instance. Instead we will distribute them onto the two machines in such a way that their total contributions to the loads of the two machines are almost the same. Let $R_1 \subseteq \{i \mid r_i \in [1-x, p_1)\}$ denote the remaining values from $[1-x, 1]$, let $R_2 \subseteq \{i \mid r_i \in [0, 1-x)\}$, and let $R = R_1 \cup R_2$. We look for a subset $R' \subseteq R$ such that the gap $\Delta(R') := \left|\sum_{i \in R'} r_i - \sum_{i \in R \setminus R'} r_i\right|$ is small. For this, we first choose a subset $R'_1 \subseteq R_1$ such that $\alpha(R'_1) := \sum_{i \in R_1 \setminus R'_1} r_i - \sum_{i \in R'_1} r_i \in [-1, 1]$. Such a subset must always exist and it can be constructed by greedily assigning the jobs from R_1 one after another to the machine with lower load.

Now we use the principle of deferred decisions and assume that the index set R_2 is already fixed. Note that the processing requirements of the jobs in R_2 have not been chosen yet. Then each value r_i with $i \in R_2$ is uniformly distributed on $[0, 1-x]$. Since we assume that the failure event $\mathcal{F}_1$ does not occur, we have $|R_2| \geq n/3$. Our goal is now to find a subset $R'_2 \subseteq R_2$ such that the gap $\beta(R'_2) := \sum_{i \in R'_2} r_i - \sum_{i \in R_2 \setminus R'_2} r_i$ is close to $\alpha(R'_1)$ because $\Delta(R'_1 \cup R'_2) = \beta(R'_2) - \alpha(R'_1)$. Lueker [80] studied the *partition gap* of a set of random numbers. Adapted to our notation, he proved that with probability exponentially close to 1, for every $\alpha \in [-1, 1]$ there exists a subset $R'_2 \subseteq R_2$ such that $\beta(R'_2)$ is exponentially close to α. In particular, the probability that there does not exist a subset R'_2 for which $|\beta(R'_2) - \alpha(R'_1)| \leq \bar{p_0}$ goes to 0 for $n \to \infty$. Failure event $\mathcal{F}_2$ is defined to be the event that no such subset R'_2 exists.

If neither $\mathcal{F}_1$ nor $\mathcal{F}_2$ occurs, we consider the scheduling instance with jobs $1, \ldots, 2k$ with sizes $p_1, \ldots, p_{2k}$ and jobs $2k+1, \ldots, n$ with sizes corresponding to the remaining, smaller values r_i. The jobs $2k+1, \ldots, n$ are assigned to the machines 1 and 2 as induced by the aforementioned partition, ensuring that the jobs from the larger class are assigned to machine 1. The jobs $1, \ldots, 2k$ are now assigned to the machines 1 and 2 according to schedule σ_k. As long as one of the jobs $1, \ldots, 2k$ can jump, none of the smaller jobs $2k+1, \ldots, n$ will move due to the Max Weight pivot rule. Hence, as long as the jobs $1, \ldots, 2k$ move, they behave exactly the same as in schedule σ_k where machine 1 is assigned jobs from $\{1, \ldots, 2k\}$ and the additional job p_0 and machine 2 is only assigned jobs from $\{1, \ldots, 2k\}$.

By Lemma 5.11, this takes at least

$$2^{k-1} = 2^{\lfloor nx/4 \rfloor - 1} = 2^{\lfloor \sqrt{n}/12 \rfloor - 1}$$

iterations. This proves the theorem because with constant probability neither $\mathcal{F}_1$ nor $\mathcal{F}_2$ occurs. ◀

5.3 Unit-Weight Jobs

In the case of unit-weight jobs, Even-Dar et al. [48] claimed that for Makespan, there exists a pivot rule which converges in mn steps and that there is a pivot rule with convergence time $\Omega(mn)$ if jobs do not necessarily move to the machine yielding the biggest improvement but only have to improve their costs by jumping. We show that all pivot rules have linear convergence time if jobs have to jump to the best machine.

▶ **Theorem 5.12.** *In both the FIFO and the Makespan model for unit-weight jobs, the convergence time for any pivot rule is n for any number $m \geq 2$ of machines.*

Proof. For the lower bound on two machines, assign all jobs to a machine of speed $1/(2n)$ and set the speed of the other machine to 1. Then all jobs jump to the faster machine.

For the upper bound, we show that every job jumps at most once. Assume to the contrary that there is a job j that jumps twice and let t be the iteration in which job j is the first time able to move again after its first jump. Then in the previous iteration, a job $j' \neq j$ jumped from a machine i_1 to a machine i_2 and job j is now able to jump to machine i_1 as the load of no other machine decreased during the last iteration. Job j cannot be on machine i_2 because it has the same size as job j' and thus we would end up in the same potential as in iteration $t-1$ if job j jumped from machine i_2 to machine i_1. But then job j could also jump to machine i_2 in the previous iteration because

$$c_j^{t-1} = c_j^t > L_{i_1}^t + \frac{1}{s_{i_1}} = L_{i_1}^{t-1} > L_{i_2}^{t-1} + \frac{1}{s_{i_2}}.$$

This contradicts the choice of t. ◀

5.4 Related Machines

For the most general case of related machines we use potential functions in order to show pseudo-polynomial convergence times for different pivot rules in both the FIFO and the Makespan model.

The potential ϕ_{FIFO} used in the FIFO model is the Rosenthal potential introduced in [95], which is the sum of the execution times of the jobs. It is easy to see that ϕ_{FIFO} decreases by at least Δ when the jumping job improves its execution time by Δ. It decreases even more if the jumping job was not on top of its original machine.

For the Makespan model, we use the potential

$$\phi_{\text{Makespan}} := \sum_{i=1}^{m} \frac{1}{s_i} \cdot \left(\left(\sum_{j \in \sigma^{-1}(i)} p_j \right)^2 + \sum_{j \in \sigma^{-1}(i)} p_j^2 \right),$$

defined by Even-Dar et al. [48].

The fastest machine has always load at most $W/s_{\max}$. If there is a machine with load greater than $2W/s_{\max}$, then a job from this machine can improve its costs by at least $W/s_{\max}$ by jumping to the fastest machine. This gives rise to the following lemma.

▶ **Lemma 5.13.** *The following two statements hold:*

1. *If there is a machine with load greater than $2W/s_{\max}$, the best improvement can be achieved by a jump from some job from a machine with load greater than $W/s_{\max}$ to a machine with load at most $W/s_{\max}$.*
2. *If there is no machine with load greater than $2W/s_{\max}$, then $\phi_{FIFO} = O(n \cdot \frac{W}{s_{\max}})$ and $\phi_{Makespan} = O(\frac{W^2}{s_{\max}})$.*

Proof. Let $T := \frac{W}{s_{\max}}$. For any given schedule, we categorize the machines in the following way:

- A machine i is a *Type* 2 machine if $L_i > 2T$.
- A machine i is a *Type* 1 machine if $2T \geq L_i > T$.
- A machine i is a *Type* 0 machine if $T \geq L_i$.

Consider a job j on a machine i that can improve by jumping onto machine i'. If i is a Type 0 machine, then the costs of j are at most T and thus j can only improve by at most T. The costs of the top-most job on any Type 2 machine are greater than $2T$. Therefore, such a job can improve by strictly more than T by jumping onto the fastest machine, where its new costs would be at most $W/s_{\max} = T$.

Second, i' must be a type 0 machine as the new costs of j are at most its costs if it would jump to the fastest machine, which is at most $W/s_{\max} = T$. This proves the first part of the lemma.

If there are no Type 2 machines, then $2T$ is an upper bound for the costs of any job and thus also for the makespan of the current schedule. Then $\phi_{\text{FIFO}} \leq n \cdot 2T = O(n \cdot \frac{W}{s_{\max}})$. The Makespan potential for a schedule σ can be bounded in the following way:

$$\begin{aligned}\phi_{\text{Makespan}} &= \sum_{i=1}^{m} \frac{1}{s_i} \cdot \left(\left(\sum_{j \in \sigma^{-1}(i)} p_j \right)^2 + \sum_{j \in \sigma^{-1}(i)} p_j^2 \right) \leq \sum_{i=1}^{m} \frac{2}{s_i} \cdot \left(\sum_{j \in \sigma^{-1}(i)} p_j \right)^2 \\ &= \sum_{i=1}^{m} \left(2L_i \cdot \sum_{j \in \sigma^{-1}(i)} p_j \right) \leq 4T \cdot \sum_{i=1}^{m} \sum_{j \in \sigma^{-1}(i)} p_j = O(T \cdot W) = O\left(\frac{W^2}{s_{\max}} \right). \quad \blacktriangleleft\end{aligned}$$

▶ **Corollary 5.14.** *For the Best Improvement pivot rule after n iterations and for the Random pivot rule after expected $O(n \log n)$ iterations there is no machine left with load greater than $2W/s_{\max}$.*

5.4.1 FIFO Model

Before we come to the general cases, let us first mention a linear-time result for the special case of $m = 2$ machines.

▶ **Theorem 5.15.** *In the FIFO model, the convergence time for any pivot rule on two related machines is at least n and at most $2n - 2$. There are pivot rules for which $2n - 2$ is tight.*

Proof. The lower bound of n for every pivot rule follows directly from Theorem 5.12.

For the lower bound of $2n - 2$ for some pivot rules, consider two machines with speeds $s_1 = 2$ and $s_2 = 1$ and n jobs with sizes $p_1 = 2, p_2 = \ldots = p_{n-1} = 1/(2n)$, and $p_n = 1$. All jobs are assigned to the slower machine 2 and their permutation π is the identity, i.e., job 1 is the first job to be processed and job n is on top.

Let the jobs $n, \dots, 1$ jump from machine 2 to machine 1 in this order. This is valid as $(\sum_j p_j)/s_1 < 2 = p_1/s_2$. Then let the jobs $2, \dots, n-1$ jump back to machine 2. This is valid as $(n-2)/(2n) < 1/2 = p_n/s_1$ and n is the first job on machine 1.

For the upper bound, consider the setting of two machines with $s_1 \geq s_2$. We claim that after a job jumped from the faster machine 1 to the slower machine 2, no job can jump in the other direction from machine 2 to machine 1. Then there are at most n jumps from machine 2 to machine 1 and at most $n-2$ jumps from machine 1 to machine 2 as the first job on the faster machine 1 does never jump to machine 2 and the last job leaving machine 2 does also not jump back.

In order to show the claim, let a job j jump from machine 1 to machine 2 in iteration t and let j' be another job on machine 2. Then,

$$c_{j'}^{t+1} = c_{j'}^{t} \leq L_2^t < L_1^t - \frac{p_j}{s_2} = L_1^{t+1} + \frac{p_j}{s_1} - \frac{p_j}{s_2} \leq L_1^{t+1} \leq L_1^{t+1} + \frac{p_{j'}}{s_1},$$

i.e., job j' cannot improve its costs by jumping. The strict inequality comes from the fact that job j improved its costs by jumping in iteration t. ◀

The main idea of the following proofs is that if a job jumps that is not on top of its machine, the costs of all jobs above the moving job and thus the potential ϕ_{FIFO} decrease by at least $p_{\min}/s_{\max}$. We are able to show that this must happen after a polynomial number of steps for the Best Improvement and for Fixed Priority pivot rules.

▶ **Theorem 5.16.** *In the FIFO model, the convergence time of the Best Improvement pivot rule is in* $O(m^2 n \cdot W/p_{\min})$.

Proof. According to Lemma 5.13 and Corollary 5.14, after $O(n)$ iterations the potential ϕ_{FIFO} is in $O(n \cdot W/s_{\max})$. Hence, it suffices to show that in every sequence of m^2 consecutive iterations, ϕ_{FIFO} drops by at least $p_{\min}/s_{\max}$. Therefore, let us consider a sequence S of maximum length in which ϕ_{FIFO} drops by strictly less than $p_{\min}/s_{\max}$. It is obvious that only jobs that are on top of some machine can jump as the running times of all the jobs above the moving job decrease by at least $p_{\min}/s_{\max}$.

For a given point in time, we call a job *active* if it jumps until the end of the sequence S. At any time, there can only be at most one active job on any machine. To see this, assume to the contrary that there are two active jobs j_1 and j_2 at the same time t_1 on a machine i. Let job j_1 w.l.o.g. be directly above job j_2, and let $t_2 > t_1$ be the first iteration in which job j_2 leaves machine i again. Define $\alpha := c_{j_1}^{t_1} - c_{j_1}^{t_2}$ as the difference of j_1's running times at time t_1 and t_2. As job j_2 was a top-most job in iteration t_2 and no job below j_2 could jump before j_2 jumped, job j_1 would have a running time of $L_i^{t_1} - p_{j_2}/s_i$ if it jumped to machine i in the next step, yielding a total improvement of j_1's running time of at least $p_{\min}/s_{\max}$. If j_1 does not jump back to machine i in the next step, then either we have reached an equilibrium (then $p_{j_2}/s_i \leq \alpha$) or there is a job (possibly also j_1) that can improve by strictly more than $p_{j_2}/s_i - \alpha$. Hence, the potential drops by at least $p_{j_2}/s_i \geq p_{\min}/s_{\max}$ during all the jumps of job j_1 between t_1 and t_2 and the iteration following $t_2 + 1$.

Thus, we have shown that also at the beginning of the sequence S there are at most m active jobs as on each machine there is at most one active job. It also implies that no job j can jump back to a machine i it has already been onto as all jobs lying underneath j stay on machine i until the end of the sequence S. Hence, every job jumps at most $m-1$ times and the length of S is bounded from above by $m(m-1)$. ◀

▶ **Theorem 5.17.** *In the FIFO model, the expected convergence time of the Random pivot rule is in* $O(m^2 n^2 \cdot W/p_{\min})$.

Proof. Due to Corollary 5.14, after an expected number of $O(n \log n)$ iterations, the potential ϕ_{FIFO} is bounded by $O(nW/s_{\max})$. As shown in the proof of Theorem 5.16, after at most $m(m-1)$ iterations there is a job j such that ϕ_{FIFO} decreases by a total of at least $p_{\min}/s_{\max}$ in this sequence if job j jumps next. Hence, after an expected number of $O(n)$ such sequences the potential ϕ_{FIFO} drops by this amount. ◀

For Fixed Priority pivot rules, we cannot assume anymore that after a linear number of iterations there is no machine with load more than $2W/s_{\max}$ left and thus that ϕ_{FIFO} is small. On the other hand, we know that the sum of the running times of all jobs that have already jumped is bounded by $O(n \cdot W/s_{\max})$ and we are able to show that during $O(n)$ consecutive iterations, either a job jumps for the first time or the potential ϕ_{FIFO} drops by at least $p_{\min}/s_{\max}$. In order to bound the potential by $O(n \cdot W/s_{\max})$, we use the modified potential function

$$\phi'_{\text{FIFO}} := \sum_{j=1}^{n} \min\left\{c_j, \frac{W}{s_{\max}}\right\}.$$

▶ **Theorem 5.18.** *In the FIFO model, the convergence time of any Fixed Priority pivot rule is in $O(n^2 \cdot W/p_{\min})$.*

Proof. As $0 \leq \phi'_{\text{FIFO}} \leq n \cdot W/s_{\max}$, we only have to show that during every sequence of $n+1$ steps, either ϕ'_{FIFO} drops by at least $p_{\min}/s_{\max}$ or a job must jump for the very first time. In such a sequence, it must be the case that a job j_2 jumps directly after a job j_1, where the priority of j_2 is greater than the priority of j_1. This means that j_2 jumps to the old machine i of job j_1 as it could not jump before the move of j_1. If it was not j_1's first jump, let t_2 be the point in time between the two jumps by j_1 and j_2, and let t_1 be the point in time before j_1 jumps the last time before $t_2 - 1$. As j_2 does not want to jump to machine i at time t_1, but does this later at time t_2, it must be the case that $L_i^{t_1} > L_i^{t_2}$. Hence, between $t_1 + 1$ and $t_2 - 1$ a job j' assigned to machine i at time t_1 must leave its machine. But during this time, job j_1 lies above job j' yielding a running time improvement of $p_{j'}/s_i \geq p_{\min}/s_{\max}$ for job j_1 through the jump by j'. As j_1 has jumped before, its running time before the jump by j' was already at most $W/s_{\max}$, meaning that also ϕ'_{FIFO} drops by at least $p_{\min}/s_{\max}$. ◀

The machine speeds do not occur in our bounds for the convergence times. Nevertheless, different machine speeds result in a higher convergence time than in the case of identical machines, as the following result shows. We believe that our proofs for the upper bounds on the convergence times are too pessimistic and thus we conjecture polynomial convergence times for all pivot rules. This is in contrast to the superpolynomial lower bounds in the Makespan and SJF model but a crucial difference is that the costs of a job can never increase in the FIFO model.

▶ **Theorem 5.19.** *In the FIFO model, local search can take $\Omega(mn)$ steps. The convergence time for the Min Weight pivot rule is in $\Omega(m^2)$.*

Proof. For the lower bound $\Omega(mn)$, let $\ell \geq 1$ and $k \geq 1$ be two integers. There are $m = 2k+1$ machines and $n = k\ell + k + 1$ jobs split up in $2k+1$ job classes $J_1, \ldots, J_{2k+1}$. The machine speeds are $s_i = 2^{i-1}$ for $1 \leq i \leq 2k$ and $s_{2k+1} = 2^{2k+1}$. The job classes $J_1, \ldots, J_k$ each contain ℓ jobs with sizes $2^0, \ldots, 2^{\ell-1}$ and the job classes $J_{k+1}, \ldots, J_{2k+1}$ each contain a single job with size $2^{\ell+j}$ for job class J_j.

Initially, each job class J_j is assigned to machine j and the jobs on a machine are processed in monotonically increasing order of the job sizes. We consider the following k rounds $1, \ldots, k$.

Before round i begins, the jobs from job class J_j, $j \leq k$, are on machine $j+i-1$ such that they are processed in increasing order of the sizes, the jobs from job classes $J_{k+1}, \ldots, J_{k+i-1}$ are on machine $2k+1$ and the other jobs have not moved before. Then we let the single job from class J_{k+i} move from machine $k+i$ to machine $2k+1$. Thereupon, the jobs from class J_k move in ascending order of the sizes from machine $k+i-1$ to machine $k+i$, the jobs from class J_{k-1} move in ascending order of the sizes from machine $k+i-2$ to machine $k+i-1$ and so on. One can easily see that every job strictly decreases its costs while moving. All jobs from the job classes $J_1, \ldots, J_k$ move in every round. Hence, there are $\Omega(k^2\ell) = \Omega(mn)$ iterations.

For the lower bound $\Omega(m^2)$ for the Min Weight pivot rule, let again k be an integer and let $\varepsilon > 0$ be appropriately small. There are $m = n = 2k+1$ machines and jobs. The machine speeds are $s_i = 1 + i \cdot \varepsilon$ for $1 \leq i \leq 2k$ and $s_{2k+1} = 4k$. The job sizes are $p_j = 1 - j \cdot \varepsilon$ for $1 \leq j \leq k$, $p_j = 2 + 2j \cdot \varepsilon$ for $k+1 \leq j \leq 2k$, and $p_{2k+1} = 4k$. Initially, every job j is assigned to machine j and the loads on the first k machines are less than 1, $L_{k+1} = \ldots = L_{2k} = 2$ and $L_{2k+1} = 1$. One can easily see that every job $k+1, \ldots, 2k$ can move to machine $2k+1$ as L_{2k+1} remains to be less than 2 and that every such jump induces jumps from the jobs $1, \ldots, k$. Hence, there are $\Omega(k^2) = \Omega(m^2)$ iterations. ◀

5.4.2 Makespan Model

In this subsection, we consider the Best Improvement pivot rule in the Makespan model. We use the fact that the potential ϕ_{Makespan} decreases by at least $2p_{\min} \cdot p_{\min}/s_{\max}$ if a sequence of jobs decrease their running time by a total of $p_{\min}/s_{\max}$ through jumping. This is due to a lemma by Even-Dar et al. [48] that if a jumping job j improves its execution time by Δ, then ϕ_{Makespan} drops exactly by $2p_j\Delta$.

Suppose that a job j wants to jump away from machine i to machine i' and there is a smaller job j' on machine i. At the current time, the costs of j and j' are the same as they are on the same machine. But the additional costs job j' would generate on any machine are strictly smaller than the additional costs job j would generate. Hence, job j' would have smaller costs on machine i' than job j. This leads to the following observation.

▶ Observation 5.20. When a job jumps away from a machine i according to the Best Improvement pivot rule, it was a smallest job on machine i.

Let us now provide the main ideas of our proof. Imagine there are two jobs j_1, j_2 on the same machine i and job j_1 jumps away in iteration t_1 making a small improvement directly before job j_2 leaves machine i in iteration $t_2 = t_1 + 1$. Then job j_1 could improve its running time by p_{j_2}/s_i by jumping back to machine i in iteration $t_2 + 1$. If, however, $t_2 > t_1 + 1$, it could happen that another job j_3 from job j_1's new machine leaves this machine leaving job j_1 unable to jump back. But then job j_3 is smaller than j_1 according to Observation 5.20 and thus could jump to machine i in iteration t_2+1 unless it already made a big improvement or another job from job j_3's new machine jumped away in the meantime etc. Lemma 5.21 proves that the potential drops significantly during such a sequence.

▶ **Lemma 5.21.** *If two jobs jump away from a machine i at iterations $t < t'$ and no job enters machine i between t and t', then the potential $\phi_{Makespan}$ drops by at least $p_{\min}^2/s_{\max}$ during the iterations $t, \ldots, t'+1$ when using the Best Improvement pivot rule.*

In order to prove Lemma 5.21, we introduce the following terminology.

▶ **Definition 5.22.** Let $(j_1, \ldots, j_\ell), (i_0, \ldots, i_\ell)$, and $t_1 < \ldots < t_\ell$ be sequences such that for any $k = 1, \ldots, \ell$, job j_k jumps from machine i_{k-1} to machine i_k in iteration t_k. These sequences are called

1. *forward thread* if from iteration $t_k + 1$ to iteration $t_{k+1} - 1$, no job leaves machine i_k for every $k = 1, \ldots, \ell - 1$.
2. *backward thread* if from iteration $t_k + 1$ to iteration $t_{k+1} - 1$, no job enters machine i_k for every $k = 1, \ldots, \ell - 1$.

Intuitively, a forward thread always follows the first job leaving a machine and a backward thread backtracks the last job entering a machine in the past.

▶ **Lemma 5.23.** *Consider a forward thread* $((j_1, \ldots, j_\ell), (i_0, \ldots, i_\ell), t_1 < \ldots < t_\ell)$*. Then* $p_{j_1} \geq \ldots \geq p_{j_\ell}$ *and the potential* $\phi_{Makespan}$ *decreases in total by at least* $2(L_{i_0}^{t_1} - L_{i_\ell}^{t_\ell+1}) \cdot p_{\min}$ *in the iterations* $t_1, \ldots, t_\ell$*.*

Proof. The job sizes are monotonically decreasing because at iteration t_k for $2 \leq k \leq \ell$, job j_{k-1} is also on machine i_{k-1} when job j_k leaves this machine and only the smallest job on a machine is able to leave according to Observation 5.20.

The improvement of the jumping job in iteration t_k is given by $L_{i_{k-1}}^{t_k} - L_{i_k}^{t_k+1}$ leading to a potential drop of at least $2(L_{i_{k-1}}^{t_k} - L_{i_k}^{t_k+1}) \cdot p_{\min}$. As no job leaves machine i_k between the iterations $t_k + 1$ and $t_{k+1} - 1$, it holds $L_{i_k}^{t_k+1} \leq L_{i_k}^{t_{k+1}}$. By summing over all possible choices for k, we attain the desired bound. ◀

Proof of Lemma 5.21. Let job j jump away from machine i at iteration t and let job j' jump away from machine i at iteration t' such that no job enters machine i in the meantime. W.l.o.g., also no job left machine i in the meantime. Let $(j = j_1, \ldots, j_\ell), (i = i_0, \ldots, i_\ell)$, and $t = t_1 < \ldots < t_\ell$ be the maximum forward thread starting in iteration t with $t_\ell < t'$. If $L_{i_\ell}^{t_\ell+1} \leq L_i^t - p_{\min}/(2s_{\max})$, the potential drop follows from Lemma 5.23. Otherwise, the potential drop gained by letting job j_ℓ jump from machine i_ℓ to machine i at iteration $t' + 1$ is at least

$$L_{i_\ell}^{t'+1} - \left(L_i^{t'+1} + \frac{p_{j_\ell}}{s_i}\right) \geq L_{i_\ell}^{t_\ell+1} - \left(L_i^{t'+1} + \frac{p_{j_\ell}}{s_i}\right) > L_i^t - L_i^{t'+1} - \frac{p_{j_\ell}}{s_i} - \frac{p_{\min}}{2s_{\max}}$$
$$= \frac{p_j + p_{j'} - p_{j_\ell}}{s_i} - \frac{p_{\min}}{2s_{\max}},$$

where the first inequality stems from the maximality of the chosen thread, i.e., no job left machine i_ℓ in the meantime. Again by Lemma 5.23, it holds $p_j \geq p_{j_\ell}$ and thus the improvement for job j_ℓ would then be at least $p_{\min}/(2s_{\max})$. The Best Improvement pivot rule chooses a job in iteration $t' + 1$ that gains at least that much leading to a potential drop of at least $p_{\min}^2/s_{\max}$. ◀

Imagine now there are two jobs j_1, j_2 entering the same machine i in two consecutive iterations t_1 and $t_2 = t_1 + 1$, where job j_1 moves first. Then job j_2 would improve its running time by at least p_{j_1}/s_i if it jumped in iteration t_1 as it also has the incentive to move to machine i after job j_1's jump. But if $t_2 > t_1 + 1$, it could be that in iteration t_1 job j_2's running time is smaller than in iteration t_2 and in the meantime another job j_3 enters job j_2's machine. If job j_3 is much larger than job j_2, then job j_2 would improve much by jumping to job j_3's old machine. Otherwise, job j_3 could have moved to machine i in iteration t_1 unless another job entered job j_3's old machine in the meantime etc. Lemma 5.24 shows that also in this case the potential drops significantly.

▶ **Lemma 5.24.** *If two jobs enter a machine* i *at iterations* $t' < t$ *and no job leaves machine* i *between* t' *and* t*, then the potential* $\phi_{Makespan}$ *drops by at least* $p_{\min}^2/(2 \cdot s_{\max})$ *between* t' *and* $t + 1$ *when using the Best Improvement pivot rule.*

Let us again first show an auxiliary lemma.

▶ **Lemma 5.25.** *Consider a backward thread $((j_1,\dots,j_\ell),(i_0,\dots,i_\ell),t_1<\dots<t_\ell)$. Then the potential $\phi_{Makespan}$ decreases in total by at least $(p_{j_1}-p_{j_\ell})\cdot p_{\min}/s_{\max}$ in the iterations $t_1,\dots,t_\ell+1$ when using the Best Improvement pivot rule.*

Proof. Let $1\le k\le \ell$ and consider the iterations t_k and t_k+1. If job j_{k+1} (which could also be equal to job j_k) jumped from machine i_k to machine i_{k-1} in iteration t_k+1, then the total improvement for the two jumping jobs would be

$$\begin{aligned}&L_{i_{k-1}}^{t_k}-\left(L_{i_k}^{t_k}+\frac{p_{j_k}}{s_{i_k}}\right)+L_{i_k}^{t_k+1}-\left(L_{i_{k-1}}^{t_k+1}+\frac{p_{j_{k+1}}}{s_{i_{k-1}}}\right)\\ =&L_{i_{k-1}}^{t_k}-\left(L_{i_{k-1}}^{t_k+1}+\frac{p_{j_{k+1}}}{s_{i_{k-1}}}\right)=\frac{p_{j_k}-p_{j_{k+1}}}{s_{i_{k-1}}},\end{aligned}$$

where we used that $L_{i_k}^{t_k}+\frac{p_{j_k}}{s_{i_k}}=L_{i_k}^{t_k+1}$ and $L_{i_{k-1}}^{t_k+1}=L_{i_{k-1}}^{t_k}-\frac{p_{j_k}}{s_{i_{k-1}}}$. This leads to a potential drop of ϕ_{Makespan} by at least $2(p_{j_k}-p_{j_{k+1}})\cdot p_{\min}/s_{\max}$. We do not know whether this term is positive and hence whether this move by job j_{k+1} is legal, but we know that the job that jumps in iteration t_k+1 improves by at least as much as job j_{k+1} would improve by jumping to machine i_{k-1}. By summing over all possible choices for k, we count every iteration at most twice leading to the lower bound of $(p_{j_1}-p_{j_\ell})\cdot p_{\min}/s_{\max}$ for the total decrease of ϕ_{Makespan}. ◀

Proof of Lemma 5.24. Let job j' enter machine i at iteration t' and let job j enter machine i at iteration t such that no job leaves machine i in the meantime. W.l.o.g., also no job enters machine i in the meantime. Let $(j_1,\dots,j_\ell=j),(i_0,\dots,i_\ell=i)$, and $t_1<\dots<t_\ell=t$ be the maximum backward thread ending in iteration t with $t_1>t'$. As job j_1 jumps from machine i_0 to machine i_1 in iteration t_1 and no other job enters machine i_1 from then on until iteration t_2, it holds $L_{i_0}^{t_1}>L_{i_1}^{t_1}+\frac{p_{j_1}}{s_{i_1}}\ge L_{i_1}^{t_2}$. Reiterating this argument yields $L_{i_0}^{t_1}>L_i^t+\frac{p_j}{s_i}$. As the chosen sequence is maximal, it also holds $L_{i_0}^{t'}\ge L_{i_0}^{t_1}>L_i^t+\frac{p_j}{s_i}=L_i^{t'}+\frac{p_{j'}+p_j}{s_i}$ and job j_1 is on machine i_0 during iteration t'. If $p_{j_1}>p_j+p_{\min}/2$, the potential drop follows from Lemma 5.25. Otherwise,

$$L_{i_0}^{t'}\ge L_i^{t'}+\frac{p_{j'}+p_{j_1}-p_{\min}/2}{s_i}\ge L_i^{t'}+\frac{p_{j_1}}{s_i}+\frac{p_{\min}}{2s_{\max}},$$

i.e., job j_1 would improve by at least $p_{\min}/(2s_{\max})$ by jumping from machine i_0 to machine i in iteration t'. Hence, the job j' moving in iteration t' must decrease its running time by at least as much yielding a potential drop of at least $p_{\min}^2/s_{\max}$. ◀

Hence, we are able to show that if there is a machine to which two jobs migrate without a job leaving or from which two jobs leave without a job entering, the potential ϕ_{Makespan} drops significantly. The proof then concludes with the observation that this must happen every $O(m^2n)$ iterations.

▶ **Theorem 5.26.** *In the Makespan model, the convergence time of the Best Improvement pivot rule is in $O(m^2n\cdot W^2/p_{\min}^2)$.*

Proof. According to Lemma 5.13, after $O(n)$ iterations the potential ϕ_{Makespan} is in $O(W^2/s_{\max})$. Hence, it suffices to show that in every sequence of m^2n consecutive iterations, ϕ_{Makespan} drops by at least $p_{\min}^2/(2s_{\max})$.

Let S be a sequence of maximum length such that ϕ_{Makespan} drops by less than $p_{\min}^2/(2s_{\max})$, lasting from iteration t_0 to iteration t_ℓ. We maintain a set of indices, which is empty at time t_0. When a job j jumps from a machine i_1 to a machine i_2 at iteration $t \in \{t_0, \dots, t_\ell\}$ and if there has not been a job that jumped to machine i_1 during the iterations $t_0, \dots, t$, generate a new index which gets attached to machine i_2. Otherwise, reattach the index previously attached to machine i_1 to machine i_2. Lemma 5.21 shows that this is well-defined as there cannot be another job leaving machine i_1 before another index gets attached to this machine.

At the end of the sequence, there can only be at most m indices. If an index gets reattached from machine i_1 to machine i_2 at iteration t, then $L_{i_1}^t > L_{i_2}^{t+1}$, i.e., the running time of the machine an index is attached to is strictly monotonically decreasing.

Consider an index that jumps with job j at iteration t and with job j' at iteration t' to the same machine i. Let $j = j_1, j_2, \dots, j_\ell$ be the jobs that entered machine i and let $j'_1, \dots, j'_\ell$ be the jobs that left machine i during the iterations $t, \dots, t'-1$ in this order. Lemma 5.21 and Lemma 5.24 show that the order in which this happened must be $j_1, j'_1, j_2, j'_2, \dots, j_\ell, j'_\ell$ and that the sequences have the same length, i.e., the sequences are well-defined. As always only a smallest job on a machine is able to achieve the best improvement and for every k, job j_k is on machine i when job j'_k leaves this machine, it must be the case that $L_i^t \leq L_i^{t'}$. But in the iterations $t+1$ and $t'+1$, the same index is attached to machine i, meaning that $L_i^t + p_j/s_i = L_i^{t+1} > L_i^{t'+1} = L_i^{t'} + p_{j'}/s_i$, i.e., $p_j > p_{j'}$. This means that an index cannot be attached twice to the same machine by a jump of the same job and thus an index gets reattached at most $n \cdot m$ times. This concludes the proof. ◀

5.4.3 SJF Model

▶ **Theorem 5.27.** *In the SJF model, the convergence time of the Min Weight pivot rule is exactly n, even on two machines. The expected convergence time of the Random pivot rule is less than n^2.*

Proof. W.l.o.g., $p_1 \leq \ldots \leq p_n$. When the Min Weight pivot rule selects a job j to jump, all jobs $1, \dots, j-1$ do not want to jump. As larger jobs do not affect the decision whether a job wants to jump, the jobs $1, \dots, j-1$ do not want to jump to job j's old machine afterwards and thus, the jobs $1, \dots, j$ will never jump again. Hence, every job jumps at most once and the convergence time is at most n. On the other hand, every job jumps if all jobs are initially assigned to an arbitrarily slow machine.

For the analysis of the Random pivot rule, let j be the smallest unsatisfied job. It takes an expected number of $O(n)$ iterations until job j gets selected to jump. Afterwards, it will not jump again. By linearity of expectation, the convergence time for the Random pivot rule is then in $O(n^2)$. ◀

5.5 Price of Anarchy for FIFO

Brunsch et al. [16] already showed that the smoothed price of anarchy for near list schedules in the Makespan model, which correspond to local optima in the FIFO model, is $\Theta(\log \phi)$. We give matching bounds for the deterministic case.

▶ **Theorem 5.28.** *In the FIFO model, the price of anarchy for local search is $\Theta(\log m)$ on related machines and $2 - 1/m$ on identical machines.*

Proof. It is easy to see that every list schedule is a local optimum w.r.t. the FIFO model. Hence, the lower bounds $\Omega(\log m)$ by Aspnes et al. [6] for related machines and $2 - 1/m$ by

Graham [57] for identical machines translate to lower bounds for local search in the FIFO model. Graham's proof for the upper bound of $2 - 1/m$ can be applied to the FIFO model without any changes.

For the upper bound for related machines, we use the notation of *near list schedules* defined in the Makespan model by Brunsch et al. [16]:

▶ **Definition 5.29.** We call a schedule σ on machines $1, \dots, m$ with speeds $s_1, \dots, s_m$ a *near list schedule*, if we can index the jobs in such a way that

$$L_{i'} + \frac{p_j}{s_{i'}} \geq L_i - \sum_{\ell \in \sigma^{-1}(i):\ \ell < j} \frac{p_\ell}{s_i} \tag{4}$$

for all machines $i' \neq i$ and all jobs $j \in J_i(\sigma)$.

This definition is equivalent to the definition of a locally optimal schedule w.r.t the FIFO model if one inverses the order of the jobs. Hence, in order to bound the price of anarchy of local search in the FIFO model, we can use their results for near list schedules in the Makespan model.

W.l.o.g., $s_1 \geq \ldots \geq s_m$ and let the makespan of an optimal schedule be exactly 1. Let σ be a near list schedule with a makespan of strictly less than $c+2$ for some integer c, and let i^* be the fastest machine with $L^\sigma_{i^*} < 2$. Brunsch et al. [16] showed that the total processing requirement on the machines $i^*, \dots, m$ in any optimal schedule is at least $(c-1) \cdot s_1$ (cf. Lemma 9 and Lemma 13 with $k = t = 2$). On the other hand, they showed that $s_1 \geq 2^{\lfloor (c-1)/6 \rfloor} \cdot s_{i^*}$ (cf. Lemma 15 with $i_1 = 1$ and $i_2 = i^*$). Since in any optimal schedule, the running times of the machines $i^*, \dots, m$ are at most 1 and the machine speeds are monotonically decreasing, it holds

$$m \geq m - i^* + 1 \geq (c-1) \cdot 2^{\lfloor (c-1)/6 \rfloor},$$

i.e., $c = O(\log m)$ and thus the price of anarchy for FIFO is in $O(\log m)$. ◀

6 Signed Max-Cut Above Edwards-Erdős Bound

We now come to the first kernelization results. We consider the Signed Max-Cut problem parameterized above the Edwards-Erdős Bound (Signed Max-Cut AEE). After the Preliminaries in Subsection 6.1, we discuss in Subsection 6.2 that Signed Max-Cut AEE can be solved in linear time, i.e., we show Theorem 2.5. Subsection 6.3 is dedicated to Signed Max-Cut AEE kernelization resulting in Theorem 2.6.

6.1 Preliminaries

Since in this and the next section we discuss general properties on possibly oriented and/or labelled graphs, we choose to define the involved terms more rigorously for this part. We use $\uplus$ to denote the disjoint union of sets. The term "graph" refers to finite undirected graphs without self-loops, parallel edges, edge directions, or labels. For a graph G, let $V(G)$ denote its set of vertices and let $E(G)$ denote its set of edges. In an oriented graph, each edge $e = \{u, v\}$ has one of two directions, $\overrightarrow{e} = (u, v)$ and $\overleftarrow{e} = (v, u)$; thus, an oriented graph is a digraph without 2-cycles and loops. Distinct vertices a, b, c are said to induce a *triangle* (a, b, c) if they form a complete subgraph. In a labelled graph, each edge in $E(G)$ receives one of a constant number of labels. For an oriented and/or labelled graph G, let $\langle G \rangle$ denote the underlying simple graph obtained from omitting orientations and/or labels. Throughout the section, we assume graphs to be encoded as adjacency lists.

A graph is *connected* if there is a path between any two of its vertices. A *connected component* of G is a maximal connected subgraph of G. A *cut vertex* of a graph G is a vertex whose removal increases the number of connected components. A graph is *2-connected* if it does not contain any cut vertices. A maximal 2-connected subgraph of a graph G is called a *block* or a *2-connected component* of G. A block that contains at most one cut vertex of G is called a *leaf block* of G. A *clique tree* is a connected graph whose blocks are *cliques*, where a clique is a complete subgraph of a graph. A *clique forest* is a graph whose connected components are clique trees.[2] For an oriented and/or labelled graph G we say that G has one of the above-defined properties if $\langle G \rangle$ does.

Let G be a graph. For a vertex subset $X \subset V(G)$, the *(vertex-)induced subgraph* $G[X]$ is the graph with vertex set X whose edge set consists of all the edges of G with both endpoints in X. Similarly, we define $G - X = G[V(G) \setminus X]$ for a vertex subset $X \subseteq V(G)$ and $G - x = G - \{x\}$ for a vertex $x \in V(G)$.

For a vertex $v \in V(G)$, let $N_G(v) = \{u \in V(G) \mid \{u, v\} \in E(G)\}$. For signed graphs G, we define $N_G(v) = N_{\langle G \rangle}(v)$. For a vertex set $V' \subseteq V(G)$, let $N_G(V') = (\bigcup_{v \in V'} N_G(v)) \setminus V'$. For disjoint vertex sets $V_1, V_2 \subseteq V(G)$, let $E(V_1, V_2)$ denote the set of edges with one endpoint in V_1 and the other endpoint in V_2. For a signed graph G, let $E^+(G) \subseteq E(G)$ be the edges with positive labels, and $E^-(G) = E(G) \setminus E^+(G)$ be the edges with negative labels. Define $N_G^+(v) = \{u \in V(G) \mid \{v, u\} \in E^+(G)\}$ and $N_G^-(v) = \{u \in V(G) \mid \{v, u\} \in E^-(G)\}$ for all $v \in V(G)$. A sequence of vertices $(v_0, v_1, \ldots, v_\ell)$ is a *path* in G if $v_0, v_1, \ldots, v_\ell$ are distinct vertices of G and $\{v_i, v_{i+1 \pmod \ell}\} \in E(G)$ for $i = 0, \ldots, \ell$. For vertices $u, v \in V(G)$, a $[u, v]$-path is a path in G between u and v. A path is *induced* if additionally $\{v_i, v_j\} \notin E(G)$ for $i = 0, \ldots, \ell$ and $j \neq i+1 \pmod \ell$. The *length* of a path is the number of edges it contains, and an *ℓ-path* is a path of length ℓ.

[2] Clique forests are sometimes called *block graphs*; however, there are competing definitions for this term in the literature and so we refrain from using it.

Given a λ-extendible property Π, we define the *excess* of G over the lower bound $\mathsf{pt}(G)$ with respect to Π as $\mathsf{ex}(G) = \max\{|E(H)| - \mathsf{pt}(G) \mid H \subseteq G, H \in \Pi\}$. When considering properties of labelled and/or oriented graphs, we denote by $\mathsf{ex}(K_t)$ the minimum value of $\mathsf{ex}(G)$ over all labelled and/or oriented graphs G with $\langle G \rangle = K_t$; here, K_t denotes the complete graph of order t.

A strongly λ-extendible property Π *diverges on cliques* if $\mathsf{ex}(K_j) > \frac{1-\lambda}{2}$ for some $j \in \mathbb{N}$. For example, every strongly λ-extendible property with $\lambda \neq \frac{1}{2}$ diverges on cliques [26]. We recall the following fact about diverging properties:

▶ Proposition 6.1 ([26, Lemmas 7-8]). Let Π be a strongly λ-extendible property diverging on cliques, and let $j \in \mathbb{N}, a > 0$ be such that $\mathsf{ex}(K_j) = \frac{1-\lambda}{2} + a$. Then $\mathsf{ex}(K_i) \geq ra$ for all $r \in \mathbb{N}$ and $i \geq rj$.

We will need the following proposition. For Signed Max-Cut, we will apply it with $\lambda = \frac{1}{2}$.

▶ Proposition 6.2 ([26, Lemma 6]). Let Π be a strongly λ-extendible property, let G be a connected graph and let $U_1 \uplus U_2$ be a partition of $V(G)$ into non-empty sets U_1, U_2. For $i \in \{1, 2\}$ let c_i be the number of connected components of $G[U_i]$. If $\mathsf{ex}(G[U_i]) \geq k_i$ for some $k_i \in \mathbb{R}$ and $i \in \{1, 2\}$, then $\mathsf{ex}(G) \geq k_1 + k_2 - \frac{1-\lambda}{2}(c_1 + c_2 - 1)$.

6.2 A Linear-Time Fixed-Parameter Algorithm for Signed Max-Cut AEE

In this subsection we show that the fixed-parameter algorithm given by Crowston et al. [23] for the Signed Max-Cut AEE problem can be implemented so as to run in time $8^k \cdot O(|E(G)|)$. That is, given a connected graph G whose edges are labelled either positive (+) or negative (−), and an integer k, we can decide in time $8^k \cdot O(|E(G)|)$ whether G has a balanced subgraph with at least $|E(G)|/2 + (|V(G)| - 1 + k)/4$ edges. This will prove Theorem 2.5.

We build on the following classical characterization of signed graphs:

▶ Proposition 6.3 (Harary [62]). A signed graph G is balanced if and only if there exists a partition $V_1 \uplus V_2 = V(G)$ such that all edges in $G[V_1]$ and $G[V_2]$ are positive and all edges $E(V_1, V_2)$ between V_1 and V_2 are negative.

The algorithm by Crowston et al. [23] starts by applying the following seven reduction rules. We restate them here, as they are crucial for our results. A reduction rule is *1-safe* if, on input (G, k) it returns a pair (G', k') such that (G, k) is a "yes"-instance for Signed Max-Cut AEE if (G', k') is. (Note that the converse direction does not have to hold.) In a signed graph G we call a triangle *positive* if its number of negative edges is even.

In the description of the rules, G is always a connected signed graph and C is *always* a clique without positive triangles. We initialize an empty set S of *marked* vertices. (Note: In previous work the term *selected* vertices was also used, so we stick to the set name S.)

▶ Reduction Rule 1. If (a, b, c) is a positive triangle such that $G - \{a, b, c\}$ is connected, add a, b, c to S and delete them from G, and set $k' = k - 3$.

▶ Reduction Rule 2. If (a, b, c) is a positive triangle such that $G - \{a, b, c\}$ has exactly two connected components C and Y, then add a, b, c to S and delete them from G, delete C, and set $k' = k - 2$.

▶ Reduction Rule 3. Let C be a connected component of $G - v$ for some vertex $v \in V(G)$. If there exist $a, b \in V(C)$ such that $G - \{a, b\}$ is connected and there is an edge $\{a, v\}$ but no edge $\{b, v\}$, then add a, b to S and delete them from G, and set $k' = k - 2$.

▶ Reduction Rule 4. Let C be a connected component of $G - v$ for some vertex $v \in V(G)$. If there exist $a, b \in V(C)$ such that $G - \{a, b\}$ is connected and (a, b, v) is a positive triangle, then add a, b to S and delete them from G, and set $k' = k - 4$.

▶ Reduction Rule 5. If there is a vertex $v \in V(G)$ such that $G - v$ has a connected component C such that $G[V(C) \cup \{v\}]$ is a clique without positive triangles, then delete C. If $|V(C)|$ is odd, set $k' = k - 1$; otherwise, set $k' = k$.

▶ Reduction Rule 6. If $a, b, c \in V(G)$ induce a 2-path (a, b, c) such that $G - \{a, b, c\}$ is connected, then add a, b, c to S and delete them from G, and set $k' = k - 1$.

▶ Reduction Rule 7. Let C, Y be the connected components of $G - \{v, b\}$ for some $v, b \in V(G)$ such that $\{v, b\} \notin E(G)$. If $G[V(C) \cup \{v\}]$ and $G[V(C) \cup \{b\}]$ are cliques without positive triangles, then add v, b to S and delete them from G, delete C, and set $k' = k - 1$.

Note: Rules 1/2/4 require positive edges. Hence, the other four rules suffice to handle the classical Max-Cut AEE problem, where all edges are negative. We will make use of this in Section 7.

We will call the vertex v of Rule 5 the *anchor* of the removed vertex set $V(C)$.

We slightly changed Rule 5. Crowston et al. [23] always set $k' = k$, whereas we set $k' = k-1$ when $|V(C)|$ is odd. In this case, $\mathsf{pt}(G[V(C) \cup \{v\}])$ cannot be integral because $|V(C) \cup \{v\}|$ is even, and thus $\mathsf{ex}(G[V(C) \cup \{v\}]) \geq \frac{1}{4}$. Therefore, our change for k is 1-safe by the following result.

▶ Proposition 6.4 ([23, Lemma 2]). Let G be a connected signed graph and let Z be a connected component of $G - v$ for some $v \in V(G)$. Then $\mathsf{ex}(G) = \mathsf{ex}(G - Z) + \mathsf{ex}(G[V(Z) \cup \{v\}])$.

We subsume the results by Crowston et al. [23] in the following proposition.

▶ Proposition 6.5 ([23]). Rules 1-7 are 1-safe. To any connected signed graph G with at least one edge, one of these rules applies and the resulting graph is connected. For the set S of vertices marked during the exhaustive application of Rules 1-7, $G - S$ is a clique forest. If $|S| > 3k$, then (G, k) is a "yes"-instance for Signed Max-Cut AEE.

Following Crowston et al. [23, Corollary 3], we assume—without loss of generality—from now on that the resulting clique forest $G - S$ does not contain positive edges.

▶ **Lemma 6.6.** *Let G be a signed graph for which $\langle G \rangle$ is a complete graph. Then in time $O(|E(G)|)$, we either find a positive triangle in G or decide that none exists.*

Proof. Let $H = (V(G), E^+(G))$, where $E^+(G)$ are the positive edges in G. As a positive triangle has either exactly zero or exactly two negative edges, our task is to find either a triangle in H or an edge $\{a, b\} \in E(H)$ and a vertex $c \in V(H)$ such that $\{a, c\}, \{b, c\} \notin E(H)$ (remember that $\langle G \rangle$ is a complete graph). In order to achieve this, we try to find a 2-coloring, i.e., a bipartition, of H using breadth-first search [76].

- If this succeeds, then we have found a bipartition $A \uplus B$ of $V(H)$ such that $H[A], H[B]$ are edgeless. If H is a complete bipartite graph or $E(H) = \emptyset$, then G does not contain a positive triangle. Otherwise, we can assume, without loss of generality, that there is a vertex $a \in A$ with $\emptyset \neq N_H(a) \neq B$, i.e., there are vertices $b \in N_H(a)$ and $c \in B \setminus N_H(a)$. Then (a, b, c) is a positive triangle.
- If it fails, then we have found an odd cycle $C = (x_1, \ldots, x_\ell, x_1)$, i.e., ℓ is odd. If $\{x_1, x_3\} \in E(G)$, then (x_1, x_2, x_3) is a positive triangle in G. Otherwise, if $\{x_1, x_3\} \notin E(G)$ and $\{x_1, x_4\} \notin E(G)$, then (x_1, x_3, x_4) is a positive triangle in G. Otherwise, $(x_1, x_4, \ldots, x_\ell, x_1)$ is an odd cycle in H with length smaller than C. Repeat this procedure until a triangle is found. Note that every iteration can be performed in constant time.

Hence, in linear time we either find a positive triangle or decide that none exists.

◀

▶ **Definition 6.7.** Let T be a DFS tree of a graph G rooted at a vertex $r \in V(G)$. For two vertices $v, w \in G$, we say that v is *lower* than w if its distance to r with respect to T is larger than the distance from w to r with respect to T. For a vertex v, we denote by T_v the subtree of T rooted at v. A *child tree* of a vertex v is the subtree T_w of a child w of v.

▶ **Lemma 6.8.** *Let G be a 2-connected graph and let $r \in V(G)$ such that $G - r$ is not 2-connected. Then in time $O(|E(G)|)$, we can find an induced 2-path P in $G - r$ such that $G - V(P)$ is connected.*

Proof. Note that $r \in V(G - V(P))$.

We first state the algorithm before we discuss why it is well-defined and correct.

1. Compute a cut vertex v of $G - r$ and let Z_1, Z_2 be 2-connected components of $G - r$ containing v.
2. For $i \in \{1, 2\}$, find a vertex u_i of $V(Z_i) \setminus \{v\}$ with minimum distance to r with respect to $G - v$, and let P_i be a shortest $[r, u_i]$-path in $G - v$.
3. For $i \in \{1, 2\}$, let T_i be a DFS tree of Z_i rooted at v such that u_i is a child of v in T_i if $\{v, u_i\} \in E(G)$.
4. For $i \in \{1, 2\}$, let w_i be a lowest (w.r.t. T_i) neighbor of v in Z_i.
5. Return the induced 2-path $P = (w_1, v, w_2)$.

Because $G - r$ is not 2-connected, a cut vertex v and thus also Z_1 and Z_2 exist. The paths P_1 and P_2 exist because G is 2-connected, i.e., $G - v$ is still connected. As w_1 and w_2 are in different 2-connected components of $G - r$, they are in different connected components of $G - \{r, v\}$ and therefore not adjacent. Hence, P is indeed an induced path and the algorithm is well-defined.

We now prove that $G - V(P)$ is connected by showing that for every $x \in V(G) - V(P)$ there is an $[x, r]$-path in $G - V(P)$. Note that r is still contained in $G - V(P)$ because it is by definition of Z_1 and Z_2 not contained in either of them.

- First look at the case that $x \in V(Z_1)$ (the case $x \in V(Z_2)$ is analogous). This implies $|V(Z_1)| \geq 3$ because two vertices of Z_1 are contained in P. Because Z_1 is 2-connected, the vertex v is adjacent to at least two vertices of $V(Z_1) \setminus \{v\}$. It follows that u_1 cannot be the lowest neighbor of v in Z_1 by construction of T_1 and thus u_1 is contained in $G - V(P)$. Because v is a cut vertex of $G - r$, every path from r to Z_1 that uses a vertex from Z_2 must also use v. But $P_1 \subseteq G - v$, i.e., P_1 cannot use vertices from Z_2 and thus it does not contain w_2. Hence, $P_1 \subseteq G - V(P)$ and therefore r is connected to a vertex from $Z_1 - \{v, w_1\}$ (namely, u_1).
  Because w_1 is the lowest (w.r.t. T_1) neighbor of v in Z_1, every child tree of w_1 is not adjacent to v. But because Z_1 is 2-connected, every child tree of w_1 is adjacent to a vertex that is higher than w_1 as otherwise w_1 would be a cut vertex of Z_1. This shows that $Z_1 - \{v, w_1\}$ is connected and thus there is an $[x, r]$-path in $G - V(P)$ via u_i.
- Now consider the remaining case that x is neither contained in Z_1 nor in Z_2. As G is 2-connected, there are two $[x, r]$-paths Q_1, Q_2 in G that do not share an internal vertex. Let y be the vertex of $V(Z_1) \cup V(Z_2)$ that is nearest to x with respect to $G - r$. Because $G - r$ is not 2-connected and Z_1 as well as Z_2 are 2-connected components of $G - r$, the vertex y is a cut vertex of $G - r$ separating x from $V(Z_1) \cup V(Z_2) \setminus \{y\}$. This means that every $[x, r]$-path that uses vertices from $V(Z_1) \cup V(Z_2)$ must also contain the

vertex y. Hence, as Q_1 and Q_2 do not share internal vertices, only at most one of these two paths can use vertices from $V(Z_1) \cup V(Z_2)$ and thus one of the paths Q_1, Q_2 is fully contained in $G - V(P)$.

Finally we show that the algorithm runs in linear time. The vertex v and the 2-connected components Z_1, Z_2 can be found in time $O(|E(G)|)$ using any linear-time algorithm for finding 2-connected components in undirected graphs. The vertices u_1, u_2 and the paths P_1, P_2 can be found via breadth-first search in $G - v$, starting in r. The DFS trees T_1, T_2 can also be computed in linear time. The restriction that u_i shall be the direct child of v if these two vertices are adjacent, can easily be followed by selecting the edge $\{v, u_i\}$ as the first traversed edge in the depth-first search. In linear time, we can find the neighbor w_i of v that is the lowest with respect to T_i, $i \in \{1, 2\}$. This completes the proof.

◀

▶ **Lemma 6.9.** *Let G be a connected signed graph, let X be a leaf block of G, and let $r \in V(G)$ such that $V(X) \setminus \{r\}$ does not contain any cut vertex of G. Then we can always apply one of Rules 1-7 to G such that only vertices from X are marked and deleted, in time $O(|E(X)|)$.*

Proof. Let us first argue why we may assume for this proof that the edges of X are given in form of an adjacency matrix (this is a standard argument):

Let $v_1, \ldots, v_{n'}$ be the vertices and $e_1, \ldots, e_{m'}$ be the edges of X. Create an array L of size m' and a 2-dimensional array M of size $n' \times n'$, both of which are not initialized and therefore need only constant construction time. For every edge $e_i = \{v_a, v_b\}$, where $a < b$, set $L[i]$ to (a, b) and set $M[a][b]$ to i. This takes $O(m')$ time in total. After it, L is completely and M is partly initialized.

In order to check now in constant time whether an edge $\{a, b\}$, $a < b$, exists in X, try to read the integer i that is stored in $M[a][b]$. Then $M[a][b]$ is initialized and thus the edge $\{a, b\}$ exists if and only if $L[i]$ is set to (a, b).

Having this construction, we can check in time $O(|E(X)|)$ whether a subset $X' \subseteq V(X)$ induces a clique in G in the following way: We test for the $O(|X'|^2)$ many pairs of vertices of X' in lexicographically ascending order whether the corresponding edge exists in X. We stop at the first vertex pair that does not exist as edge. This way we check at most $|E(X)| + 1 = O(|E(X)|)$ pairs and every check can be processed in constant time.

Let us now turn to the proof of the lemma. We consider the following cases:

1. If X is a clique, then we can find with Lemma 6.6 in time $O(|E(X)|)$ a positive triangle (a, b, c) in X or decide that none exists. If none exists, then Rule 5 applies to $X - r$ and r. If $G - \{a, b, c\}$ is connected, then Rule 1 applies to (a, b, c). If $G - \{a, b, c\}$ is not connected, then Rule 2 applies if $r \notin \{a, b, c\}$, and Rule 4 applies if $r \in \{a, b, c\}$.
2. If X is not a clique, but $X - r$ is a clique, then again try to find a positive triangle (a, b, c) in $X - r$. If this fails, then Rule 3 applies. Otherwise, $r \notin \{a, b, c\}$ and thus Rule 1 or Rule 2 applies.
3. If X is not a clique, $N_X(r) = \{x, y\}$ for some vertices $x, y \in V(X)$ with $\{x, y\} \notin E(X)$, and $X - \{r, x\}$ as well as $X - \{r, y\}$ are cliques, then again try to find a positive triangle (a, b, c) in $X - \{r, x\}$ or $X - \{r, y\}$. If this fails, then Rule 7 applies. Otherwise, $r \notin \{a, b, c\}$ and thus Rule 1 or Rule 2 applies.
4. Now assume that none of the previous cases applies. If $X - r$ is not 2-connected, then we can find with Lemma 6.8 a path P in X that does not use r such that $X - V(P)$ and thus also $G - V(P)$ is connected. Hence, Rule 6 applies to P. From now on we assume that $X - r$ is 2-connected.

We perform a breadth-first search on X starting in r to compute the distance from r to all vertices $x \in V(X) \setminus \{r\}$. For $i \geq 1$, let $L_i \subseteq V(X) \setminus \{r\}$ be the set of vertices with distance i to r.
Find vertices $x, y \in V(X) \setminus \{r\}$ such that $\{x, y\} \notin E(X)$, $L_1 \neq \{x, y\}$, and the distance from r to x is minimum. We do this again by testing all possible vertex pairs in lexicographically ascending order. After at most $|E(X)|$ tested pairs, two non-adjacent vertices must have been found. Note that these vertices must exist, as otherwise one of the previous cases would be applicable.
Find a shortest path Q from r to x; by breadth-first search, this can be done in time $O(|E(X)|)$. The length of Q is at most 2, because if $L_3 \neq \emptyset$, then every pair of vertices from L_1 and L_3 is non-adjacent.
Then we try to find via breadth-first search in time $O(|E(X)|)$ a shortest path P from x to y in $X - (V(Q) \setminus \{x\})$. If P exists, then P is an induced path. Let P' be the unique connected subgraph of P containing x with $|V(P')| = 3$ (i.e., P' contains the "first" three vertices of P).
If P' exists and $G - V(P')$ is connected, then Rule 6 applies. Otherwise, we have found a (not necessarily induced) path $(p_1, \ldots, p_\ell)$ of vertices from $X[V(Q) \cup V(P')]$ with $p_1 = r$ such that $G[X - \{p_1, \ldots, p_\ell\}]$ is not connected. By construction, it holds that $\ell \leq 6$. As X is 2-connected, there is an $i \in \{0, \ldots, \ell - 1\}$ such that $X' := X \setminus \{p_1, \ldots, p_i\}$ is 2-connected, but $X' \setminus \{p_{i+1}\}$ is not. Using Lemma 6.8, we can find a vertex-induced path P' in X' that does not use p_{i+1} such that $X' - V(P')$ is connected. In particular, every vertex is reachable from p_{i+1} and thus from $p_1 = r$ in $X - V(P')$. It follows that $G - V(P')$ is connected and Rule 6 applies to P'. ◀

Given an instance (G, k), we can thus compute in time $O(k \cdot |E(G)|)$ a vertex set S that either proves that (G, k) is a "yes"-instance or $G - S$ is a clique forest. We now show that, if a partition for the vertices in S is already given, we can in time $O(|E(G)|)$ compute an optimal extension to G. We use the following problem, which goes back to Crowston et al. [25]:

Max-Cut Extension
Input: A clique forest G_S and weight functions $w_0, w_1 : V(G_S) \to \mathbb{N}_0$.
Task: Find an assignment $\varphi : V(G_S) \to \{0, 1\}$ maximizing $\sum_{\{x,y\} \in E(G_S)} |\varphi(x) - \varphi(y)| + \sum_{i=0}^{1} \sum_{x:\ \varphi(x)=i} w_i(x)$.

▶ **Lemma 6.10.** Max-Cut Extension *can be solved in time* $O(|V(G_S)| + |E(G_S)|)$ *on clique forests* G_S.

Proof. In order to solve Max-Cut Extension on G_S in time $O(|V(G_S)| + |E(G_S)|)$, we use the natural approach suggested by Crowston et al. [25], and argue why it runs in the desired time. We provide a transformation that replaces an instance $I = (G_S, w_0, w_1)$ with an equivalent instance $I' = (G'_S, w'_0, w'_1)$ such that G'_S has fewer blocks than G_S, and that we can recover an optimal solution for I from an optimal solution for I'. By repeatedly applying the transformation we obtain a trivial instance, and thus the optimal solution for I.

We may assume that G_S is connected, as otherwise we can handle each connected component of G_S separately. Let $X \cup \{r\}$ be the vertices of a leaf block in G, with r a cut vertex of G_S (unless G_S consists of a single block, in which case let r be an arbitrary vertex and $X = V(G_S) \setminus \{r\}$). Recall that by definition of a clique forest, $X \cup \{r\}$ is a clique. For each possible assignment to r, we will calculate the optimal extension to the vertices in X. (This optimal extension depends only on the assignment to r, since no other vertices are

adjacent to vertices in X.) We can then remove all vertices in X, and change the values of $w_0(r)$ and $w_1(r)$ to reflect the optimal extension for each assignment.

Suppose we assign r the value 1. Let $\varepsilon(x) = w_1(x) - w_0(x)$ for each $x \in X$. Now arrange the vertices of X in order $x_1, x_2, \ldots, x_{|X|}$, such that if $i < j$ then $\varepsilon(x_i) \geq \varepsilon(x_j)$. Observe that there is an optimal assignment for which x_i is assigned 1 for every $i \leq t$, and x_i is assigned 0 for every $i > t$, for some $t \in \{0, \ldots, |X|\}$. (Consider an assignment for which $\varphi(x_i) = 0$ and $\varphi(x_j) = 1$, for $i < j$, and observe that switching the assignments of x_i and x_j will increase $\sum_{i=0}^{1} \sum_{x\colon \varphi(x)=i} w_i(x)$ by an amount of $\varepsilon(x_i) - \varepsilon(x_j)$ while $\sum_{\{x,y\}\in E(G_S)} |\varphi(x) - \varphi(y)|$ stays the same.) So we only need to try $|X| + 1$ different assignments to the vertices in X in order to find the optimal coloring when $\varphi(r) = 1$. Let V_1 be the value of this optimal assignment over $X \cup \{r\}$. By the same method we can find the optimal assignment when r is assigned 0, whose value we denote by V_0. Now remove the vertices in X from G_S, and change $w_i(r)$ to V_i for $i = 0, 1$.

Let us now analyze the running time of this procedure. If $\varepsilon(v) > |X|$ for a vertex $v \in X$, then $\varphi(v)$ must be 1 in an optimal assignment. Similarly, $\varphi(v) = 0$ if $\varepsilon(v) < -|X|$. Hence, we only have to sort at most $|X|$ vertices according to their value $\{-|X|, \ldots, |X|\}$, which we can do in time $O(|X|)$ using counting sort.

The value of the first tested single assignment φ can be computed in time $O(|E(G_S[X \cup \{r\}])|$. The next assignment φ' we want to test differs in only one vertex v from the last assignment. Hence, the only differences between φ and φ' are in $E(\{v\}, X \cup \{r\} \setminus \{v\})$. Therefore we can compute the value of φ' in time $O(|N(v)|)$. This way, we can check all $|X| + 1$ assignments in time $O(|E(G_S[X])|)$. Since each edge of $E(G_S)$ belongs to exactly one block of G_S, the entire procedure runs in time $O(|E(G_S)|)$. ◀

We now give a proof for Theorem 2.5. Given a connected signed graph G on m edges, by Lemma 6.9 we find the set S from Proposition 6.5 in time $O(km)$ (the case that k is not decreased can only take $O(m)$ total time). Guess one of the at most 2^{3k} partitions on S and solve the corresponding Max-Cut Extension problem with Lemma 6.10.

Proof of Theorem 2.5. Let (G, k) be an instance of Signed Max-Cut AEE. Compute the 2-connected components of G and apply Lemma 6.9 to a leaf block X of G to obtain an instance (G', k'). Repeat this procedure exhaustively or until $k' \leq 0$.

If Rule 5 was applied, the only remaining vertex of X in G' is the cut vertex in X. Thus we do not need to recompute the 2-connected components of G and we can use Lemma 6.9 immediately again. This way, all applications of Rule 5 take time $O(|E(G)|)$ in total. For every other rule, it holds $k' \leq k - 1$. This means that the other rules are applied at most k times and thus the whole procedure runs in time $O(k \cdot |E(G)|)$.

Let S be the set of marked vertices. If $k' \leq 0$, then (G, k) is a "yes"-instance. Otherwise, $|S| \leq 3k$. We guess a 2-coloring $\varphi_S\colon S \to \{0, 1\}$ for the vertices in S; there are $2^{|S|} \leq 2^{3k} = 8^k$ such 2-colorings. For φ_S, we solve Max-Cut Extension on the clique forest $G - S$, where we try to extend φ_S to a maximum cut in G.

Formally, for an assignment $\varphi\colon S \to \{0, 1\}$ let $S_i = \{v \in S \mid \varphi(v) = i\}$ for $i = 0, 1$. For a vertex $v \in V(G) \setminus S$, define the weight functions $w_0(v) := |N_G^+(v) \cap S_0| + |N_G^-(v) \cap S_1|$ and $w_1(v) := |N_G^+(v) \cap S_1| + |N_G^-(v) \cap S_0|$. Then remove the vertices of S from G. By Proposition 6.5, the resulting graph $G_S = G - S$ is a clique forest. Let p be the number of edges within $G[S]$ that are satisfied by the restriction of φ to $G[S]$. Then for any assignment to the vertices of

G_S, the maximum number of satisfied edges in G is exactly equal to

$$p + \sum_{\{x,y\} \in E(G_S)} |\varphi(x) - \varphi(y)| + \sum_{i=0}^{1} \sum_{x\colon \varphi(x)=i} w_i(x),$$

where $\varphi : V(G_S) \to \{0,1\}$ is the desired bipartition. Thus, (G,k) is a "yes"-instance if and only if the instance of Max-Cut Extension has optimal value at least $|E(G)|/2 + (|V(G)| - 1 + k)/4 - p$. We can test this in time $O(m)$ for every assignment φ_S according to Lemma 6.10. ◀

6.3 A Linear Vertex Kernel for Signed Max-Cut AEE

In this subsection we will show how to obtain a kernel with $O(k)$ vertices and thus prove Theorem 2.6. Let G^0 be the original graph, let S be the set of marked vertices during the exhaustive application of Rules 1-7 on G^0, and let G^r be the resulting graph after the exhaustive application of our kernelization Rules 8-9 (to be defined later) on G^0.

Let C be a block in the clique forest $G - S$. Define

$$C_{\text{int}} = \{v \in V(C) \mid N_{G-S}(v) \subseteq V(C)\}$$

as the *interior* of C, and $C_{\text{ext}} = V(C) \setminus C_{\text{int}}$ as the *exterior* of C. The block C is called *special* if $C_{\text{int}} \cap N_G(S)$ is non-empty. Let $\mathcal{B}$ be the set of blocks in $G^r - S$ and let $\mathcal{B}^\star$ be the set of special blocks in $G^r - S$. A Δ-*block* is a non-special block C on exactly three vertices for which $|C_{\text{ext}}| \leq 2$.

If there is a (unique by Proposition 6.5) remaining vertex v left after the exhaustive application of Rules 1-7, then add an induced 2-path (v,w,x) to G^0, i.e., define $G' = (V(G^0) \cup \{w,x\}, E(G^0) \cup \{\{v,w\},\{w,x\}\})$. Then $(G', k+2)$ is an instance of Max-Cut AEE, that by Proposition 6.4 is equivalent to (G,k) because the excess of an induced 2-path equals 2/4. Therefore, we can assume that every vertex gets removed during the exhaustive application of the reduction rules because we can assume that Rule 6 removes the path (v,w,x) in the last iteration. Furthermore, as Rule 5 can then not be applied last, we can assume that at least one of the vertices that are removed in the last iteration is contained in S.

We will now use two-way reduction rules to reduce the size of $G^0 - S$ by shrinking or merging blocks that satisfy certain conditions. These rules are similar to the two-way reduction rules by Crowston et al. [23]. However, our two-way reduction rules have the property that connected components of $G - S$ cannot "fall apart", i.e., two blocks in $G^r - S$ are reachable from each other if and only if the corresponding blocks in $G^0 - S$ are reachable from each other. We can then show that Rules 1-7 can behave "equivalently" on G^r as on G^0 (Lemma 6.13), i.e., that the same set S of vertices can also be marked in G^r. This is the crucial idea which allows us to obtain better kernelization results than previous work, as it allows the following analysis.

To show size bounds for our kernel G^r, we first argue that (G^r, k) is a "yes"-instance if there are many special blocks. Intuitively, if there are many special blocks in $G^r - S$, we can find large pairwise vertex-disjoint stars Y_s for every $s \in S$, whose leaves are internal vertices of blocks of $G^r - S$. The excess of such a star Y_s grows linearly in its size because a star is a bipartite graph. We then (hypothetically) modify Rules 1-7 in such a way that whenever a vertex $s \in S$ is about to be removed, we additionally remove the associated star Y_s. We can distribute the internal vertices of blocks from $G^r - S$ in such a way to the different stars Y_s

that the generated intermediate graphs during the exhaustive application of these rules are all still connected. Therefore we can conclude with Proposition 6.2 that the excess of G^r can only be by $O(|S|)$ smaller than the total excess of all the stars Y_s. Hence, we can show that there are only $O(k)$ special blocks or (G^r, k) is a "yes"-instance (Lemma 6.17).

Next we limit the total number of blocks in $G^r - S$ by $O(k)$. On a high level, Rule 8 deletes two internal vertices of a block and Rule 9 merges two Δ-blocks. There can only be $O(k)$ blocks in $G^r - S$ with an even number of vertices (Lemma 6.22) because every block corresponds to an application of Rules 1-7 where k was decreased (every application of a rule can "generate" only one block of $G^r - S$ and the only case in which k is not decreased is when Rule 5 removes an even number of vertices, which together with their anchor form a block of odd order).

On the other hand, non-special blocks of odd order can be shrunk by Rule 8. If they have only at most two external vertices, they eventually become Δ-blocks. There cannot be more Δ-blocks than non-Δ-blocks (Lemma 6.20) because Rule 9 merges adjacent Δ-blocks. We conclude in Lemma 6.23 that the total number of blocks is in $O(k)$.

The total number of external vertices in blocks of $G^r - S$, i.e., the number of cut vertices, is of course bounded by the total number of blocks in $G^r - S$. Due to Rule 8, every non-special block in $G^r - S$ contains at most as many internal as external vertices. This is why the total number of vertices in non-special blocks is also bounded by $O(k)$. In order to bound the number of vertices in special blocks (Lemma 6.25), we reuse the approach of Lemma 6.17. The difference is that we do not take only single vertices from special blocks in order to build stars $Y_s, s \in S$, but larger sets of internal vertices from each block. The idea will be described in more detail before Lemma 6.24. This will complete the proof.

6.3.1 Kernelization Rules

We now give our two-way reduction rules, which on an input (G, k) produce an instance (G', k) of Signed Max-Cut AEE. Note that the parameter k does not change. We call a rule *2-safe* if (G, k) is a "yes"-instance if and only if (G', k) is. The first rule is again due to Crowston et al. [23], who showed it to be 2-safe; here we contribute its improved running time analysis. Recall our assumption that (without loss of generality) $G - S$ does not contain any positive edges.

▶ Reduction Rule 8. Let C be a block in $G - S$. If there exists $X \subseteq C_{\text{int}}$ such that $|X| > \frac{|V(C)|+|N_G(X)\cap S|}{2} \geq 1$, $N_G^+(x) \cap S = N_G^+(X) \cap S$ and $N_G^-(x) \cap S = N_G^-(X) \cap S$ for all $x \in X$, then delete two arbitrary vertices $x_1, x_2 \in X$.

▶ Reduction Rule 9. Let C_1, C_2 be Δ-blocks in $G - S$ which share a common vertex v. Make a block out of $V(C_1) \cup V(C_2)$, i.e., add negative edges $\{\{u, w\} \mid u \in V(C_1)\backslash\{v\}, w \in V(C_2)\backslash\{v\}\}$ to G.

The combination of these two rules is a powerful tool to eliminate non-special blocks of odd order: Rule 8 ensures that in every non-special block C it holds $|C_{\text{int}}| \leq |C_{\text{ext}}|$ (otherwise, set X to C_{int}, then $|X| > \frac{|V(C)|}{2} = \frac{|C_{\text{int}}|+|C_{\text{ext}}|}{2} \geq 1$, where the last inequality holds because every non-special block contains at least two vertices). This means that Rule 8 reduces non-special blocks C of odd order with $|C_{\text{ext}}| \leq 2$ to blocks of order 1 (i.e., deleting the block if C was a leaf block of odd order) or order 3. In the latter case, C becomes a Δ-block.

Rule 9 combines two adjacent Δ-blocks to a block of order 5. If the common external vertex of the Δ-blocks is not adjacent to S, the resulting block is also non-special and can therefore again be shrunk by Rule 8. We can therefore contract arbitrarily large chains of non-special blocks.

▶ **Lemma 6.11.** *Rules 8-9 are 2-safe. If they are applied to a connected graph G, then the resulting graph G' is also connected.*

Proof. For Rule 8 we have nothing to show because it is Rule 8 from Crowston et al. [23]. Rule 9 does not destroy connectivity, as nothing is deleted. It remains to show that Rule 9 is 2-safe.

Let $C = V(C_1) \cup V(C_2)$. Consider a partition $V_1 \uplus V_2$ of $V(G)$. This partition induces balanced subgraphs H in G and H' in G' (see Proposition 6.3). Let us first assume that neither $V(C_1)$ nor $V(C_2)$ is completely contained in either V_1 or V_2. Then also $|V_1 \cap C| \leq 3$ and $|V_2 \cap C| \leq 3$. Because $G[C]$ is the union of two triangles and $G'[C]$ is a clique of size 5, the partition induces subgraphs $H[C]$ and $H'[C]$ with exactly four and exactly six edges, respectively. Hence, $|E(H')| = |E(H)| + 2$. It also holds that $\mathsf{pt}(G') = \mathsf{pt}(G) + 2$, as G' is equal to G with four additional edges. It remains to show that there always is a partition which induces maximum balanced subgraphs for G and G' such that neither $V(C_1)$ nor $V(C_2)$ is completely contained in one of the sides of the partition.

Therefore, let us assume w.l.o.g. that $V(C_1)$ be completely contained in V_1. Let b be an internal vertex of C_1. Because b is in G as well as in G' only adjacent to vertices in C, it holds $|N_H(b)| = 0$ and $|N_{H'}(b)| \leq 2$. As it also holds $|N_G(b)| = 2$ and $|N_{G'}(b)| = 4$, the partition $(V_1 \setminus \{b\}) \uplus (V_2 \cup \{b\})$ induces balanced subgraphs of G and G' that cannot be smaller than H and H', respectively. This completes the proof.

◀

▶ **Lemma 6.12.** *Given S, Rules 8-9 can be applied exhaustively to G^0 in total time $O(m)$.*

Proof. First observe that we can compute the blocks of $G^0 - S$ in time $O(m)$ using any linear-time algorithm for detecting 2-connected components. Then we can store for every cut vertex the list of Δ-blocks it belongs to. An update of this list after an application of one of the rules can be done in constant time. As Rule 9 can be applied $O(n)$ times and merging two Δ-blocks takes constant time, all applications of this rule can be done in total time $O(n)$.

We now discuss the running time of Rule 8. Let B be a block in $G^0 - S$. Let S_B be the vertices from S adjacent to B, i.e., $S_B = S \cap N_{G^0}(B)$. Consider the auxiliary graph $H_B := (S_B \cup B_{\text{int}}, E(S_B, B_{\text{int}}))$. We use partition refinement to find the partition $V_1 \uplus \ldots \uplus V_p = B_{\text{int}}$ of the internal vertices of B such that two vertices v, w are in the same set V_i if and only if $N^+_{G^0}(v) = N^+_{G^0}(w)$ and $N^-_{G^0}(v) = N^-_{G^0}(w)$. To be more precise, let $\mathcal{P}$ be a partition of B_{int}. Initially, $\mathcal{P} = \{B_{\text{int}}\}$. Then for every $v \in V$, we refine $\mathcal{P}$ by $N(v)$, i.e., we split every set $X \in \mathcal{P}$ into three sets $X \cap N^+(v)$, $X \cap N^-(v)$, and $X \setminus N(v)$. Using appropriate data structures [75], this refinement can be executed in time $O(|N(v)|)$ in every iteration. Thus, we can compute $V_1 \uplus \ldots \uplus V_p$ in time $O(|V(H_B)| + |E(H_B)|)$. As every edge of G^0 is in at most one auxiliary graph and every vertex $s \in S$ is in at most $|N_{G^0}(s)|$ auxiliary graphs, we can do these computations for all blocks of $G^0 - S$ in total time $O(m)$.

For a block B, we can find the biggest class V_{i^*} in linear time. Then, as long as B does not get merged due to Rule 9, V_{i^*} is the only class from which Rule 8 can delete vertices. (This is a bit subtle, as $|V_{i^*}|$ can be $\frac{|V(B)|-1}{2}$ after deleting vertices from V_{i^*}, but then $N_{G^0}(V_{i^*}) \cap S = \emptyset$; hence, every other V_i has a neighbor in S and would thus need size strictly larger than $\frac{|V(B)|+1}{2}$ in order to meet the requirements of Rule 8).

It is trivial to compute the number of possible applications of Rule 8 to V_{i^*}. This means that we can apply Rule 8 exhaustively (without allowing Rule 9 to be applied in the meantime) on $G^0 - S$ in total time $O(m)$.

Now observe that every block newly created by Rule 9 has constant size. Hence, if we have also computed a partition according to the neighborhoods for the whole graph $G^0 - S$

(in time $O(m)$), we can check in constant time whether we can apply Rule 8 again on a newly created block. As this happens $O(n)$ times, the total time required for all applications of Rule 8 is $O(m)$. ◀

▶ **Lemma 6.13.** *Rules 1-7 can be applied exhaustively to the graph G^r in such a way that the set S' of marked vertices is equal to S. Moreover, if only the Rules 3/5/6/7 are applied to G^0, the same set of rules is applied to G^r.*

The last part of the lemma will be needed later in Section 7.2. It can easily be seen by the fact that the mentioned rules are the only rules applicable to an instance of Max-Cut AEE (where all edges are negative).

Proof of Lemma 6.13. It suffices to show the following: Let G be a connected graph and let G' be the resulting graph after a single application of Rule 8 or Rule 9 to G. Then Rules 1-7 can be applied exhaustively to G' in such a way that the same vertices are marked as during the exhaustive application of these rules to G.

Let $(G = G_0, G_1, \ldots, G_q)$ be the sequence of graphs generated by the exhaustive application of Rules 1-7 to G and let S be the set of marked vertices. Let $X_i := V(G_i) \setminus V(G_{i+1})$ for $i < q$ be the set of vertices removed in the i-th application of one of the rules (we start counting at 0 here for convenience).

- We first consider the case that G' resulted from an application of Rule 8. Let C, X, x_1, x_2 be defined like in this rule. Furthermore, let i and j be the indices such that $x_1 \in X_i$ and $x_2 \in X_j$. W.l.o.g., $i \leq j$. Because $x_1 \notin S$, the set X_i is removed by one of the Rules 2/5/7. The only possible neighbor of x_1 remaining in G_{i+1} is either contained in S or an external vertex in $G - S$. Because x_2 is by the definition of Rule 8 an internal vertex of $G - S$, it follows that $i = j$.
  Consider now the sequence of graphs $(G' = G'_0, G'_1, \ldots, G'_q)$ defined by $G'_{i'} = G_{i'} - \{x_1, x_2\}$ for every index i'. Note that $G'_{i'+1} = G'_{i'} - X_{i'}$ for every index $i' \neq i$, and $G'_{i+1} = G'_i - (X_i \setminus \{x_1, x_2\})$. We show that the exhaustive application of Rules 1-7 can yield this sequence of graphs.
  Consider first the iteration $i' = i$. Because X_i contains vertices that are not in S, this set must be removed from G_i by one of the Rules 2/5/7. If Rule 2 removes X_i from G_i, then Rule 1 or Rule 2 can remove $X_i \setminus \{x_1, x_2\}$ from G'_i, depending on whether $X_i \setminus \{x_1, x_2\} \subseteq S$. If Rule 5 removes X_i from G_i, then either the same rule can remove $X_i \setminus \{x_1, x_2\}$ from G'_i or $G'_i = G'_{i+1}$ due to $|X_i| = 2$. If Rule 7 removes X_i from G_i, then $|X| \geq 3$ by the definition of Rule 8 and therefore the same rule can remove $X_i \setminus \{x_1, x_2\}$ from G'_i.
  For every other iteration we just have to ensure that connectivity is preserved, i.e., it suffices to show that for every $Y \subseteq V(G'_{i'})$ and two vertices $a, b \in V(G'_{i'}) \setminus Y$, the vertices a, b are in the same connected component of $G'_{i'} - Y$ if and only if they are in the same connected component of $G_{i'} - Y$.
  Let P be a shortest a-b-path in $G_{i'} - Y$ (if one exists). If P does not contain x_1 and x_2, then P also exists in $G'_{i'}$. Otherwise, P contains exactly one of these two vertices because they share the same closed neighborhood. Let w.l.o.g. x_1 be the vertex contained in P. Because x_1 is an internal vertex in $G - S$, the predecessor and the successor of x_1 in P must be contained in $C_{\text{ext}} \cup S$. This means that $|V(C)| + |N_G(X) \cap S| \geq |\{x_1, x_2\}| + 2 \geq 4$ and thus $|X| \geq 3$, i.e., there is a vertex $x_3 \in V(G)$ with the same closed neighborhood as x_1. With the same arguments as for x_2 one can conclude that also $x_3 \in X_i$. Therefore, x_3 is contained in $G'_{i'}$ and thus we can replace x_1 by x_3 in P. Hence, a and b are in

the same connected component of $G'_{i'} - Y$ if they are in the same connected component of $G_{i'}$. The converse direction holds trivially.

- Now consider the case that G' resulted from an application of Rule 9. Let v, C_1, C_2 be defined like in this rule. Because C_1 and C_2 are both Δ-blocks and hence not special, the following is well-defined: Let i be the index such that $X_i \subseteq V(C_1) \cup V(C_2)$ and X_i is removed by Rule 5 using v as anchor, and let j be the index such that $v \in X_j$ and X_j is removed by Rule 5 using some anchor w. Let ℓ be the index such that $w \in X_\ell$.
  We define the sequence of graphs $(G' = G'_0, G'_1, \ldots, G'_q)$ in the following way: In iteration i, do nothing, i.e., $G'_{i+1} = G'_i$. In iteration j, remove $X_i \cup X_j$ from G'_j. In all other iterations i', remove $X_{i'}$ from $G'_{i'}$. We show again that this sequence can be generated by the exhaustive application of Rules 1-7 to G'.
  Consider first the j-th iteration. Obviously Rule 5 can remove $X_i \cup X_j$ from G'_j using w as anchor. For every other iteration, observe that w is the only vertex in $V(G) \setminus (X_i \cup X_j)$ with $N_G(w) \neq N_{G'}(w)$. Hence, one can easily check for every single rule and every iteration $i' \neq \ell$ that whenever a rule removes $X_{i'}$ from $G_{i'}$, it can also remove the same set from $G'_{i'}$ (intuitively because the rule cannot "see" the difference between G'_i and G_i). It remains to look at iteration ℓ. Because $G'_{i'} = G_{i'}$ for every index $i' > j$, it follows $G'_\ell = G_\ell$ and thus also the ℓ-th iteration is safe. This completes the proof. ◀

6.3.2 Bounding the Kernel Size

After having shown Lemma 6.13, we can now turn to the task of showing a linear kernel size. We first show some auxiliary lemmas, which will be useful in the proofs of the main Lemmas 6.17/6.25.

For the whole subsection, let $(G^r = G_0, \ldots, G_q)$ be the sequence of graphs generated by the exhaustive application of Rules 1-7 to G^r such that the set of marked vertices is S, and let $X_i := V(G_i) \setminus V(G_{i+1})$ be the set of vertices removed in the $(i+1)$-th application. Recall that we assumed, without loss of generality, that G_q is the empty graph, i.e., $\bigcup_{i<q} X_i$ covers $V(G^r)$.

▶ **Definition 6.14.** Let Int be the set of internal vertices in $G^r - S$. Furthermore we call a vertex v *fixed* if it is removed as the only vertex in an application of Rule 5 with anchor in S, but v is not an isolated vertex in $G^r - S$. Denote by F the set of all fixed vertices. Let $\text{Cand} := \text{Int} \setminus F$ be the set of *candidate vertices*.

Fixed vertices play a special role in our clique forest. If a block B contains a fixed vertex v, then $V(B) \setminus \{v\}$ is removed by Rule 5 using v as anchor, before in a later iteration $\{v\}$ is removed by Rule 5 as the last vertex of its connected component of $G^r - S$. As a consequence, the number of blocks in $G^r - S$ does not increase when Rule 5 removes a fixed vertex. In other words, the total number of applications of Rules 2/5/7 is equal to the number of blocks in $G^r - S$ minus $|F|$. It is also clear that there can only be at most k fixed vertices or (G^r, k) is a "yes"-instance. The name "fixed" stems from the fact that later in the proofs of Lemma 6.17 and Lemma 6.25 we do not want to "reattach" these fixed vertices.

▶ **Lemma 6.15.** *Let G be a connected signed graph, and let X be a vertex set from G such that $G[X]$ is a clique containing only negative edges. If Rule 5 can remove X using some anchor $w \in V(G)$, then* $\mathsf{ex}(G) \geq \mathsf{ex}(G - X) + \min\{|X \cap N_G^+(w)|, |X \cap N_G^-(w)|\}$.

Proof. Let $\alpha := \min\{|X \cap N_G^+(w)|, |X \cap N_G^-(w)|\}$. Order the vertices $\{x_1, \ldots, x_\ell\} = X$ in such a way that there is an index i^* such that the edge wx_i is positive if and only if $i \leq i^*$. Because $G[X]$ is a clique containing only negative edges, the partition

$(\{x_1, \ldots, x_{\lfloor \ell/2 \rfloor}\}, \{x_{\lfloor \ell/2 \rfloor + 1}, \ldots, x_\ell\})$ induces a balanced subgraph of size $\mathsf{pt}(G[X]) + \mathsf{ex}(G[X])$. Now we add the vertex w to the left-hand side.

If $i^* \leq \lfloor \frac{\ell}{2} \rfloor$, then $\alpha = i^*$ and $(\{w, x_1, \ldots, x_{\lfloor \ell/2 \rfloor}\}, \{x_{\lfloor \ell/2 \rfloor + 1}, \ldots, x_\ell\})$ induces a balanced subgraph of size

$$\mathsf{pt}(G[X]) + \mathsf{ex}(G[X]) + i^* + (\ell - \lfloor \ell/2 \rfloor) = \mathsf{pt}(G[X]) + \mathsf{ex}(G[X]) + \lceil \ell/2 \rceil + \alpha.$$

If $i^* \geq \lceil \frac{\ell}{2} \rceil$, then $\alpha = \ell - i^*$ and $(\{w, x_1, \ldots, x_{\lceil \ell/2 \rceil}\}, \{x_{\lceil \ell/2 \rceil + 1}, \ldots, x_\ell\}$ induces a balanced subgraph of size

$$\mathsf{pt}(G[X]) + \mathsf{ex}(G[X]) + \lceil \ell/2 \rceil + (\ell - i^*) = \mathsf{pt}(G[X]) + \mathsf{ex}(G[X]) + \lceil \ell/2 \rceil + \alpha.$$

Because $G[X \cup \{w\}]$ contains ℓ edges and one vertex more than $G[X]$, it follows that $\mathsf{pt}(G[X \cup \{w\}]) = \mathsf{pt}(G[X]) + \ell/2 + 1/4$ and thus

$$\mathsf{ex}(G[X \cup \{w\}]) \geq \mathsf{ex}(G[X]) + \lceil \ell/2 \rceil + \alpha - \ell/2 - 1/4 \geq \alpha,$$

where the last inequality follows from the fact that $\mathsf{ex}(G[X]) = \frac{1}{4}$ if ℓ is even. The lemma now follows from Proposition 6.4. ◀

▶ **Lemma 6.16.** *Let (G, k') be an instance of* Signed Max-Cut AEE *that arises during the exhaustive application of Rules 1-7 to (G^r, k). If in the next step Rule 5 removes a vertex set X from G with an anchor $s \in S$, and if the connected component of $G^r - S$ that contains X consists of a single block, then $\mathsf{ex}(G) \geq \mathsf{ex}(G - X) + \frac{1}{4}$.*

Proof. Let C be the block of $G^r - S$ containing X. For a better understanding, we first point out the relation between C and X: Either $V(C) = X$ or the single vertex in X was the anchor of $V(C) \setminus X$ in a previous iteration and now this single vertex is removed by Rule 5.

We now turn to the proof. If $|X|$ is odd, then the lemma follows immediately from Proposition 6.5. Otherwise $|X|$ is even, $X = V(C)$ and $N_G(x) \cap S = \{s\}$ for every $x \in X$.

If $N_G^+(s) \cap X = \emptyset$ or $N_G^-(s) \cap X = \emptyset$, then Rule 8 would have eliminated X, as X is an isolated block in $G^r - S$, i.e., it contains only internal vertices, which have all the same neighborhood Hence, $\min\{|X \cap N_G^+(w)|, |X \cap N_G^-(w)\} \geq 1$. Because $G^r - S$ contains only negative edges, in particular $G[X]$ contains only negative edges. Therefore we can use Lemma 6.15. ◀

The following lemma is dedicated to bounding the number of special blocks to $O(k)$. Our approach is the following. Let G be a connected vertex-induced subgraph of G^r and let $s \in V(G) \cap S$. Furthermore, let Y_s be a subgraph of G that contains s and internal vertices of blocks from $G^r - S$ that are all adjacent to s. If all these vertices are from different blocks, then Y_s is a star, that is, it is a bipartite graph. Hence, if $G - Y_s$ is connected, it holds $\mathsf{ex}(G) \geq \mathsf{ex}(G - V(Y_s)) + \mathsf{ex}(Y_s) - \frac{1}{4} = \mathsf{ex}(G - V(Y_s)) + (|E(Y_s)| - 1)/4$.

The idea is now to apply Rules 1-7 again to G^r in a modified way: Whenever a vertex from S is removed, then at the same time also the star Y_s is removed, resulting in a decrease of k by roughly $|E(Y_s)|$. If there are more than $\Theta(k)$ many special blocks, i.e., blocks with internal vertices that are adjacent to S, then we should be able to identify G^r as a "yes"-instance.

In order to make this approach work, we make sure that the following properties hold:

- All resulting graphs should be connected. In particular, we do not want to add fixed vertices to Y_s. For an illustration of the arising problem, take a look at the graph G depicted in Fig. 3, in which Rule 6 can remove the sets $X_1 = \{s_1, s_2, s_3\}$ and $X_4 = \{s_4, s_5, s_6\}$

from the graph, whereas Rule 5 removes $X_2 = \{x_2\}$ using x_1 as anchor, and $X_3 = \{x_1\}$ using s_4 as anchor. By definition the vertex x_1 is a fixed vertex. If we added x_1 to Y_{s_3}, then $G'_1 = G[\{x_2, s_4, s_5, s_6\}]$ would not be connected.

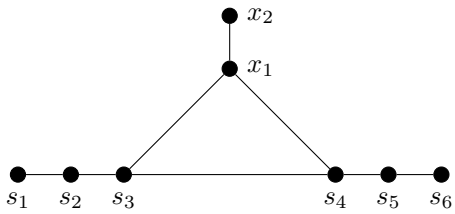


Figure 3 An example where putting the fixed vertex x_1 into Y_{s_3} disconnects the graph.

- Whenever a set X gets removed by Rule 5 during the "original application" and we want to remove the set $X' \subseteq X$ in our modified setting, then X' should also be removable by Rule 5. This means that, if $s \in S$ is the anchor of X, then Y_s cannot contain a vertex $w \in X$, as otherwise both s and w would be in the neighborhood of X', contradicting the conditions of Rule 5.

▶ **Lemma 6.17.** *If $G^r - S$ has more than $11k$ special blocks, then (G^r, k) is a "yes"-instance of* Signed Max-Cut AEE*.*

Proof. For a vertex $s \in S$, let W_s be the union of all vertex sets X_i such that s is the anchor when Rule 5 removes X_i from G_i. Furthermore, let $\Gamma := \emptyset$ be the set of *reattached* vertices.

For every $i = q-1, \ldots, 0$ in decreasing order, do the following procedure successively for every $s \in X_i \cap S$: Let $Y_s \subseteq G^r[(\mathrm{Cand} \setminus W_s) \cup \{s\}]$ be a maximum vertex-induced star centred in s. Add $V(Y_s) \setminus \{s\}$ to Γ.

We define a sequence of graphs $(G^r = G'_0, \ldots, G'_q)$ in the following way: For every $i \in \{0, \ldots, q-1\}$, let $X'_i := (X_i \setminus \Gamma) \cup \bigcup_{s \in X_i \cap S} V(Y_s)$, and let $G'_{i+1} := G'_i - X'_i$.

▶ Claim 6.18. For every $i = 0, \ldots, q-1$, the following properties hold.

1. The graph G'_i is a vertex-induced subgraph of G_i with $G_i - \mathrm{Cand} = G'_i - \mathrm{Cand}$.
2. Both $G^r[X'_i]$ and G'_i are connected.

Proof of the claim. Let $s \in S$ and $w \in V(Y_s) \setminus \{s\}$. As w is not contained in S, it originally got removed by Rule 2/5/7 and this happened not before s got removed (by the definitions of these rules). Hence, every vertex of Γ cannot be removed later than originally, which is why G'_i must be fully contained in G_i. Furthermore, as only candidate vertices of $G^r - S$ get reattached, the claim $G_i - \mathrm{Cand} = G'_i - \mathrm{Cand}$ follows trivially.

For the second part, first note that the subgraphs $G^r[X'_i \cap S]$ and Y_s for all $s \in X_i \cap S$ are connected. If Rule 5 removes X_i from G_i, then $X_i \cap S = \emptyset$, i.e., $G^r[X'_i]$ is a vertex-induced subgraph of the clique $G^r[X_i]$ and thus connected. If Rule 2/7 removes X_i from G_i, then $N_{G_i}(X_i \setminus S) \subseteq X_i \cap S$, i.e., each vertex $s' \in S \cap X_j$ for $j > i$ is non-adjacent to $X_i \setminus S$ in G^r. Hence, at least one star $Y_s, s \in X_i \cap S$, contains a vertex v from $X_i \setminus S$. The graph $G^r[(X'_i \cap X_i) \setminus S]$ is a vertex-induced subgraph of the clique $G^r[X_i \setminus S]$ and thus connected. Hence, every vertex of X'_i is adjacent to $X_i \cap S = X'_i \cap S$ or to v. Thus, $G^r[X'_i]$ is connected.

Regarding G'_i, we show that for every $v, v' \in V(G'_i)$ there is a $[v, v']$-path in G'_i. The vertices v and v' are also contained in G_i because G'_i is a subgraph of G_i. Because G_i is connected, there is a path $v = p_1, \ldots, p_\ell = v'$ in G_i. If this path is not fully contained in G'_i,

let $j, j' \in \{1, \dots, \ell\}$ be indices such that $p_j, p_{j'} \in V(G'_i)$ and $p_{j+1}, \dots, p_{j'-1} \notin V(G'_i)$. We show that there are vertices $w, w' \in V(G'_i)$ for which $G^r[\{p_j, w, w', p_{j'}\}]$ is connected. This suffices to show the claim, as G'_i is a vertex-induced subgraph of G^r.

Because $G_i - \text{Cand} = G'_i - \text{Cand}$, the vertices $p_{j+1}, \dots, p_{j'-1}$ must all be non-fixed internal vertices, and because they form a path, they must all belong to the same block C of $G^r - S$. There are now two possibilities for p_j and $p_{j'}$: Each of the two is either an external vertex of C or contained in S. If p_j is adjacent to every internal vertex of C, we can set $w := p_j$.

Otherwise $p_j \in S$. Let d be the index such that $p_j \in X_d$. Because $p_j \in V(G_i)$, it holds $d \geq i$. Furthermore, as $p_{j+1} \notin V(G'_i)$, it holds $p_{j+1} \in X'_t$ for some $t < i \leq d$. This means that Y_{p_j} contains a vertex w from C_{int}, for otherwise p_{j+1} could have been added to Y_{p_j}. In the same way we can find a vertex $w' \in (C_{\text{int}} \cap Y_{p_{j'}}) \cup \{p_{j'}\}$. In any case, $w, w' \in V(C) \cap V(G'_i)$, and $\{w, w'\} \in E(G'_i)$. This shows the claim. ◀

▶ Claim 6.19. If $|\Gamma| \geq 5k$, then (G^r, k) is a "yes"-instance.

Proof of the claim. Let $i \in \{0, \dots, q-1\}$. If $X'_i = \emptyset$, then trivially $\mathsf{ex}(G'_{i+1}) = \mathsf{ex}(G'_i)$. Therefore we assume $X'_i \neq \emptyset$ from now on.

If X_i is removed by Rule 5, then $X_i \cap S = \emptyset$, i.e., $X'_i \subseteq X_i$ and thus $N_{G'_i}(X'_i) \subseteq N_{G_i}(X_i) = \{v\}$ for some anchor v. Because an anchor cannot be a candidate vertex by definition, and because $G'_i - \text{Cand} = G_i - \text{Cand}$, the vertex v is also contained in G'_i. This means that Rule 5 can remove X'_i from G'_i with the same anchor v. Thus, $\mathsf{ex}(G'_i) \geq \mathsf{ex}(G'_{i+1})$.

Consider now the case that a rule different to Rule 5 removes X_i from G_i. Then $X'_i \cap S \neq \emptyset$. For every vertex $s \in X'_i \cap S$, the subgraph Y_s is a star, and thus by Proposition 6.4 it holds $\mathsf{ex}(Y_s) = \frac{|E(Y_s)|}{4}$. Furthermore, if $X'' := X'_i \setminus \bigcup_{s \in X'_i \cap S} V(Y_s)$ is non-empty, then X'' is a clique (removed by Rule 2/5/7) and thus connected.

Hence, repeated applications of Proposition 6.2 in appropriate order (such that all generated intermediate graphs are connected) then yields

$$\mathsf{ex}(G'_i[X'_i]) \geq \sum_{s \in X'_i \cap S} \frac{|E(Y_s)|}{4} - \frac{|X'_i \cap S|}{4} = \sum_{s \in X'_i \cap S} \frac{|E(Y_s)| - 1}{4}.$$

A further application results in

$$\mathsf{ex}(G'_i) \geq \mathsf{ex}(G'_{i+1}) + \mathsf{ex}(G'_i[X'_i]) - \frac{1}{4} = \mathsf{ex}(G'_{i+1}) + \sum_{s \in X'_i \cap S} \frac{|E(Y_s)| - 1}{4} - \frac{1}{4}.$$

It holds $|\Gamma| = \sum_{s \in S} |V(Y_s) \setminus \{s\}| = \sum_{s \in S} |E(Y_s)|$. Because $|S| \leq 3k$ and only at most k times a rule different to Rule 5 is applied (otherwise, (G^r, k) would be a "yes"-instance), it follows that

$$\mathsf{ex}(G'_0) \geq \mathsf{ex}(G'_q) + \sum_{s \in S} \frac{|E(Y_s)| - 1}{4} - \frac{k}{4} \geq \frac{|\Gamma| - 3k - k}{4} \geq \frac{|\Gamma| - 4k}{4},$$

which is larger than $\frac{k}{4}$ if $|\Gamma| \geq 5k$. Therefore, (G^r, k) is a "yes"-instance if $|\Gamma| \geq 5k$. This shows the claim. ◀

Up to now we have already shown that at most $5k$ special blocks contribute a vertex to Γ or (G^r, k) is a "yes"-instance. It remains to find a bound for the number of special blocks C that do not share a vertex with Γ.

Let C be such a special block, and let i be the largest index such that $X_i \cap V(C) \neq \emptyset$. If C contains a fixed vertex, then X_i consists of this fixed vertex and k was decreased by 1

when Rule 5 removed X_i. If the intersection $N_{G^r}(C_{\text{Int}}) \cap S$ would contain a vertex s' that is not the anchor of X_i, then the star $Y_{s'}$ could be enhanced by a vertex from C_{int}. Hence, $N_{G^r}(C_{\text{Int}}) \cap S$ consists of a single vertex s, which is the anchor of X_i. In particular, X_i got removed by Rule 5.

Let Z be the connected component of $G^r - S$ containing the block C. We now consider the following two cases:

1. If Z contains another special block C', then one vertex of C' is contained in Γ. This is because in every connected component of $G^r - S$ only at most one block can have an anchor in S (namely, the one that is removed last) and only these blocks can contain fixed vertices. Hence, the number of such blocks C is bounded by $|\Gamma|$.
2. Now let Z not contain another special block. If $|X_i|$ is odd, then k is decreased by 1 during the application of Rule 5. If C is an isolated block in $G^r - S$, then Lemma 6.16 assures that $\mathsf{ex}(G'_i) \geq \mathsf{ex}(G'_{i+1}) + \frac{1}{4}$.
   It remains the case that C is not an isolated block, but all other blocks of Z are not special. Let $C' \neq C$ be a leaf block of Z. As C' has not been eliminated by Rule 8, it must contain exactly two vertices, namely an internal vertex and an external vertex w. Then $V(C') \setminus \{w\}$ got removed by Rule 5 using w as anchor. Because $|V(C') \setminus \{w\}| = 1$ is odd, k was decreased by 1 in that iteration.
   Combining these two observations yields that the case that Z does not contain another special block can only occur at most k times or (G^r, k) is a "yes"-instance.

Hence, the number of special blocks that do not share a vertex with Γ is bounded from above by $|\Gamma| + k$. This means that in total there are at most $2|\Gamma| + k$ special blocks. As we already pointed out that (G^r, k) is a "yes"-instance if $|\Gamma| \geq 5k$, the lemma follows. ◀

Now that we have bounded the number of special blocks in $G^r - S$, we can turn to the task of bounding the total number of blocks in $G^r - S$. In the following lemmas we show that a constant fraction of all blocks is special.

▶ **Lemma 6.20.** *The number of Δ-blocks in G^r is at most the number of non-Δ-blocks in G^r.*

Proof. Assume the contrary. By Rule 8, any leaf block C of $G^r - S$ cannot be a Δ-block, as otherwise one could set $X = C_{\text{int}}$ with $|X| = 2 > \frac{3}{2} = \frac{|V(C)|+|N_{G^r}(X)\cap S|}{2}$. Hence, there are two Δ-blocks C_1, C_2 which share a common vertex v. Then Rule 9 applies. ◀

▶ **Definition 6.21.** We define a *block forest* F of $G^r - S$ in the following way. For a connected component Z of $G^r - S$, let C_R be an arbitrary block in Z. For every block C in Z, there is a vertex v_C in F. Add an edge $\{v_{C_R}, v_C\}$ for every block C sharing a vertex with C_R. Additionally, add an edge $\{v_{C_1}, v_{C_2}\}$ if C_1 and C_2 share a vertex and every path from a vertex in C_R to a vertex in C_2 contains at least two vertices from C_1.

It is easy to verify that any block forest is actually a forest.

▶ **Lemma 6.22.** *If more than k non-special blocks in $G^r - S$ have an even number of vertices, then (G^r, k) is a "yes"-instance of* Signed Max-Cut AEE.

Proof. Let B be a non-special block of $G^r - S$, and let w be the external vertex of B. Then Rule 5 removed $V(B) \setminus \{w\}$ using w as anchor. If $|V(B)|$ is even, then $|V(B) \setminus \{w\}|$ is odd and hence k was decreased by 1 in that iteration. ◀

▶ **Lemma 6.23.** *If $G^r - S$ has more than $48k$ blocks, then (G^r, k) is a "yes"-instance of* Signed Max-Cut AEE*. Otherwise, $G^r - S$ has at most $48k$ external vertices, and $\sum_{B\in\mathcal{B}} |B_{\text{ext}}| \leq 96k$.*

Proof. Consider a leaf block C of $G^r - S$ that is not special. Then C cannot have at least three vertices due to Rule 8. Every block with exactly one vertex must be special. Hence, every leaf block is either special or it has exactly two vertices, i.e., due to Lemma 6.17 and Lemma 6.22 there are at most $12k$ blocks that are leaf blocks or have an even number of vertices.

Let F be a block forest of $G^r - S$. Every leaf of F corresponds to a leaf block in $G^r - S$ and every block C in $G^r - S$ with $|C_{\text{ext}}| \geq 3$ corresponds to a vertex with degree at least three in F. Because in every forest the number of leaves is at least the number of vertices with degree at least three, there are at most $12k$ such blocks.

Now consider one of the remaining blocks C. Then C is not special, contains at most two external vertices, and $|V(C)|$ is odd. Because it cannot be shrunk by Rule 8, it holds that $|V(C)| = 3$ and thus C is a Δ-block. This means that with the above arguments we bounded the number of blocks that are not Δ-blocks by $24k$. Lemma 6.20 yields that there can only be up to $24k$ Δ-blocks. From this the bound of $48k$ blocks follows.

Let U be the set of external vertices in $G^r - S$. Every external vertex in $G^r - S$ which is in $c(v)$ blocks induces $c-1 \geq 1$ edges in F. Because F is a forest, it holds $|E(F)| < |V(F)| \leq 48k$. Thus, $\sum_{B\in\mathcal{B}} |B_{\text{ext}}| \leq \sum_{v\in U} c(v) \leq |E(F)| + |U| \leq 96k$. ◀

Now we have bounds for the total number of blocks and the number of external vertices in $G^r - S$. The number of internal vertices of non-special blocks is easily upper-bounded by the number of external vertices (otherwise apply Rule 8). So the remaining challenge is to find an upper bound for the number of internal vertices in special blocks.

We use the same approach as for Lemma 6.17, which already bounded the number of special blocks. There we generated a vertex-induced star Y_s for every $s \in S$, which contained internal vertices of $G^r - S$ that were all adjacent to s. Intuitively speaking, every special block adjacent to s contributed a leaf to Y_s, which lead to a constant gain in our bounds for the excess of G^r. Now a constant gain per special block is not enough for our purposes. Instead we need a gain that grows proportionally to the number of internal vertices in a block.

Let B be a special block of $G^r - S$. First note that due to Rule 8 at most $|B_{\text{ext}}| + |B_{\text{int}}|/2$ vertices of B_{int} can be non-adjacent to S, i.e., it suffices to find a bound for $|N_{G^r}(S) \cap B_{\text{int}}|$. Let $s \in S$ be a vertex that is adjacent to B_{int}. Select subsets $U^+ \subseteq N^+_{G^r}(s) \cap V(B_{\text{int}}), U^- \subseteq N^-_{G^r}(s) \cap V(B_{\text{int}})$, and $\bar{U} \subseteq V(B_{\text{int}}) \setminus N_{G^r}(s)$ with the following properties:

- $|U^+ \cup U^-|$ is maximal.
- $\big||U^+| - |U^-|\big| = |\bar{U}| + 1$.

We will show that we can cover a constant fraction of all internal vertices of B_{int} if we repeat this procedure for every vertex in S. Then we follow the lines of the proof of Lemma 6.15: We can subdivide $\bar{U}$ into to sets $\bar{U}^+$ and $\bar{U}^-$ such that$|U^+ \cup \bar{U}^+| = |U^- \cup \bar{U}^-| - 1$. This means that $(U^+ \cup \bar{U}^+, U^- \cup \bar{U}^-)$ induces a maximum balanced subgraph of $G^r[U^+ \cup U^- \cup \bar{U}]$. Then we add the vertex s to the left-hand side of the partition and increase thereby the number of edges in the induced subgraph by $|U^+ \cup U^-|$, whereas the Poljak-Turzík bound only increases by roughly the half of it. Thus the excess bound grows linearly in $|U^+ \cup U^-|$ and thus in $|N_{G^r}(s) \cap B_{\text{int}}|$.

▶ **Lemma 6.24.** *Let G be a connected signed graph with a cut vertex $s \in V(G)$ such that for every connected component C of $G - s$ the following properties hold:*

- *C is a clique containing only negative edges.*
- *$\big||N_G^+(s) \cap V(C)| - |N_G^-(s) \cap V(C)|\big| = |V(C) \setminus N_G(s)| + 1$.*

Then $\mathsf{ex}(G) \geq \frac{|V(G)\setminus\{s\}|}{4}$.

Proof. Let C be a connected component of $G - s$. Furthermore, let $U^+ = V(C) \cap N_G^+(s)$, $U^- = V(C) \cap N_G^-(s)$, and $\bar{U} = V(C) \setminus N_G(s)$.

First note that C is an odd clique, i.e., $\mathsf{ex}(C) = 0$. Because $\big||U^+| - |U^-|\big| = |\bar{U}| + 1$ and $G - s$ only contains negative edges, there is a partition $\bar{U}^+ \uplus \bar{U}^-$ of $\bar{U}$ such that $|U^+ \cup \bar{U}^+| = |U^- \cup \bar{U}^-| - 1$, i.e., $(U^+ \cup \bar{U}^+, U^- \cup \bar{U}^-)$ induces a balanced subgraph of size $\mathsf{pt}(C)$ in C. Then $(\{s\} \cup U^+ \cup \bar{U}^+, U^- \cup \bar{U}^-)$ induces a balanced subgraph of size $\mathsf{pt}(C) + |N_G(s) \cap V(C)|$, whereas $\mathsf{pt}(G[V(C) \cup \{s\}]) = \mathsf{pt}(C) + \frac{|N_G(s)\cap V(C)|}{2} + \frac{1}{4}$. Therefore,

$$\mathsf{ex}(G[V(C) \cup \{s\}]) \geq \frac{|N_G(s) \cap V(C)|}{2} - \frac{1}{4} = \frac{|V(C)|}{4},$$

where the last equality holds because $|V(C) \cap N_G(s)| = |V(C) \setminus N_G(s)| + 1$. By Proposition 6.4, we conclude that $\mathsf{ex}(G) \geq \frac{|V(G)\setminus\{s\}|}{4}$. ◀

▶ **Lemma 6.25.** *If there are more than $111k$ internal vertices in special blocks in $G^r - S$, then (G^r, k) is a "yes"-instance of* Signed Max-Cut AEE.

Proof. For notational simplicity, all neighborhoods in this proof are with respect to G^r.

For a vertex $s \in S$, let W_s be the union of all vertex sets X_i such that s is the anchor when Rule 5 removes X_i from G_i. Let $W := \bigcup_{s\in S} W_s$. Furthermore, let $\Gamma := \emptyset$ be the set of *reattached* vertices, and let $Y_s := \{s\}$ for every $s \in S$. (Note that Y_s is a set as opposed to a subgraph in the proof of Lemma 6.17, but this is merely a technical issue.)

For all blocks B of $G^r - S$ and every $i = q-1, \ldots, 0$ in decreasing order, run the following procedure successively for every $s \in X_i \cap S$: Let $X := B_{\text{int}} \setminus (F \cup \Gamma \cup W_s)$. Let $U^- \subseteq X \cap N^-(s)$, $U^+ \subseteq X \cap N^+(v)$ and $\bar{U} \subseteq X \setminus N(s)$ with $|U^- \cup U^+|$ maximal such that $\big||U^-| - |U^+|\big| = |\bar{U}| + 1$. If such sets exist, i.e., if $X \cap N(s) \neq \emptyset$, then add $U^- \cup U^+ \cup \bar{U}$ to Y_s and to Γ.

We define a sequence of graphs $(G^r = G'_0, \ldots, G'_q)$ in the following way: For every $i \in \{0, \ldots, q-1\}$, let $X'_i := (X_i \setminus \Gamma) \cup \bigcup_{s\in X_i\cap S} Y_s$, and let $G'_{i+1} := G'_i - X'_i$.

▶ Claim 6.26. For every $i = 0, \ldots, q-1$, the following properties hold.

1. The graph G'_i is a vertex-induced subgraph of G_i with $G_i - \text{Cand} = G'_i - \text{Cand}$.
2. Both $G^r[X'_i]$ and G'_i are connected.

Proof of the claim. The proof is identical to the proof of the corresponding claim in Lemma 6.17. For completeness we repeat it here.

Let $s \in S$ and $w \in V(Y_s) \setminus \{s\}$. As w is not contained in S, it originally got removed by Rule 2/5/7 and this happened not before s got removed (by the definitions of these rules). Hence, every vertex of Γ cannot be removed later than originally, which is why G'_i must be fully contained in G_i. Furthermore, as only candidate vertices of $G^r - S$ get reattached, the claim $G_i - \text{Cand} = G'_i - \text{Cand}$ follows trivially.

For the second part, first note that the subgraphs $G^r[X'_i \cap S]$ and $G[Y_s]$ for all $s \in X_i \cap S$ are connected. If Rule 5 removes X_i from G_i, then $X_i \cap S = \emptyset$, i.e., $G^r[X'_i]$ is a vertex-induced subgraph of the clique $G^r[X_i]$ and thus connected. If Rule 2/7 removes X_i from G_i, then $N_{G_i}(X_i \setminus S) \subseteq X_i \cap S$, i.e., each vertex $s' \in S \cap X_j$ for $j > i$ is non-adjacent

to $X_i \setminus S$ in G^r. Hence, at least one set $Y_s, s \in X_i \cap S$, contains a vertex v from $X_i \setminus S$. The graph $G^r[(X'_i \cap X_i) \setminus S]$ is a vertex-induced subgraph of the clique $G^r[X_i \setminus S]$ and thus connected. Hence, every vertex of X'_i is adjacent to $X_i \cap S = X'_i \cap S$ or to v. Thus, $G^r[X'_i]$ is connected.

Regarding G'_i, we show that for every $v, v' \in V(G'_i)$ there is a $[v, v']$-path in G'_i. The vertices v and v' are also contained in G_i because G'_i is a subgraph of G_i. Because G_i is connected, there is a path $v = p_1, \ldots, p_\ell = v'$ in G_i. If this path is not fully contained in G'_i, let $j, j' \in \{1, \ldots, \ell\}$ be indices such that $p_j, p_{j'} \in V(G'_i)$ and $p_{j+1}, \ldots, p_{j'-1} \notin V(G'_i)$. We show that there are vertices $w, w' \in V(G'_i)$ for which $G^r[\{p_j, w, w', p_{j'}\}]$ is connected. This suffices to show the claim, as G'_i is a vertex-induced subgraph of G^r.

Because $G_i - \mathrm{Cand} = G'_i - \mathrm{Cand}$, the vertices $p_{j+1}, \ldots, p_{j'-1}$ must all be non-fixed internal vertices, and because they form a path, they must all belong to the same block C of $G^r - S$. There are now two possibilities for p_j and $p_{j'}$: Each of the two is either an external vertex of C or contained in S. If p_j is adjacent to every internal vertex of C, we can set $w := p_j$.

Otherwise $p_j \in S$. Let d be the index such that $p_j \in X_d$. Because $p_j \in V(G_i)$, it holds $d \geq i$. Furthermore, as $p_{j+1} \notin V(G'_i)$, it holds $p_{j+1} \in X'_t$ for some $t < i \leq d$. This means that Y_{p_j} contains a vertex w from C_{int}, for otherwise p_{j+1} could have been added to Y_{p_j}. In the same way we can find a vertex $w' \in (C_{\mathrm{int}} \cap Y_{p_{j'}}) \cup \{p_{j'}\}$. In any case, $w, w' \in V(C) \cap V(G'_i)$, and $\{w, w'\} \in E(G'_i)$. This shows the claim. ◀

▶ Claim 6.27. The following inequalities hold for every $i \in \{0, \ldots, q-1\}$.

1. If $X_i \cap S \neq \emptyset$, then $\mathsf{ex}(G'_i) \geq \mathsf{ex}(G'_{i+1}) + \sum_{s \in X'_i \cap S} \frac{|Y_s \setminus \{s\}|}{4} - 1$.
2. If Rule 5 removes X_i from G_i with an anchor $w \in V(G_i)$, then

$$\mathsf{ex}(G'_i) \geq \mathsf{ex}(G'_{i+1}) + \min\{|X'_i \cap N^+(s)|, |X'_i \cap N^-(s)|\} \ .$$

In particular, $\mathsf{ex}(G'_i) \geq \mathsf{ex}(G'_{i+1})$ for the case that $w \notin S$.

Proof of the claim. 1. First of all we know from Lemma 6.24 that $\mathsf{ex}(G[Y_s]) \geq \frac{|Y_s \setminus \{s\}|}{4}$ for every vertex $s \in X'_i \cap S$. Furthermore, every subgraph $G[Y_s], s \in X'_i \cap S$, is connected by construction. The set $X'_i \setminus (\Gamma \cup S)$ is also connected because it is a vertex-induced subgraph of $X_i \setminus S$, which is a clique by the definitions of Rules 2/7. Hence, there is an ordering $Z_1, \ldots, Z_d$ of the sets $X'_i \setminus (\Gamma \cup S)$ and $(Y_s)_{s \in S}$ such that every graph $(G[Z_1 \cup \ldots \cup Z_i])_{i \leq d}$ is connected. We can deduce from the definitions of Rules 1-7 that $d \leq 4$. Hence, a $(d-1)$-fold application of Proposition 6.2 results in

$$\mathsf{ex}(G'_i[X'_i]) \geq \sum_{s \in X'_i \cap S} \frac{|Y_s \setminus \{s\}|}{4} - \frac{3}{4} \ .$$

Another application of the same proposition yields the first part of the claim.

2. Like in the proof of Lemma 6.17, we use the fact that G'_i is a vertex-induced subgraph of G_i with $G'_i - \mathrm{Cand} = G_i - \mathrm{Cand}$. This means that the anchor w of X_i is also contained in G'_i and that $X'_i \subseteq X_i$ cannot have any other neighbors than w. Then Rule 5 can remove X'_i from G'_i using w as anchor. The second part of the claim follows now from Lemma 6.15. ◀

For an index i such that X'_i is removed by Rule 5 from G'_i using an anchor $s \in S$, define $\alpha_i := \min\{|X'_i \cap N^+(s)|, |X'_i \cap N^-(s)|\}$ and $\beta_i := \max\{|X'_i \cap N^+(s)|, |X'_i \cap N^-(s)|\}$. Furthermore, let α_s be the sum of all α_i such that s is the anchor of X'_i, and let $\alpha := \sum_{s \in S} \alpha$.

Because $X_i \cap S \neq \emptyset$ for at most k many indices i (otherwise, (G^r, k) is a "yes"-instance), the previous claim now shows that

$$\mathsf{ex}(G^r) = \mathsf{ex}(G'_0) \geq \sum_{s \in S} \frac{|Y_s \setminus \{s\}|}{4} - k + \alpha = \frac{|\Gamma|}{4} + \alpha - k \ .$$

Hence, if $\alpha \geq \frac{5k}{4}$ or if $|\Gamma| \geq 5k$, then $\mathsf{ex}(G^r) \geq \frac{k}{4}$ and (G^r, k) is a "yes"-instance.

It remains to show that the total number of internal vertices in special blocks is bounded by $\Theta(|\Gamma|+\alpha)$. Let B be a special block of $G^r - S$ and let $X = B_{\text{int}} \setminus (\Gamma \cup F) = (B_{\text{int}} \cap \text{Cand}) \setminus \Gamma$ be the remaining candidate vertices in B. Let i be the smallest index such that $V(B) \cap X_i \neq \emptyset$. There are the following possibilities.

1. The vertex set X_i is removed by Rule 2/7 from G_i or X_i is removed by Rule 5 using an anchor that is an external vertex of $G^r - S$. Then $V(B) = B_{\text{ext}} \cup (B_{\text{int}} \cap \Gamma) \cup X$, because if B contains a fixed vertex, then this vertex must be an external vertex of $G^r - S$.
   Suppose there were $x_1, x_2 \in X$ such that there is a vertex $s \in (N^+(x_1) \setminus N^+(x_2)) \cap S$. Then x_1 and x_2 could have been added to Y_s. (Note that x_1 and x_2 cannot be contained in W_s because the anchor of X_i is not in S.) The same argument for negative edges yields $N^+(x) = N^+(X)$ and $N^-(x) = N^-(X)$ for all $x \in X$.
   Because Rule 8 cannot delete vertices from B, it now holds that

$$|X| \leq \frac{|V(B)| + |N(X) \cap S|}{2} = \frac{|B_{\text{ext}}| + |B_{\text{int}} \cap \Gamma| + |X| + |N(X) \cap S|}{2},$$

   i.e., $|X| \leq |B_{\text{ext}}| + |B_{\text{int}} \cap \Gamma| + |N(X) \cap S|$. Now for every $s \in N(X) \cap S$ there is a vertex of B_{int} in Y_s (otherwise we could add an arbitrary vertex of $N(s) \cap X$ to it). Hence, $|N(X) \cap S| \leq |B_{\text{int}} \cap \Gamma|$ and the bound simplifies to $|X| \leq |B_{\text{ext}}| + 2 \cdot |B_{\text{int}} \cap \Gamma|$.
2. The vertex set X_i is removed by Rule 5 using a vertex $w \in B_{\text{int}} \cap F$ as anchor.
   The only difference to the case before is that B_{int} now contains a (single) fixed vertex and thus $|V(B)| = |B_{\text{ext}}| + |B_{\text{int}} \cap \Gamma| + |X| + 1$. The same reasoning results in the bound $|X| \leq |B_{\text{ext}}| + 2 \cdot |B_{\text{int}} \cap \Gamma| + 1$. Note that this case occurs only at most $|F| \leq k$ times.
3. The vertex set X_i is removed by Rule 5 using a vertex $s \in S$ as anchor. Then B does not contain a fixed vertex because we chose i as the *smallest* index such that X_i contains vertices from B, and an anchor from s means that G_{i+1} cannot contain vertices from $V(B)$ any more.
   The difference to the cases before is that edges from s to X can be positive or negative, i.e., not all vertices of X have the exactly same neighborhood. But with the same arguments one shows that $N^+(x) \setminus \{s\} = N^+(X) \setminus \{s\}$ and $N^-(x) \setminus \{s\} = N^-(X) \setminus \{s\}$ for all $x \in X$. Furthermore, because X_i does not contain vertices from S, the set X is identical to X'_i, which has size $|X'_i| = \alpha_i + \beta_i$.
   Again due to Rule 8 we see that

$$\beta_i \leq \frac{|V(B)| + |N(X) \cap S|}{2} \leq \frac{|B_{\text{ext}}| + |B_{\text{int}} \cap \Gamma| + \alpha_i + \beta_i + |N(X) \cap S|}{2},$$

   i.e., $\beta_i \leq |B_{\text{ext}}| + |B_{\text{int}} \cap \Gamma| + \alpha_i + |N(X) \cap S| \leq |B_{\text{ext}}| + 2 \cdot |B_{\text{int}} \cap \Gamma| + \alpha_i$. Then $|X| = \alpha_i + \beta_i \leq |B_{\text{ext}}| + 2 \cdot |B_{\text{int}} \cap \Gamma| + 2\alpha_i$.

Putting these bounds together shows that the number of internal vertices in special blocks

can be bounded by

$$\begin{aligned}\sum_{B\in\mathcal{B}^\star}|B_{\text{int}}| &= \sum_{B\in\mathcal{B}^\star}(|B_{\text{int}}\cap F| + |B_{\text{int}}\cap\Gamma| + |B_{\text{int}}\setminus(F\cup\Gamma)|)\\ &\leq k + |\Gamma| + \sum_{B\in\mathcal{B}^\star}|B_{\text{int}}\setminus(F\cup\Gamma)|\\ &\leq k + |\Gamma| + \sum_{B\in\mathcal{B}^\star}(|B_{\text{ext}}| + 2\cdot|B_{\text{int}}\cap\Gamma|) + k + 2\alpha\\ &= \sum_{B\in\mathcal{B}^\star}|B_{\text{ext}}| + 2\cdot|\Gamma| + 2k + 2\alpha.\end{aligned}$$

Lemma 6.23 showed that $\sum_{B\in\mathcal{B}^\star}|B_{\text{ext}}| \leq 96k$. Furthermore, we discussed already that $\alpha \leq \frac{5k}{4}$ and $|\Gamma| \leq 5k$ or (G^r, k) is a "yes"-instance. Thus, the total number of internal vertices in special blocks is bounded from above by $96k + 10k + 2k + \frac{5k}{2} \leq 111k$. ◀

We are now ready to prove Theorem 2.6.

Proof of Theorem 2.6. Let (G^0, k) be an instance of Signed Max-Cut AEE. Like in Section 6.2, apply Rules 1-7 exhaustively to (G^0, k) in time $O(k\cdot|E(G^0|)$, producing an instance (G', k') and a vertex set S of marked vertices. If $k' \leq 0$, then (G', k') and thus also (G, k) is a "yes"-instance.

Now apply Rules 8-9 exhaustively to (G^0, k) in time $O(|E(G)|)$ (Lemma 6.12) to obtain an equivalent instance (G^r, k). Check whether (G^r, k) is a "yes"-instance due to Lemma 6.23 or Lemma 6.25. If this is not the case, then there are at most $3k$ vertices in S, at most $48k$ external vertices in $G^r - S$ and at most $111k$ internal vertices in special blocks. If there were more internal than external vertices in a non-special block, we could apply Rule 8 to this block. Thus, the number of internal vertices in non-special blocks is bounded by $96k$ according to Lemma 6.23. Hence, the total number of vertices in G^r is bounded by $3k + 48k + 111k + 96k = 258k$. ◀

7 Linear Vertex Kernels for λ-Extendible Properties

In this section we extend our linear kernels for Signed Max-Cut to all strongly λ-extendible properties satisfying (P1), or (P2), or (P3) (see page 23). Henceforth, fix a strongly λ-extendible property Π, and let (G^0, k) be an instance of Above Poljak-Turzík(Π). For notational brevity, we assume the empty graph to be in Π. We refer to Section 6.1 for the definitions used throughout this section.

As in the previous section, we use a set of 1-safe reduction rules to find a set S such that $G^0 - S$ is a clique forest; the difference compared to Signed Max-Cut is the different change of k. These rules were initially devised by Mnich et al. [87]; for sake of completeness, we list them here. Every rule takes an instance (G, k) and produces an instance (G', k') such that (G, k) is a "yes"-instance if (G', k') is. Initially, $S := \emptyset$.

▶ Reduction Rule 10. Let $v \in V(G)$ and C be a connected component of $G - v$ such that $G[V(C) \cup \{v\}]$ is a clique. Delete C from G and set $k' = k$.

▶ Reduction Rule 11. Suppose Rule 10 does not apply. Let $C_1, \ldots, C_\ell$ be the connected components of $G - v$ for some vertex $v \in V(G)$. If at least one of the C_is is a clique, and at most one of them is *not* a clique, then add v to S, delete v and all the C_is which are cliques from G, and set $k' = k - d \cdot \frac{1-\lambda}{2}$, where d is the number of deleted cliques.

▶ Reduction Rule 12. For vertices $a, b, c \in V(G)$ inducing a path (a, b, c) such that $G - \{a, b, c\}$ is connected, add a, b, c to S, delete them from G, and set $k' = k - \frac{1-\lambda}{2}$.

▶ Reduction Rule 13. Suppose Rule 12 does not apply. Let $v, b \in V(G)$ such that $\{v, b\} \notin E(G)$. Let $C_1, \ldots, C_\ell$ be the connected components of $G - \{v, b\}$. If there is at least one C_i such that both $G[V(C_i) \cup \{v\}]$ and $G[V(C_i) \cup \{v\}]$ are cliques, and there is at most one C_i for which this does *not* hold, then add v, b to S, delete them from G, delete all the C_is which satisfy the conditions, and set $k' = k - \frac{1-\lambda}{2}$.

▶ Proposition 7.1 ([87], Lemmas 6-8). Rules 10-13 are 1-safe and can each be applied in polynomial time. To any connected graph with at least one edge, one of these rules applies and the resulting graph is connected. The exhaustive application of the rules to (G^0, k) either decides that $\mathsf{ex}(G^0) \geq k$, or finds a set S of at most $\frac{6k}{1-\lambda}$ vertices such that $G^0 - S$ is a clique forest. This also holds for all strongly λ-extendible properties of oriented and/or labelled graphs.

The detection which of the reduction rules can be applied to a graph G is completely analogous to the Signed Max-Cut reduction rules. Hence, it follows immediately from Lemma 6.9 that the rules can be applied exhaustively in time $O(km)$.

7.1 Linear Kernel for Properties Diverging on Cliques

We first show that Above Poljak-Turzík Bound(Π) admits kernels with $O(k)$ vertices for all strongly λ-extendible properties Π that are diverging on cliques and for which $\mathsf{ex}(K_i) > 0$ for all $i \geq 2$.

For this subsection, let $(G^0 = G_0, \ldots, G_q)$ be the sequence of graphs generated by the exhaustive application of Rules 10-13 to (G^0, k), let S be the set of marked vertices, and let $X_0, \ldots, X_{q-1}$ be vertex sets such that $G_i - X_i = G_{i+1}$ for all $i < q$.

▶ **Definition 7.2.** Let $i < q$ be an index such that Rule 10 removes X_i from G_i. Let $w \in V(G_{i+1})$ be anchor of X_i. We call the vertices from X_i *bad* if $\mathsf{ex}(G[X_i \cup \{w\}]) = 0$. We also call the block of $G^0 - S$ containing X_i *bad*.

▶ **Lemma 7.3.** *Let Π be a strongly λ-extendible property diverging on cliques, and let (G^0, k) be an instance of* Above Poljak-Turzík(Π)*. Let Γ be the set of bad vertices resulting from an exhaustive application of Rules 10-13 to (G^r, k). Then $|V(G^0) \setminus \Gamma| = O(k)$ or (G^r, k) is a "yes"-instance.*

As a consequence, if $\mathsf{ex}(K_i) > 0$ for all $i \geq 2$, then Above Poljak-Turzík(Π) *admits a kernel with $O(k)$ vertices.*

Proof. We know already that $|S| = O(k)$ or (G^0, k) is a "yes"-instance. If $\mathsf{ex}(K_i) > 0$ for all $i \geq 2$, then there are no bad vertices, i.e., $|\Gamma| = 0$. In the following we show that there is a constant $\varepsilon > 0$ such that $\mathsf{ex}(G_i) \geq \mathsf{ex}(G_{i+1}) + \varepsilon \cdot |X_i \setminus (S \cup \Gamma)|$ for every index $i < q$. This implies that $|V(G) \setminus (S \cup \Gamma)| = O(k)$ or (G^0, k) is a "yes"-instance, showing the lemma.

Because Π diverges on cliques, there is by definition an integer $j \in \mathbb{N}$ and a constant $a > 0$ such that $\mathsf{ex}(K_j) = \frac{1-\lambda}{2} + a$. Note that j only depends on Π and not the instance (G^0, k). Hence, we can treat j as constant for a given property Π. Let

$$\tau' := \min \left\{ \frac{\mathsf{ex}(G)}{|V(G)|} \mid \mathsf{ex}(G) > 0 \text{ and } \langle G \rangle = K_i \text{ for some } 2 \leq i < j \right\}.$$

Then τ' is well-defined, since j is constant; moreover, it can be computed in polynomial time by computing the excess of all graphs G such that $\langle G \rangle$ is a clique of size up to j. Furthermore, $\tau' > 0$ holds by definition.

Let $\tau := \min\{\tau', \frac{a}{2j}\} > 0$.

▶ Claim 7.4. Let C be a clique with $|V(C)| \geq (a + 3 \cdot \frac{1-\lambda}{2}) \cdot (\frac{a}{j} - \tau)^{-1} =: M$. Then $\mathsf{ex}(C) \geq \tau \cdot |V(C)| + 3 \cdot \frac{1-\lambda}{2}$.

Proof of the claim. Let $i := |V(C)|$, and let $r := \lfloor \frac{i}{j} \rfloor$. Then Proposition 6.1 assures that $\mathsf{ex}(C) \geq r \cdot a$. This means that

$$\mathsf{ex}(C) \geq ra \geq \frac{i}{j} \cdot a - a \geq 3 \cdot \frac{1-\lambda}{2} + \tau \cdot i,$$

where the last inequality holds if

$$i \cdot \left(\frac{a}{j} - \tau \right) \geq a + 3 \cdot \frac{1-\lambda}{2} \ .$$

This is exactly the bound for $i = |V(C)|$ given in the claim. Note that $\frac{a}{j} - \tau > 0$ by the definition of τ. ◀

Let now $\varepsilon := \min\{\frac{\tau}{2}, \frac{1-\lambda}{2M}\} > 0$.

▶ Claim 7.5. For every $i < q$, it holds $\mathsf{ex}(G_i) \geq \mathsf{ex}(G_{i+1}) + \varepsilon \cdot |X_i \setminus S|$.

Proof of the claim. Let $i < q$. We consider the following different cases.

1. Rule 12 removes X_i from G_i. Then there is nothing to show, as $X_i \setminus S = \emptyset$.
2. Rule 10 removes X_i from G_i. Let C and v be defined like in Rule 10, i.e., $X_i = V(C)$, C is a connected component of $G_i - v$, and $B := G_i[V(C) \cup \{v\}]$ is a clique of size at least 2. If C contains bad vertices, there is again nothing to show. Assume therefore from now on that $V(C) \cap \Gamma = \emptyset$. This means that $\mathsf{ex}(B) > 0$.
   Because $|X_i \setminus S| = |V(B)| - 1 \geq |V(B)|/2$ and because B is a block of G_i, by the block additivity of Π it suffices to show that $\mathsf{ex}(B) \geq \tau \cdot |V(B)|$.
   If $|V(B)| < j$, this is immediately clear. Otherwise, let $r := \lfloor \frac{|V(B)|}{j} \rfloor$. Proposition 6.1 assures that $\mathsf{ex}(B) \geq r \cdot a$, whereas $|V(B)| \leq (r+1) \cdot j$. Because $r \geq 1$, we therefore obtain $\frac{\mathsf{ex}(B)}{|V(B)|} \geq \frac{r \cdot a}{(r+1) \cdot j} \geq \frac{a}{2j} \geq \tau$.

3. Rule 13 removes X_i from G_i. Let $v, b \in X_i \cap S$. Mnich et al. [87, Observation 19] showed that X_i consists of v, b, and a single clique C. If $|V(C)| \geq M$, then $\mathsf{ex}(C) \geq \tau \cdot |V(C)| + 3 \cdot \frac{1-\lambda}{2}$. Proposition 6.1 then shows

$$\begin{aligned}\mathsf{ex}(G_i[X_i]) &\geq \mathsf{ex}(G_i[X_i \setminus \{v\}]) - \frac{1-\lambda}{2} \geq \mathsf{ex}(G_i[X_i \setminus \{v,b\}]) - 2 \cdot \frac{1-\lambda}{2} \\ &\geq \tau \cdot |V(C)| + \frac{1-\lambda}{2}.\end{aligned}$$

Another application yields

$$\mathsf{ex}(G_i) - \mathsf{ex}(G_{i+1}) \geq \mathsf{ex}(G_i[X_i]) - \frac{1-\lambda}{2} \geq \tau \cdot |V(C)| \geq \varepsilon \cdot |V(C)| \ .$$

Otherwise, $|V(C)| \leq M$. Because Rule 13 is 1-safe, it holds

$$\mathsf{ex}(G_i) - \mathsf{ex}(G_{i+1}) \geq \frac{1-\lambda}{2} \geq \frac{1-\lambda}{2M} \cdot |V(C)| \geq \varepsilon \cdot |V(C)| \ .$$

4. Rule 11 removes X_i from G_i. Let $C_1, \ldots, C_d$ be the connected components of $G_i[X_i \setminus S]$. Then every C_i is a clique. Note that d is exactly the variable named d in the definition of Rule 11.
Order the cliques so that $|V(C_j)| \geq M$ if and only if $j \leq p$ for some $p \in \{0, \ldots, d\}$.
If $p = 0$, then, as Rule 11 is 1-safe, it holds

$$\mathsf{ex}(G_i) - \mathsf{ex}(G_{i+1}) \geq d \cdot \frac{1-\lambda}{2} \geq d \cdot \frac{1-\lambda}{2dM} \cdot |X_i \setminus S| \geq \varepsilon \cdot |X_i \setminus S| \ .$$

Consider now the case $p > 0$. The graph $G' := G - \bigcup_{j \leq p} V(C_j)$ is still connected. Then the previous claim together with Proposition 6.2 yields

$$\mathsf{ex}(G_i) \geq \mathsf{ex}(G') + \sum_{j=1}^{p} \tau \cdot |V(C_j)| + 2p \cdot \frac{1-\lambda}{2} \geq \mathsf{ex}(G') + \sum_{j=1}^{p} \tau \cdot |V(C_j)| + \frac{1-\lambda}{2} \ .$$

Now if $C_1, \ldots, C_p$ did not exist, then Rule 11/13 could still remove $C_{p+1}, \ldots, C_d$, decreasing k by $(d-p) \cdot \frac{1-\lambda}{2} \geq \varepsilon \cdot \sum_{j>p} |V(C_j)|$. Now observe that

$$\mathsf{ex}(G_i - (X_i \setminus S)) \geq \mathsf{ex}(G_{i+1}) - \frac{1-\lambda}{2},$$

because $X_i \cap S$ consists of a single vertex. Putting these bounds together yields the desired result $\mathsf{ex}(G_i) \geq \mathsf{ex}(G_{i+1}) + \varepsilon \cdot |X_i \setminus S|$. ◀

The lemma follows now immediately from the previous claim. ◀

▶ **Theorem 7.6.** *Let Π be a strongly λ-extendible property. If $\lambda \neq \frac{1}{2}$ or $G \in \Pi$ for every G with $\langle G \rangle = K_3$, then* Above Poljak-Turzík(Π) *admits a kernel with $O(k)$ vertices.*

Proof. Crowston et al. [26, Lemmas 24-26] show that if $\lambda \neq \frac{1}{2}$ or $K_3 \in \Pi$, then Π diverges on cliques and $\mathsf{ex}(K_i) > 0$ for all $i \geq 2$. Therefore, we can apply Lemma 7.3. ◀

7.2 Strongly $\frac{1}{2}$-Extendible Properties on Oriented Graphs

We now turn to strongly $\frac{1}{2}$-extendible properties Π on oriented graphs. We can now use a subset of Rules 1-7 again, to be more precise, exactly the rules that are applicable to signed graphs with only negative edges. This has the advantage that we will be able to reuse Lemma 6.13.

We restate the rules here because the parameter k is scaled by a factor of $\frac{1}{4}$ due to the different problem definitions. Let G always be a connected graph.

▶ Reduction Rule 14. Let C be a connected component of $G - v$ for some vertex $v \in V(G)$ such that $G[V(C) \cup \{v\}]$ is a clique. Delete C and set $k' = k$.

▶ Reduction Rule 15. Let C be a connected component of $G - v$ for some vertex $v \in V(G)$ such that C is a clique. If there exist $a, b \in V(C)$ such that $G - \{a, b\}$ is connected and $\{a, v\} \in E(G)$, but $\{b, v\} \notin E(G)$, then add a, b to S and delete them from G, and set $k' = k - \frac{1}{2}$.

▶ Reduction Rule 16. Let (a, b, c) be an induced 2-path for some $a, b, c \in V(G)$ such that $G - \{a, b, c\}$ is connected. Add a, b, c to S and delete them from G, and set $k' = k - \frac{1}{4}$.

▶ Reduction Rule 17. Let $v, b \in V(G)$ such that $\{v, b\} \notin E(G)$ and $G - \{v, b\}$ has exactly two connected components C, Y. If $G[V(C) \cup \{v\}]$ and $G[V(C) \cup \{b\}]$ are cliques, then add v, b to S and delete them from G together with C, and set $k' = k - \frac{1}{4}$.

Rules 14-17 are exactly Rules 5/3/6/7 for Signed Max-Cut AEE with all edges negative.

Note that the concept of *bad* vertices translates naturally to vertices removed by Rule 14, which is equivalent to Rule 10.

▶ **Lemma 7.7.** *Rules 14-17 are 1-safe. To any connected graph with at least one edge, one of the rules applies and the resulting graph is connected. If S is the set of marked vertices, then $G - S$ is a clique forest. If $|S| > 12k$, then (G, k) is a "yes"-instance.*

Proof. First note that we cannot deduce 1-safeness from the fact that these rules are 1-safe for Signed Max-Cut AEE. But Rules 14/16/17 are 1-safe because they are equal to or special cases of Rules 10-13, which were shown to be 1-safe for all λ-extendible properties.

Now we show that Rule 15 is 1-safe. Let G, a, b, v and C like in the description of the rule, and let G' be the resulting graph. Let H' be a Π-subgraph of G'. We extend H' to a Π-subgraph H of G by adding the vertices a, b, the edge $\{a, b\}$, and at least $\lceil \frac{|E(\{a,b\}, V(G'))|}{2} \rceil$ edges between $\{a, b\}$ and $V(G')$. We can do this due to the extendibility property of Π. Now observe that $|E(\{a, b\}, V(G'))|$ is odd because every vertex of $V(G') \setminus \{v\}$ is adjacent to a if and only if it is adjacent to b. Therefore, we can guarantee that

$$\begin{aligned} |E(H)| &\geq |E(H')| + 1 + \left\lceil \frac{|E(\{a, b\}, V(G'))|}{2} \right\rceil \\ &= |E(H')| + 1 + \frac{|E(\{a, b\}, V(G'))| + 1}{2}. \end{aligned}$$

As $\mathsf{pt}(G) = \mathsf{pt}(G') + \frac{2}{4} + \frac{1}{2} + \frac{|E(\{a,b\}, V(G'))|}{2}$, it follows that $\mathsf{ex}(G) \geq \mathsf{ex}(G') + \frac{1}{2}$, i.e., Rule 15 is 1-safe.

Now we argue that one of the rules applies to any connected graph G with at least one edge. Let G' be the signed graph that results from adding a negative sign to every edge of $\langle G \rangle$. We know from Proposition 6.5 that one of the Rules 1-7 applies to G'. To be more precise, Rule 3/5/6/7 applies to G' because all other rules require at least one positive edge. But the mentioned rules correspond exactly to Rules 14-17, i.e., one of these rules must apply to G.

Let H be the resulting graph after applying one of the Rules 14-17 to G, and let H' be the resulting graph after applying the corresponding rule of the Rules 3/5/6/7 to G'. Then $\langle G' \rangle = \langle H' \rangle$ and the same set of vertices has been marked. Therefore, we can follow from Proposition 6.5 that H is connected and that $G - S$ is a clique forest.

The last claim simply follows from the fact that Rules 14-17 decrease k by at least $\frac{1}{12}$ for every vertex that is added to S. Note that the change from $|S| \geq 3k$ to $|S| \geq 12k$ stems from the different meanings of k: in the Signed Max-Cut AEE problem the question is whether $\mathsf{ex}(G) \geq \frac{k}{4}$, whereas Above Poljak-Turzík(Π) asks whether $\mathsf{ex}(G) \geq k$. ◀

Like Crowston et al. [26], we restrict ourselves to hereditary properties. Let $\vec{K}_3$ be the orientation of K_3 which is an oriented cycle, and let $\vec{\vec{K}}_3$ be the only (up to isomorphisms) other orientation of K_3. Crowston et al. [26] showed that if $\vec{K}_3 \in \Pi$, then also $\vec{\vec{K}}_3 \in \Pi$, and thus Theorem 7.6 applies. We now consider the case that $\vec{K}_3 \notin \Pi$ together with $\vec{\vec{K}}_3 \in \Pi$.

▶ Proposition 7.8 ([26]). Let Π be a hereditary strongly $\frac{1}{2}$-extendible property on oriented graphs with $\vec{\vec{K}}_3 \in \Pi$. Then $\mathsf{ex}(K_i) > 0$ for all $i \geq 4$ and Π diverges on cliques.

Following Proposition 7.8, Lemma 7.3 shows a "kernel" with $O(k) + |\Gamma|$ vertices, where Γ is the set of bad vertices of an input graph. The only oriented clique with at least two vertices, but without positive excess is $\vec{K}_3$, because $\mathsf{ex}(K_2) = \frac{1}{4}$ for $\frac{1}{2}$-extendible properties. Therefore, a vertex is bad if and only if it belongs to a set X removed from G by Rule 14 using an anchor w such that $G[X \cup \{w\}] \cong \vec{K}_3$. Hence, we only need reduction rules to bound the number of blocks B in a clique forest with $B \cong \vec{K}_3$. (Note: If the anchor w is contained in S, then the resulting block in the clique forest has only two vertices, but this can happen only once per connected component of $G^r - S$, the number of which is dominated by other tractable terms.)

Let Π be a hereditary strongly $\frac{1}{2}$-extendible property on oriented graphs with $\vec{\vec{K}}_3 \in \Pi$. Let (G^0, k) be an instance of Above Poljak-Turzík(Π). Lemma 7.7 either proves that (G^0, k) is a "yes"-instance, or it finds a set S of at most $12k$ vertices such that $G^0 - S$ is a clique forest. Starting with (G^0, k), we apply the following reduction rules, which on input (G, k) produce an equivalent instance (G', k).

▶ Reduction Rule 18. Let $a, b \in V(G) \setminus S$ and $w \in V(G) \setminus \{a, b\}$ such that $N_G(a) = \{w, b\}$, $N_G(b) = \{w, a\}$, and $G[\{w, a, b\}] \cong \vec{K}_3$. Delete a and b from G.

▶ Reduction Rule 19. Let B_1, B_2, B_3 be non-leaf-blocks in $G - S$ and let $v_1, \ldots, v_4 \in V(G)$ be such that

- $v_i, v_{i+1} \in (B_i)_{\text{ext}}$ for all $i \in \{1, 2, 3\}$;
- $B_i \cong \vec{K}_3$ for all $i \in \{1, 2, 3\}$; and
- $N_G(\{v_2, v_3, w_1, w_2, w_3\}) = \{v_1, v_4\}$, where w_i is the internal vertex of B_i.

Delete v_3, w_3, and add edges $\{v_2, v_4\}$ and $\{w_2, v_4\}$ to G.

Intuitively speaking, Rule 19 takes three blocks in $G - S$ that form a "path" and are all isomorphic to $\vec{K}_3$. If all vertices except the "endpoints" v_1 and v_4 are not adjacent to S and not contained in other blocks from $G - S$, then it is safe to delete one block. For an illustration, see Fig. 4.

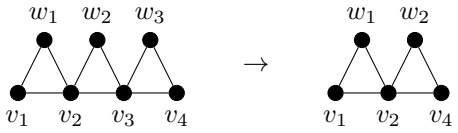


Figure 4 Illustration of Rule 19.

▶ **Lemma 7.9.** *Let Π be a hereditary strongly $\frac{1}{2}$-extendible property on oriented graphs with $\vec{K}_3 \in \Pi$. Then Rules 18-19 are 2-safe. The resulting graphs are connected.*

Proof. Let G' be the resulting graph. Then G' is clearly connected. Rule 18 is 2-safe because $\mathsf{ex}(G)$ is additive on 2-connected components and $G[\{a, b, w\}]$ is a 2-connected component of G.

To show that Rule 19 is 2-safe, let $\widetilde{G} = G - \{v_2, v_3, w_1, w_2, w_3\}$. Let $X := V(B_1) \cup V(B_2) \cup V(B_3)$ and $X' := X \cap V(G')$. First note that $|E(H[X])| \leq 6$ for any Π-subgraph H of G and $|E(H[X'])| \leq 4$ for any Π-subgraph H of G', for otherwise there would be a block B_i with $|E(H[V(B_i)])| = 3$. This is a contradiction, as Π is hereditary and $\vec{K}_3 \notin \Pi$. Also note that $\mathsf{pt}(G) = \mathsf{pt}(G') + 2$.

As Π is hereditary, it holds that $H[V(\widetilde{G})] \in \Pi$ and $H'[V(\widetilde{G})] \in \Pi$ for all Π-subgraphs H and H' of G and G'. Hence, it suffices to show that we can extend every Π-subgraph $\widetilde{H}$ of $\widetilde{G}$ to Π-subgraphs H and H' of G and G' such that $|E(H[X])| = 6$ and $|E(H[X'])| = 4$.

Let $\widetilde{H}$ be a Π-subgraph of $\widetilde{G}$. Let $Y = G[X \setminus \{v_1, v_4\}] - \{v_2v_3\}$ and $Y' = G'[X' \setminus \{v_1, v_4\}]$. Examine Fig. 4 again for an illustration. Then Y and Y' both are trees with 4 and 2 edges, respectively, and hence $Y, Y' \in \Pi$; see Fig. 5. By the extendibility of strongly $\frac{1}{2}$-extendible

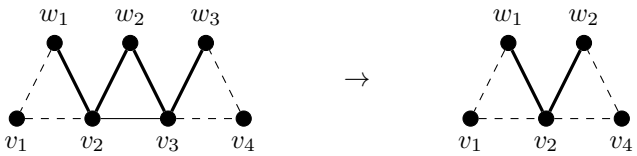


Figure 5 The Π-subgraph Y in G and the Π-subgraph Y' in G' are highlighted by thick edges. At least half of the dotted edges between $\tilde{H}$ and Y or Y', respectively, can be added to a Π-subgraph.

properties, there is a Π-subgraph H of G which contains all edges of $\widetilde{H}$, all edges of Y and at least half of the edges between them. There are exactly 4 edges between them $\widetilde{H}$ and Y, at most two of which can be contained in H as otherwise $\vec{K}_3$ would be a subgraph of H. Hence, $|E(H[X])| = 6$. With the same arguments we obtain a Π-subgraph H' of G' with $|E(H'[X'])| = 4$; this completes the proof. ◀

From now on, let G^r be the resulting graph after the exhaustive application of Rules 18-19 on G^0. Furthermore, let $(G^r = G_0, \ldots, G_q)$ be the sequence of graphs generated by the exhaustive application of Rules 14-17 to (G^r, k), let S be the set of marked vertices, and let $X_0, \ldots, X_{q-1}$ be the vertex sets such that $G_i - X_i = G_{i+1}$ for every $i < q$.

Rules 18-19 are special cases of Rules 8-9. As Rules 14-17 are Rules 5/3/6/7 for Signed Max-Cut AEE with all edges negative, the next lemma follows from Lemma 6.13.

▶ **Lemma 7.10.** *Rules 14-17 can be applied exhaustively on the graph G^r in such a way that the set S' of vertices removed by their application is equal to S.*

Let $\mathcal{B}^-$ be the set of bad blocks in $G^r - S$ and let $\mathcal{B}^+$ be the set of all other blocks in $G^r - S$. Let $\tilde{\mathcal{B}}$ be the set of bad special blocks, i.e., the set of blocks $B \in \mathcal{B}$ with $B_{\text{int}} \cap N_{G^r}(S) \neq \emptyset$. Furthermore, let R be the set of vertices $r \in V(G^r) \setminus S$ such that

1. r is contained in exactly two blocks B_1, B_2 of $G^r - S$,
2. both blocks B_1 and B_2 are bad,
3. $(B_1)_{\text{int}} \neq \emptyset \neq (B_2)_{\text{int}}$, and
4. B_1 and B_2 are not special.

▶ **Lemma 7.11.** *It holds* $|\mathcal{B}^-| = O(|\mathcal{B}^+| + |\tilde{\mathcal{B}}| + |R|)$.

Proof. Let $V^- := \bigcup_{B \in \mathcal{B}^-} V(B)$ be the set of vertices contained in bad blocks (i.e., the bad vertices together with anchors that are not in S and used to removed bad vertices), and let $H := G[V^-]$. Because $V^- \subseteq V(G^r) \setminus S$ and because $G^r - S$ is a clique forest, also the graph H is a clique forest.

Consider now a block forest F of H (see Definition 6.21). For a block B of H, denote the corresponding vertex in F by f_B. Let n_1, n_2, and $n_{\geq 3}$ be the number of vertices in F with degree 1, 2, and at least 3, respectively. Every leaf f_B of F corresponds to a leaf block B in H, i.e., n_1 is at most the number of leaf blocks in H. Therefore we first bound the number of leaf blocks in H.

▶ Claim 7.12. The number of leaf blocks in H is bounded by $O(|\mathcal{B}^+| + |\tilde{\mathcal{B}}|)$.

Proof of the claim. Let B be a leaf block of H. Then there are the following possibilities:

- B is an isolated block in $G^r - S$. Let i be the index such that X_i contains two vertices of B. Then X_i was removed by Rule 14 using an anchor $w \in V(G)$ with $G^r[V(B) \cup \{w\}] \cong \vec{K}_3$. Because B did not get eliminated by Rule 18, it must hold that $N_{G^r}(V(B)) \setminus \{w\} \neq \emptyset$. This means that B is special. There can only be at most $|\tilde{\mathcal{B}}|$ many of these blocks.
- $|V(B)| = 2$ and B is isolated in H, but not in $G^r - S$. This means that B shares a vertex v with a block from $\mathcal{B}^+$. This vertex v is not contained in any other block from $\mathcal{B}^-$. Therefore, there can only be at most $|\mathcal{B}^+|$ many of these blocks.
- $|V(B)| = 2$ and B is not isolated in H. Then B shares an external vertex with a block B' in H. Because every connected component of H can only contain at most one block with exactly two vertices (namely, the block removed last), the number of such blocks B is bounded by the number of blocks in H with three vertices.
- $|V(B)| = 3$. Then B contains three vertices u, v, w such that $N_H(u) = \{v, w\}$ and $N_H(v) = \{u, w\}$. Because u and v did not get eliminated by Rule 18, there must be a vertex $z \in V(G)$ such that $z \in N_{G^r}(\{u, v\})$. If $z \in S$, then B is special. Otherwise, B shares a vertex with a block from $\mathcal{B}^+$, and this vertex is not contained in any other block from $\mathcal{B}^-$. Therefore the total number of blocks B with $|V(B)| = 3$ is bounded by $|\tilde{\mathcal{B}}| + |\mathcal{B}^+|$. ◀

Because F is a forest, it holds $n_{\geq 3} \leq n_1$. It now suffices to bound n_2 because the number of blocks in H is equal to $n_1 + n_2 + n_{\geq 3}$.

Let f_{B_1} and f_{B_2} be two adjacent vertices in F. There are the following possibilities.

- The degrees of both f_{B_1} and f_{B_2} are not equal to 2. The number of such pairs is already bounded by $O(n_1)$.
- $|N_H(f_{B_1})| = 2$ and $|N_H(f_{B_2})| = 1$. There are at most n_1 such pairs.
- $|N_H(f_{B_1})| = 2$ and $|N_H(f_{B_2})| \geq 3$. There are at most $n_1 + n_{\geq 3} \leq 2n_1$ such pairs. One can easily see this by contracting all vertex sets X of F such that X is a path and $|N_H(x)| = 2$ for all $x \in X$.
- $|N_H(f_{B_1})| = 2$ and $|N_H(f_{B_2})| = 2$. The number of cases where B_1 or B_2 is a leaf block (this can happen if this block is selected as root of the connected component of F in the definition of block forests) is bounded by the number of leaf blocks in H.
  The number of cases that B_1 or B_2 contains only two vertices is bounded by the number of connected components of H, which is in turn bounded by the number of leaf blocks in H.
  It remains the case that B_1 and B_2 are not leaf blocks and B_1 and B_2 both contain exactly 3 vertices. Then both blocks each contain exactly one internal vertex. The number of cases where B_1 or B_2 is special is bounded by $|\tilde{\mathcal{B}}|$.

If this is not the case, let w be the external vertex shared by B_1 and B_2. Then $w \in R$. Hence, the number of these remaining pairs is bounded by $|R|$.

This concludes the proof. ◀

▶ **Lemma 7.13.** *It holds $|\mathcal{B}^+| + |\tilde{\mathcal{B}}| = O(k)$ or (G^r, k) is a "yes"-instance.*

Proof. Let $B \in \mathcal{B}^+$ and let $i < q$ be the index that generates B. If Rule 15-17 removes X_i from G_i, then $\mathsf{ex}(G_i) \geq \mathsf{ex}(G_{i+1}) + \frac{1}{4}$ because these rules are 1-safe.

On the other hand, if Rule 14 removes X_i from G_i using an anchor $w \in V(G_i)$, then $\mathsf{ex}(G_i[X_i \cup \{w\}]) > 0$ because B is not a bad block. Because $\lambda = \frac{1}{2}$, the term $4 \cdot \mathsf{ex}(G)$ must be integral for every graph G. It follows from the block additivity of Π that $\mathsf{ex}(G_i) - \mathsf{ex}(G_{i+1}) \geq \mathsf{ex}(G_i[X_i \cup \{w\}]) \geq \frac{1}{4}$. Hence, $|\mathcal{B}^+| \leq 4k$.

We now turn to the number of bad special blocks. Because Rules 14-17 is exactly the subset of Rules 1-7 applicable to signed graphs without positive edges, and because the kernelization Rules 8-9 for Signed Max-Cut AEE and Rules 18-19 for Above Poljak-Turzík(Π) do not change the number of special blocks, we can derive from Lemma 6.17 that the number of special blocks in $G^r - S$ is bounded by $O(k)$ or (G^r, k) is a "yes"-instance. ◀

▶ **Lemma 7.14.** *It holds $|R| = O(|R \cap N_{G^r}(S)|)$. Furthermore, it holds $|R \cap N_{G^r}(S)| = O(|\mathcal{B}^+| + |\tilde{\mathcal{B}}| + k)$ or (G^r, k) is a "yes"-instance.*

Proof. For the first part of the proof, let $r_1, r_2 \in R$ be adjacent in G^r. Then there are by the definition of R three blocks $B_1, B_2, B_2 \in \mathcal{B}^-$ such that $r_1 \in V(B_1)$, $r_1, r_2 \in V(B_2)$, and $r_2 \in V(B_3)$, such that B_1, B_2, B_3 are bad and each have an internal vertex that is not adjacent to S. Furthermore, there are no other blocks of $G^r - S$ containing r_1 or r_2.

If both r_1 and r_2 were not adjacent to S, then all conditions of Rule 19 would be met (r_1 and r_2 would correspond to v_2 and v_3 in the definition of this rule). Hence, at least one of r_1 and r_2 is adjacent to S. Because every vertex of R is adjacent to at most two other vertices from R, this means that $|R| = O(|R \cap N_{G^r}(S)|)$.

Now we turn to the second part of the lemma. Let T be a maximum independent set in $G[R \cap N_{G^r}(S)]$. Again, as every vertex of R is adjacent to at most two other vertices from R, it holds $|T| \geq \frac{1}{3} \cdot |R \cap N_{G^r}(S)|$.

Every vertex $t \in T \subseteq R$ is by definition of R adjacent to exactly two internal vertices t', t'' of $G^r - S$, and these two vertices are not adjacent to S. For $t \in T$, let $U_t = \{t, t', t''\}$, and let $U := \bigcup_{t \in T} \{t, t', t''\}$. Then $G^r[U]$ is a forest, because if there was an internal vertex x from $G^r - S$ adjacent to two vertices $t_1, t_2 \in T$, then t_1 and t_2 would be adjacent. See Figure 6 for an illustration.

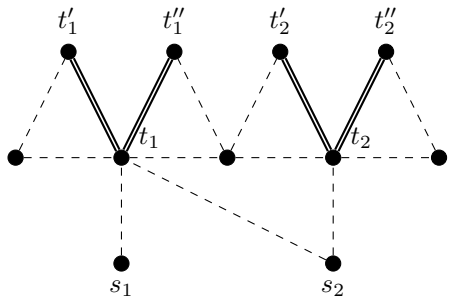


Figure 6 An example for the sets T and U. The vertices t_1, t_2 are in T, the vertices t'_1, t''_1, t'_2, t''_2 are the corresponding internal vertices in U, and s_1, s_2 are in S. Edges contained in $G[U]$ are bold.

Because every vertex of T is adjacent to S, there is a set $(Y_s)_{s\in S}$ of stars with the following properties:

- Every star Y_s is a vertex-induced subgraph of G^r centered in s and its leafs are from T.
- $\bigcup_{s\in S} V(Y_s)$ covers T.

Now we enhance the stars Y_s by replacing every leaf t with U_t, i.e., we define Z_s to be the subgraph of G^r induced by s and U_t for every $t \in T$ that is a leaf of Y_s. Then every $Z_s, s \in S$, is a forest with $|E(Z_s)| = 3 \cdot |E(Y_s)| = 3 \cdot |V(Y_s) \cap T|$ many edges. By the block additivity of Π, it follows $\mathsf{ex}(Z_s) = \frac{3\cdot|V(Y_s)\cap T|}{4}$ for every $s \in S$.

Repeated application of Proposition 6.2 yields

$$\mathsf{ex}(G[Z]) \geq \sum_{s\in S} \frac{3 \cdot |V(Y_s) \cap T|}{4} - \frac{|S|}{4} = \frac{3 \cdot |T| - |S|}{4},$$

where $Z := \bigcup_{s\in S} V(Z_s)$. This means that

$$\mathsf{ex}(G^r) \geq \mathsf{ex}(G[Z]) - \frac{c+1}{4} \geq \frac{3 \cdot |T| - |S| - c - 1}{4},$$

where c is the number of connected components of $G' := G - Z$. Now it suffices to show that $c \leq |T| + |\mathcal{B}^+| + |\tilde{\mathcal{B}}|$, since then

$$\mathsf{ex}(G^r) \geq \frac{2 \cdot |T| - |\mathcal{B}^+| - |\tilde{\mathcal{B}}| - O(k)}{4},$$

i.e., $|T| = O(|\mathcal{B}^+| + |\tilde{\mathcal{B}}| + k)$ or (G^r, k) is a "yes"-instance.

Let us first bound the number of connected components of $G^r - S$. It is clear that $G^r - S$ can only have $|\mathcal{B}^+|$ many connected components that contain a block that is not bad. Let now W be a connected component of $G^r - S$ that contains only bad blocks. If W contains only one block, then this block is special. Otherwise, W contains at least two leaf blocks, and because only one of the (bad) blocks of W can have size 2, there is a leaf block B in W with $|V(B)| = 3$. Because B was not eliminated by Rule 18, it must be special. Hence, the number of connected components of $G^r - S$ containing only bad blocks is bounded by $|\tilde{\mathcal{B}}|$ and thus the total number of connected components of $G^r - S$ is bounded by $|\mathcal{B}^+| + |\tilde{\mathcal{B}}|$.

Because every vertex from $U \setminus T$ is an internal vertex in $G^r - S$, the removal of these vertices from $G^r - S$ cannot increase the number of connected components. Furthermore, as every vertex from T is contained in exactly two blocks of $G^r - S$, its removal can increase the number of connected components by at most 1. Hence, the number c of connected components of $G - Z$ is at most the number of connected components of $G - S$ plus $|T|$. This completes the proof. ◀

▶ **Theorem 7.15.** *Let Π be a hereditary strongly $\frac{1}{2}$-extendible property on oriented graphs with $\vec{K}_3 \in \Pi$. Then* Above Poljak-Turzík(Π) *admits a kernel with $O(k)$ vertices.*

Proof. Let (G^0, k) be an instance of Above Poljak-Turzík(Π). Apply Rules 14-17 exhaustively, producing an instance (G', k') and a vertex set S. If $k' \leq 0$, then (G', k') and thus also (G^0, k) is a "yes"-instance. Otherwise, $|S| = O(k)$.

Now apply Rules 18-19 exhaustively on (G^0, k) to obtain an equivalent instance (G^r, k). Let Γ be the set of bad vertices of G^r. Lemma 7.3 shows that $|V(G^r) \setminus \Gamma| = O(k)$ or (G^r, k) is a "yes"-instance. Because every bad vertex is in a bad block and every bad block contains at

most three vertices, it holds $|\Gamma| = O(|\mathcal{B}^-|)$. Using Lemma 7.11, Lemma 7.13, and Lemma 7.14, we can bound this cardinality by

$$|\mathcal{B}^-| = O(|\mathcal{B}^+| + |\tilde{\mathcal{B}}| + |R|) = O(|\mathcal{B}^+| + |\tilde{\mathcal{B}}| + k) = O(k)$$

or (G^r, k) is a "yes"-instance. This shows the theorem. ◀

We are ready to complete the proof of Theorem 2.7.

Proof of Theorem 2.7. Let $\lambda \in (0, 1)$ and let Π be a strongly λ-extendible property of (possibly oriented and/or labelled) graphs. If $\lambda \neq \frac{1}{2}$ or $G \in \Pi$ for every G with $\langle G \rangle = K_3$, we can use Theorem 7.6. Otherwise, we only have to consider the case that Π is a hereditary property of simple or oriented graphs.

Consider the case that $\overrightarrow{K}_3 \in \Pi$ or $\overrightarrow{\overrightarrow{K}}_3 \in \Pi$. If $\overrightarrow{K}_3 \in \Pi$, then Crowston et al. [26] show that $\overrightarrow{\overrightarrow{K}}_3 \in \Pi$, i.e., we can use Theorem 7.6. And if $\overrightarrow{\overrightarrow{K}}_3 \in \Pi$, we use Theorem 7.15.

Now we may suppose that $G \notin \Pi$ for every G with $\langle G \rangle = K_3$. Then Crowston et al. [26] show that Π is the set of all bipartite graphs. Hence, in the case of simple graphs as well as if $\overrightarrow{K}_3, \overrightarrow{\overrightarrow{K}}_3 \notin \Pi$ for oriented graphs, we can use Theorem 2.6 to obtain a linear vertex kernel.

It is easy to see that Rules 18-19 can be applied exhaustively in time $O(m)$. As λ is constant and we can apply every other reduction rule in linear time, it follows a total running time of $O(\lambda \cdot km) = O(km)$. ◀

8 Polynomial Kernel for Weighted Problems

In this section we use a theorem of Frank and Tardos [54] to obtain kernels for weighted problems. Section 8.1 is devoted to the introduction of the technique by showing deterministic kernels for Subset Sum(n) and Knapsack(n). We present polynomial kernels for various weighted problems in Section 8.2. Next, we give positive and negative results for the Knapsack problem and its special case Subset Sum with only few different item sizes in Section 8.3. Finally, we consider polynomial ILPs in Section 8.4.

8.1 Settling Open Problems via the Frank-Tardos Theorem

8.1.1 Frank and Tardos' theorem

Frank and Tardos [54] describe an algorithm which proves the following theorem.

▶ **Theorem 8.1** ([54]). *There is an algorithm that, given a vector $w \in \mathbb{Q}^r$ and an integer N, in polynomial time finds a vector $\overline{w} \in \mathbb{Z}^r$ with $\left\|\overline{w}\right\|_\infty \leq 2^{4r^3} N^{r(r+2)}$ such that $\operatorname{sign}(w \cdot b) = \operatorname{sign}(\overline{w} \cdot b)$ for all vectors $b \in \mathbb{Z}^r$ with $\left\|b\right\|_1 \leq N-1$.*

This theorem allows us to compress linear inequalities to an encoding length which is polynomial in the number of variables. Frank and Tardos' algorithm runs even in strongly polynomial time. As a consequence, all kernelizations presented in this work also have a strongly polynomial running time.

▶ **Corollary 8.2.** *There is an algorithm that, given a vector $w \in \mathbb{Q}^r$ and a rational $W \in \mathbb{Q}$, in polynomial time finds a vector $\overline{w} \in \mathbb{Z}^r$ with $\left\|\overline{w}\right\|_\infty = 2^{\mathcal{O}(r^3)}$ and an integer $\overline{W} \in \mathbb{Z}$ with total encoding length $\mathcal{O}(r^4)$, such that $w \cdot x \leq W$ if and only if $\overline{w} \cdot x \leq \overline{W}$ for every vector $x \in \{0,1\}^r$.*

Proof. Use Theorem 8.1 on the vector $(w, W) \in \mathbb{Q}^{r+1}$ with $N = r+2$ to obtain the resulting vector $(\bar{w}, \bar{W})$. Now let $b = (x, -1) \in \mathbb{Z}^{r+1}$ and note that $\left\|b\right\|_1 \leq N-1$. The inequality $w \cdot x \leq W$ is false if and only if $\operatorname{sign}(w \cdot x - W) = \operatorname{sign}((w, W) \cdot (x, -1)) = \operatorname{sign}((w, W) \cdot b)$ is equal to $+1$. The same holds for $\overline{w} \cdot x \leq \overline{W}$.

As each $|\bar{w}_i|$ can be encoded with $\mathcal{O}(r^3 + r^2 \log N) = \mathcal{O}(r^3)$ bits, the whole vector $\bar{w}$ has encoding length $\mathcal{O}(r^4)$. ◀

Note that also $w \cdot x \geq W$ if and only if $\overline{w} \cdot x \geq \overline{W}$ for every $x \in \{0,1\}^r$. Therefore, also $w \cdot x = W$ if and only if $\overline{w} \cdot x = \overline{W}$.

8.1.2 Polynomial Kernelization for Knapsack

A first easy application of Corollary 8.2 is the kernelization of Knapsack with the number n of different items as parameter.

Knapsack(n)	
Input:	An integer $n \in \mathbb{N}$, rationals $W, P \in \mathbb{Q}$, a weight vector $w \in \mathbb{Q}^n$, and a profit vector $p \in \mathbb{Q}^n$.
Parameter:	n.
Question:	Is there a vector $x \in \{0,1\}^n$ with $w \cdot x \leq W$ and $p \cdot x \geq P$?

▶ **Theorem 8.3.** Knapsack(n) *admits a kernel of size $\mathcal{O}(n^4)$.* ◀

As a consequence, also the special case Subset Sum(n) admits a kernel of size $\mathcal{O}(n^4)$.

8.2 Small Kernels for Weighted Parameterized Problems

The result of Frank and Tardos implies that we can easily handle large weights or numbers in kernelization provided that the number of different objects is already sufficiently small (e.g., polynomial in the parameter). In the present subsection we show how to handle the first step, i.e., the reduction of the number of objects, in the presence of weights for a couple of standard problems. Presumably the reduction in size of numbers is not useful for this first part since the number of different values is still exponential.

8.2.1 Hitting Set and Set Packing

In this subsection we outline how to obtain polynomial kernelizations for Weighted d-Hitting Set and Weighted d-Set Packing. Since these problems generalize quite a few interesting hitting/covering and packing problems, this extends the list of problems whose weighted versions directly benefit from our results. The problems are formally defined as follows, where $\binom{U}{d}$ denotes the set of all subsets of U with exactly d elements.

Weighted d-Hitting Set(k)	
Input:	A set family $\mathcal{F} \subseteq \binom{U}{d}$, a function $w\colon U \to \mathbb{N}$, and $k, W \in \mathbb{N}$.
Parameter:	k.
Question:	Is there a set $S \subseteq U$ of cardinality at most k and weight $\sum_{u \in S} w(u) \leq W$ such that S intersects every set in $\mathcal{F}$?

Weighted d-Set Packing(k)	
Input:	A set family $\mathcal{F} \subseteq \binom{U}{d}$, a function $w\colon \mathcal{F} \to \mathbb{N}$, and $k, W \in \mathbb{N}$.
Parameter:	k.
Question:	Is there a family $\mathcal{F}^* \subseteq \mathcal{F}$ of exactly k pairwise disjoint sets of weight $\sum_{F \in \mathcal{F}^*} w(F) \geq W$?

Note that we treat d as a constant. We point out that the definition of Weighted Set Packing(k) restricts attention to exactly k disjoint sets of weight at least W. If we were to relax to at least k sets then the problem would be NP-hard already for $k = 0$, as there would be no restriction at all. On the other hand, the kernelization that we present for Weighted Set Packing(k) holds also if we require $\mathcal{F}^*$ to be of cardinality at most k (and total weight at least W, as before).

Both kernelizations rely on the Sunflower Lemma of Erdős and Rado [39], same as their unweighted counterparts. We recall the lemma.

▶ **Lemma 8.4** (Erdős and Rado [39]). *Let $\mathcal{F}$ be a family of sets, each of size d, and let $k \in \mathbb{N}$. If $|\mathcal{F}| > d!k^d$ then we can find in time $\mathcal{O}(|\mathcal{F}|)$ a so-called $k+1$-sunflower, consisting of $k+1$ sets $F_1, \ldots, F_{k+1} \in \mathcal{F}$ such that the pairwise intersection of any two F_i, F_j with $i \neq j$ is the same set C, called the core.*

▶ **Theorem 8.5.** Weighted d-Hitting Set(k) *admits a kernelization to $\mathcal{O}(k^d)$ sets and total size bounded by $\mathcal{O}(k^{4d})$.*

Proof. We can apply the Sunflower Lemma directly, same as for the unweighted case: Say we are given $(U, \mathcal{F}, w, k, W)$. If the size of $\mathcal{F}$ exceeds $d!(k+1)^d$ then we find a $(k+2)$-sunflower $\mathcal{F}_s$ in $\mathcal{F}$ with core C. Any hitting set of cardinality at most k must contain an element of C. The same is true for $k+1$-sunflowers so we may safely delete any set $F \in \mathcal{F}_s$ since hitting the

set $C \subseteq F$ is enforced by the remaining $k+1$-sunflower. Iterating this reduction rule yields $\mathcal{F}' \subseteq \mathcal{F}$ with $|\mathcal{F}'| = \mathcal{O}(k^d)$ and such that $(U, \mathcal{F}, w, k, W)$ and $(U, \mathcal{F}', w, k, W)$ are equivalent.

Now, we can apply Theorem 8.1. We can safely restrict U to the elements U' present in sets of the obtained set family $\mathcal{F}'$, and let $w' = w|_{U'}$. By Theorem 8.1 applied to weights w' and target weight W with $N = k+2$ and $r = \mathcal{O}(k^d)$ we get replacement weights of magnitude bounded by $2^{\mathcal{O}(k^{3d})} N^{\mathcal{O}(k^{2d})}$ and bit size $\mathcal{O}(k^{3d})$. Note that this preserves, in particular, whether the sum of any k weights is at most the target weight W, by preserving the sign of $w_{i_1} + \ldots + w_{i_k} - W$. The total bitsize is dominated by the space for encoding the weight of all elements of the set U'. ◀

Setting $d = 2$, we derive a kernelization to $\mathcal{O}(k^2)$ edges with total encoding size $\mathcal{O}(k^8)$ for Weighted Vertex Cover(k). For Weighted d-Set Packing(k) a similar argument works.

▶ **Theorem 8.6.** Weighted d-Set Packing(k) *admits a kernelization to* $\mathcal{O}(k^d)$ *sets and total size bounded by* $\mathcal{O}(k^{4d})$.

Proof. If the size of $\mathcal{F}$ exceeds $d!(dk)^d$ then we find a $dk+1$-sunflower $\mathcal{F}_s$ in $\mathcal{F}$ with core C. We argue that we can safely discard the set $F_0 \in \mathcal{F}_s$ of least weight according to $w\colon \mathcal{F} \to \mathbb{N}$: This could only fail if there is a solution that includes F_0, namely k disjoint sets $F_0, \ldots, F_{k-1}$ of total weight at least W. Notice that no set $F_1, \ldots, F_{k-1}$ can contain C, since $C \subseteq F_0$. Since $|\mathcal{F}_s| = dk+1$ there must be another set F_k, apart from F_0, that has an empty intersection with $F_1, \ldots, F_{k-1}$, as the sets in $\mathcal{F}_s$ are disjoint apart from C and there are in total $d(k-1)$ elements in $F_1, \ldots, F_{k-1}$. It follows that $F_1, \ldots, F_k$ is also a selection of k disjoint sets. Since F_0 is the lightest set in $\mathcal{F}_s$ we must have that the total weight of $F_1, \ldots, F_k$ is at least W.

Iterating this rule gives $|\mathcal{F}| = \mathcal{O}(k^d)$. Again, it suffices to preserve how the sum of any k weights compares with W. Thus, we get the same bound of $\mathcal{O}(k^{3d})$ bits per element (of $\mathcal{F}$, in this case). ◀

8.2.2 Max-Cut

Let us derive a polynomial kernel for Weighted Max-Cut(W), which is defined as follows.

Weighted Max-Cut(W)	
Input:	A graph G, a function $w\colon E \to \mathbb{Q}_{\geq 1}$, and $W \in \mathbb{Q}_{\geq 1}$.
Parameter:	$\lceil W \rceil$.
Question:	Is there a set $C \subseteq V(G)$ such that $\sum_{e \in \delta(C)} w(e) \geq W$?

Note that we chose the weight of the resulting cut as parameter, which is most natural for this problem. The number k of edges in a solution is not a meaningful parameter: If we restricted the cut to have at least k edges, then for $k = 0$ we would consider the general Weighted Max-Cut problem, which is NP-hard. If we required at most k edges, we could, in this example for integral weights, multiply all edge weights by n^2 and add arbitrary edges with weight 1 to our input graph. When setting the new weight bound to $n^2 \cdot W + \binom{n}{2}$, we would not change the instance semantically but there may be no feasible solution left with at most k edges.

The restriction to edge weights at least 1 is necessary as otherwise the problem becomes intractable. This is because when allowing arbitrary positive rational weights, we can transform instances of the NP-complete Unweighted Max-Cut problem (with all weights equal to 1

and parameter k, which is the number of edges in the cut) to instances of the Weighted Max-Cut problem on the same graph with edge weights all equal to $1/k$ and parameter $W = 1$.

▶ **Theorem 8.7.** Weighted Max-Cut(W) *admits a kernel of size* $\mathcal{O}(W^4)$.

Proof. Let T be the total weight of all edges. If $T \geq 2W$, then the greedy algorithm yields a cut of weight at least $T/2 \geq W$. Therefore, all instances with $T \geq 2W$ can be reduced to a constant-size positive instance. Otherwise, there are at most $2W$ edges in the input graph as every edge has weight at least 1. Thus, we can use Theorem 8.1 to obtain an equivalent (integral) instance of encoding length $\mathcal{O}(W^4)$. ◀

8.2.3 Bin Packing with Additive Error

Bin Packing is another classical NP-hard problem involving numbers. Therein, we are given n positive integer numbers $a_1, \ldots, a_n$ (the items), a bin size $b \in \mathbb{N}$, and an integer k; the question is whether the integer numbers can be partitioned into at most k sets, the bins, each of sum at most b. From a parameterized perspective the problem is highly intractable for its natural parameter k, because for $k = 2$ it generalizes the (weakly) NP-hard Partition problem.

Jansen et al. [71] proved that the parameterized complexity improves drastically if instead of insisting on exact solutions the algorithm only has to provide a packing into $k + 1$ bins or correctly state that k bins do not suffice. Concretely, it is shown that this problem variant is fixed-parameter tractable with respect to k. The crucial effect of the relaxation is that small items are of almost no importance: If they cannot be added greedily "on top" of a feasible packing of big items into $k + 1$ bins, then the instance trivially has no packing into k bins due to exceeding total weight kb. Revisiting this idea, with a slightly different threshold for being a small item, we note that after checking for total weight being at most kb (else reporting that there is no k-packing) we can safely discard all small items before proceeding. Crucially, this cannot turn a "no"- into a "yes"-instance because the created $k + 1$-packing could then also be lifted to one for all items (contradicting the assumed "no"-instance). An application of Theorem 8.1 then yields a polynomial kernelization because we can have only few large items.

Additive One Bin Packing(k)	
Input:	Item sizes $a_1, \ldots, a_n \in \mathbb{N}$, a bin size $b \in \mathbb{N}$, and $k \in \mathbb{N}$.
Parameter:	k.
Task:	Give a packing into at most $k + 1$ bins of size b, or correctly state that k bins do not suffice.

▶ **Theorem 8.8.** Additive One Bin Packing(k) *admits a polynomial kernelization to* $\mathcal{O}(k^2)$ *items and bit size* $\mathcal{O}(k^8)$.

Proof. Let an instance $(a_1, \ldots, a_n, b, k)$ be given. If any item size a_i exceeds b, or if the total weight of items a_i exceeds $k \cdot b$, then we may safely answer that no packing into k bins is possible. In all other cases the kernelization will return an instance whose answer will be correct for the original instance: if it reports a $(k + 1)$-packing then the original instance has a $(k + 1)$-packing. If it reports that no k-packing is possible then the same holds for the original instance.

Assume that the items a_i are sorted decreasingly by value. Consider the subsequence, say, $a_1, \ldots, a_\ell$, of items of size at least $\frac{b}{k+1}$. If the instance restricted to these items permits

a packing into at most $k+1$ bins, then we show that the items $a_{\ell+1}, \ldots, a_n$ can always be added, giving a $(k+1)$-packing for the input instance: assume that a greedy packing of the small items into the existing packing for $a_1, \ldots, a_\ell$ fails. This implies that some item, say a_i, of size less than $\frac{b}{k+1}$ does not fit. But then all bins have less than $\frac{b}{k+1}$ remaining space. It follows that the total packed weight, excluding a_i, is more than

$$(k+1) \cdot \left(b - \frac{b}{k+1}\right) = (k+1)b - b = kb \ .$$

This contradicts the fact that this part of the kernelization is only run if the total weight is at most kb. Thus, a $k+1$-packing for $a_1, \ldots, a_\ell$ implies a $k+1$-packing for the whole set $a_1, \ldots, a_n$.

Clearly, if the items $a_1, \ldots, a_\ell$ permit no packing into k bins then the same is true for the whole set of items.

Observe now that ℓ cannot be too large: Indeed, since the total weight is at most kb (else we returned "no" directly), there can be at most

$$\frac{kb}{\frac{b}{k+1}} = k(k+1) = \mathcal{O}(k^2)$$

items of weight at least $\frac{b}{k+1}$. Thus, an application of Corollary 8.2 yields a size of $\mathcal{O}(k^6)$ per large item and a total encoding size of $\mathcal{O}(k^8)$. ◀

8.3 Kernel Bounds for Knapsack Problems

In this subsection we provide lower and upper bounds for kernel sizes for variants of the Knapsack problem.

8.3.1 Exponential Kernel for Knapsack with Few Item Sizes

First, consider the Subset Sum problem restricted to instances with only k distinct item weights, which are not restricted in any other way (except for being non-negative integers). Then the problem can be solved by a fixed-parameter algorithm for parameter k by a reduction to integer linear programming in fixed dimension, and applying Lenstra's algorithm [78] or one of its improvements [73, 54]. This was first observed by Fellows et al. [51]. Let us restate the integer linear programming result, as we will use it several times.

▶ **Theorem 8.9** ([54]). Integer Linear Programming *on n variables and input length s can be solved using $s \cdot n^{2.5n+o(n)}$ arithmetic operations.*

We now generalize the results by Fellows et al. [51] to Knapsack with few item weights. More precisely, we are given an instance I of the Knapsack problem consisting of n items that have only k distinct item weights; however, the number of item values is unbounded. This means in particular, that the "number of numbers" is not bounded as a function of the parameter, making the results by Fellows et al. [51] inapplicable.

▶ **Theorem 8.10.** *The* Knapsack *problem with k distinct weights can be solved in time $k^{2.5k+o(k)} \cdot \mathrm{poly}(|I|)$, where $|I|$ denotes the encoding length of the instance.*

Proof. Observe that when packing x_i items of weight w_i, it is optimal to pack the x_i items with largest value among all items of weight w_i. Assume the items of weight w_i are labeled as $j_1^{(i)}, \ldots, j_{n_i}^{(i)}$ by non-increasing values. For each $s \in \mathbb{N}$, define $f_i(s) := \sum_{\ell=1}^{s} v(j_\ell^{(i)})$, where

$v(j_\ell^{(i)})$ denotes the value of item $j_\ell^{(i)}$. We can formulate the knapsack problem as the following program, in which variable x_i encodes how many items of weight w_i are packed into the knapsack and g_i encodes their total value:

$$\begin{aligned} \max \sum_{i=1}^{k} g_i \quad \text{s.t.} \quad & \sum_{i=1}^{k} w_i \cdot x_i \leq W, \\ & g_i \leq f_i(x_i), && i = 1, \ldots, k, \\ & x_i \in \{0, 1, \ldots, n_i\},\ g_i \in \mathbb{N}_0, && i = 1, \ldots, k. \end{aligned}$$

The functions f_i are in general non-linear, but they can be replaced by n_i many linear functions, as the following lemma shows.

▶ **Lemma 8.11.** *For each i there exists a set of linear functions $p_i^{(1)}, \ldots, p_i^{(n_i)}$ such that $f_i(s) = \min_\ell p_i^{(\ell)}(s)$ for every $s \in \{0, \ldots, n_i\}$.*

Proof. For each $\ell \in \{1, \ldots, n_i\}$ we define $p_i^{(\ell)}(s)$ to be the unique linear function such that

$$p_i^{(\ell)}(\ell - 1) = f_i(\ell - 1) \quad \text{and} \quad p_i^{(\ell)}(\ell) = f_i(\ell).$$

The function $f_i(s)$ is concave because

$$f_i(\ell + 1) - f_i(\ell) = v(j_{\ell+1}^{(i)}) \leq v(j_\ell^{(i)}) = f_i(\ell) - f_i(\ell - 1)$$

for each $\ell \in \{1, \ldots, n_i - 1\}$. Therefore, the definition of the linear functions $p_i^{(\ell)}$ implies that $f_i(s) \leq p_i^{(\ell)}(s)$ for every $\ell \in \{1, \ldots, n_i\}$ and $s \in \{0, \ldots, n_i\}$. Since for each $s \in \{1, \ldots, n_i\}$ we have that $p_i^{(s)}(s) = f_i(s)$ and $p_i^{(1)}(0) = f_i(0)$, we conclude that $f_i(s) = \min_\ell p_i^{(\ell)}(s)$ for every $s \in \{0, \ldots, n_i\}$. ◀

Hence in the program above, we can, for every $i \in \{1, \ldots, k\}$, replace the constraint $g_i \leq f_i(x_i)$ by the set of constraints $g_i \leq p_i^{(\ell)}(x_i)$ for $\ell \in \{1, \ldots, n_i\}$. This way we obtain a formulation of the knapsack problem as an integer linear program with $2k$ variables. The encoding length of this integer linear program is polynomially bounded in the encoding length of the instance of Knapsack. Together with Theorem 8.9 this implies the fixed-parameter tractability of Knapsack with k item weights. ◀

8.3.2 Polynomial Kernel for Subset Sum with Few Item Sizes

We now improve the work of Fellows et al. [51] in another direction. Namely, we show that the Subset Sum problem admits a polynomial kernel for parameter the number k of item weights; this improves upon the exponential-size kernel due to Fellows et al. [51]. To show the kernel bound of $k^{\mathcal{O}(1)}$, consider an instance I of Subset Sum with n items that have only k distinct item weights. For each item weight s_i, let n_i be its multiplicity, that is, the number of items in I of weight s_i. Given I, we formulate an ILP for the task of deciding whether some subset S of items has weight exactly t. The ILP simply models for each item weight s_i the number of items $x_i \leq n_i$ selected from it as to satisfy the subset sum constraint:

$$\left.\begin{aligned} s_1 x_1 + \ldots + s_k x_k &= t, \\ 0 \leq x_i &\leq n_i, \quad i = 1, \ldots, k, \\ x_i &\in \mathbb{N}_0, \quad i = 1, \ldots, k \ . \end{aligned}\right\} \tag{5}$$

Then (5) is an Integer Linear Programming instance on $m = 1$ relevant constraint and each variable x_i has maximum range bound $u = \max_i n_i \leq n$.

Now consider two cases:

- If $\log n \leq k \cdot \log k$, then we apply Theorem 8.1 to (5) to reduce the instance to an equivalent instance I' of size $\mathcal{O}(k^4 + k^3 \log n) = \mathcal{O}(k^4 + k^3 \cdot (k \log k)) = \mathcal{O}(k^4 \log k)$ with item weights $s'_1, \ldots, s'_k$.
  We now transform I' to an equivalent Subset Sum instance with fewer items: Given n_i items with weight s'_i in I', we remove these items from the instance and add items with weights $2^j \cdot s'_i$ for $0 \leq j \leq \ell_i$ and $\left(n_i - \sum_{j=0}^{\ell_i} 2^j\right) \cdot s'_i$, where ℓ_i is the largest integer such that $\sum_{j=0}^{\ell_i} 2^j < n_i$ (add one item per weight). It is now easy to check that for every number $0 \leq x_i \leq n_i$ of chosen items with weight s'_i in the original instance, there is a combination of the new items with the same total weight and vice versa (consider the binary representation of x_i).
  If we repeat this argument for every item weight s_i, we obtain an equivalent instance with $\mathcal{O}(k \log n) = \mathcal{O}(k^2 \log k)$ items each with a weight which can be encoded in length $\mathcal{O}(k^3 + k^2 \log n + \log n) = \mathcal{O}(k^3 \log k)$. Therefore we obtain an equivalent instance with encoding length $\mathcal{O}((k^2 \log k) \cdot k^3 \log k) = \mathcal{O}(k^5 \log^2 k)$. If we are allowed to encode the instance by writing the different item weights followed by the multiplicity of every item weight, encoded in binary, we have even obtained a kernel of size $\mathcal{O}(k \cdot k^3 \log k) = \mathcal{O}(k^4 \log k)$.
- If $k \log k \leq \log n$, then we solve the integer linear program (5) using Theorem 8.9 in time $k^{2.5k+o(k)} \cdot |I|$. As $k^k \leq n \leq |I|$, this means that we can solve the ILP (and hence decide the instance I) in polynomial time $k^{2.5k+o(k)} \cdot |I| = |I|^{\mathcal{O}(1)}$.

In summary, we have shown the following:

▶ **Theorem 8.12.** Subset Sum *with k item weights admits a kernel of size $\mathcal{O}(k^5 \log^2 k)$. Moreover, it admits a kernel of size $\mathcal{O}(k^4 \log k)$ if the multiplicities of the item weights can be encoded in binary.*

We remark that this method does not work if the instance I is succinctly encoded by specifying the k distinct item weights w_i in binary and for each item weight s_i its multiplicity n_i in binary: then the running time of Theorem 8.9 is exponential in k and the input length of the subset sum instance, which is $\mathcal{O}(k \cdot \log n)$.

8.3.3 A Kernelization Lower Bound for Subset Sum

In the following we show a kernelization lower bound for Subset Sum assuming the *Exponential Time Hypothesis*. The Exponential Time Hypothesis [69] states that there does not exist a $2^{o(n)}$-time algorithm for 3-SAT, where n denotes the number of variables.

▶ **Lemma 8.13.** Subset Sum *does not admit a $2^{o(n)}$-time algorithm assuming the Exponential Time Hypothesis, where n denotes the number of numbers.*

Proof. Gurari [59, Theorem 5.4.1] gives a polynomial-time reduction that transforms any 3-SAT formula ϕ with n variables and m clauses into an equivalent instance of Subset Sum with exactly $2n + 2m + 1$ numbers. Assume there is an algorithm for Subset Sum that runs in time $2^{o(\ell)}$, where ℓ denotes the number of numbers. With Gurari's reduction we could use this algorithm to decide whether or not a 3-SAT formula ϕ is satisfiable in time $2^{o(n+m)}$. Due to the sparsification lemma of Impagliazzo et al. [69], this contradicts the Exponential Time Hypothesis. ◀

▶ **Theorem 8.14.** Subset Sum *does not admit kernels of size* $\mathcal{O}(n^{2-\varepsilon})$ *for any* $\varepsilon > 0$ *assuming the Exponential Time Hypothesis, where* n *denotes the number of numbers.*

Proof. Assume there exists a kernelization algorithm A for Subset Sum that produces instances of size at most $\kappa n^{2-\varepsilon}$ for some $\kappa > 0$ and some $\varepsilon > 0$. We show that A can be utilized to solve Subset Sum in time $2^{o(n)}$, which contradicts the Exponential Time Hypothesis due to Lemma 8.13.

Let I be an arbitrary Subset Sum instance with n items. We apply the kernelization algorithm A to obtain an equivalent instance I' whose encoding size is at most $\kappa n^{2-\varepsilon}$. Let $a_1, \ldots, a_m$ be the numbers in I' and let c be the target value.

Let $k = n^{1-\varepsilon/2}$. We divide the numbers in I' into two groups: a number a_i is called *heavy* if $a_i \geq 2^k$ and *light* otherwise. Since one needs at least k bits to encode a heavy number, the number of heavy numbers is bounded from above by $\kappa n^{2-\varepsilon}/k = \kappa n^{1-\varepsilon/2}$.

We solve instance I' as follows: for each subset J_H of heavy numbers, we determine whether or not there exists a subset J_L of light numbers such that $\sum_{i \in J_L \cup J_H} a_i = c$ via dynamic programming. Since there are at most $\kappa n^{1-\varepsilon/2}$ heavy numbers, there are at most $2^{\kappa n^{1-\varepsilon/2}}$ subsets J_H. The dynamic programming algorithm runs in time $\mathcal{O}(m^2 \cdot 2^{n^{1-\varepsilon/2}})$, as each of the at most m light numbers is bounded from above by $2^{n^{1-\varepsilon/2}}$. Hence, instance I' can be solved in time $\mathcal{O}(m^2 \cdot 2^{(1+\kappa)n^{1-\varepsilon/2}}) = 2^{o(n)}$, where the equation follows because $m \leq \kappa n^{2-\varepsilon} = 2^{o(n)}$. ◀

8.4 Integer Polynomial Programming with Bounded Range

Up to now, we used Frank and Tardos' result only for linear inequalities with mostly binary variables. But it also turns out to be useful for more general cases, namely for polynomial inequalities with integral bounded variables. We use this to show that Integer Polynomial Programming instances can be compressed if the variables are bounded. As a special case, Integer Linear Programming admits a polynomial kernel in the number of variables if the variables are bounded.

Let us first transfer the language of Theorem 8.1 to arbitrary polynomials.

▶ **Lemma 8.15.** *Let* $f \in \mathbb{Q}[X_1, \ldots, X_n]$ *be a polynomial of degree at most* d *with* r *non-zero coefficients, and let* $u \in \mathbb{N}$. *Then one can efficiently compute a polynomial* $\tilde{f} \in \mathbb{Z}[X_1, \ldots, X_n]$ *of encoding length* $\mathcal{O}\big(r^4 + r^3 d \log(ru) + rd \log(nd)\big)$ *such that* $\mathit{sign}(f(x) - f(y)) = \mathit{sign}(\tilde{f}(x) - \tilde{f}(y))$ *for all* $x, y \in \{-u, \ldots, u\}^n$.

Proof. Let $w_1, \ldots, w_r \in \mathbb{Q}$ and $f_1, \ldots, f_r \in \mathbb{Q}[X_1, \ldots, X_n]$ be pairwise distinct monomials with coefficient 1 such that $f = \sum_{i=1}^{r} w_i \cdot f_i$. Apply Theorem 8.1 to $w = (w_1, \ldots, w_r)$ and $N = 2ru^d + 1$ to obtain $\tilde{w} = (\tilde{w}_1, \ldots, \tilde{w}_r) \in \mathbb{Z}^r$. Set $\tilde{f} = \sum_{i=1}^{r} \tilde{w}_i \cdot f_i$.

The encoding length of each $\tilde{w}_i$ is upper bounded by $\mathcal{O}(r^3 + r^2 \log N) = \mathcal{O}(r^3 + r^2 \cdot d \cdot \log(r \cdot u))$. As there are $\binom{n+d}{d}$ monomials of degree at most d, the information to which monomial a coefficient belongs can be encoded in $\mathcal{O}(\log((n+d)^d)) = \mathcal{O}(d \log(nd))$ bits. Hence, the encoding length of $\tilde{f}$ is upper bounded by

$$\mathcal{O}\big(r^4 + r^3 d \log(ru) + rd \log(nd)\big).$$

To prove the correctness of our construction, let $x, y \in \{-u, \ldots, u\}^n$. For $1 \leq i \leq r$, set $b_i = f_i(x) - f_i(y) \in \mathbb{Z} \cap [-2u^d, 2u^d]$, and set $b = (b_1, \ldots, b_r)$. Then $\|b\|_1 \leq r \cdot 2u^d$, and thus by Theorem 8.1, $\operatorname{sign}(w \cdot b) = \operatorname{sign}(\tilde{w} \cdot b)$. Then also $\operatorname{sign}(f(x) - f(y)) = \operatorname{sign}(\tilde{f}(x) - \tilde{f}(y))$, as

$$\tilde{f}(x) - \tilde{f}(y) = \sum_{i=1}^{r} \tilde{w}_i \cdot (f_i(x) - f_i(y)) = \sum_{i=1}^{r} \tilde{w}_i \cdot b_i = \tilde{w} \cdot b,$$

and

$$f(x) - f(y) = \sum_{i=1}^{r} w_i \cdot (f_i(x) - f_i(y)) = \sum_{i=1}^{r} w_i \cdot b_i = w \cdot b.$$

This completes the proof of the lemma. ◀

We use this lemma to compress Integer Polynomial Programming instances.

Integer Polynomial Programming	
Input:	Polynomials $c, g_1, \ldots, g_m \in \mathbb{Q}[X_1, \ldots, X_n]$ of degree at most d encoded by the coefficients of the $\mathcal{O}(n^d)$ monomials, rationals $b_1, \ldots, b_m, z \in \mathbb{Q}$, and $u \in \mathbb{N}$.
Question:	Is there a vector $x \in \{-u, \ldots, u\}^n$ with $c(x) \leq z$ and $g_i(x) \leq b_i$ for $i = 1, \ldots, m$?

▶ **Theorem 8.16.** *Every* Integer Polynomial Programming *instance in which c and each g_i consist of at most r monomials can be efficiently compressed to an equivalent instance of encoding length $\mathcal{O}\big(m(r^4 + r^3 d \log(ru) + rd \log(nd))\big)$.*

Proof. Define $c', g'_1, \ldots, g'_m \colon \mathbb{Z}^n \times \{0,1\} \to \mathbb{Q}$ as

$$c'(x, y) := c(x) + y \cdot z,$$
$$g'_i(x, y) := g'_i(x) + y \cdot b_i \quad i = 1, \ldots, m.$$

Now apply Lemma 8.15 to c' and $g'_1, \ldots, g'_m$ to obtain $\tilde{c'}$ and $\tilde{g}'_1, \ldots, \tilde{g}'_m$. Thereafter, split these functions up into their parts $(\tilde{c}, \tilde{z})$ and $(\tilde{g}_1, \tilde{b}_1), \ldots, (\tilde{g}_m, \tilde{b}_m)$. We claim that the instance $\tilde{I} = (\tilde{c}, \tilde{g}_1, \ldots, \tilde{g}_m, d, \tilde{b}_1, \ldots, \tilde{b}_m, \tilde{z}, u)$ is equivalent to I. To see this, we have to show that a vector $x \in \{-u, \ldots, u\}^n$ satisfies $c(x) \leq z$ if and only if it satisfies $\tilde{c}(x) \leq \tilde{z}$ (and analogously $g_i(x) \leq b_i$ if and only if $\tilde{g}_i(x) \leq \tilde{b}_i$ for all i). This follows from

$$\begin{aligned} \operatorname{sign}(c(x) - z) &= \operatorname{sign}(c'(x, 0) - c'(0, 1)) \\ &\overset{(\star)}{=} \operatorname{sign}(\tilde{c}'(x, 0) - \tilde{c}'(0, 1)) \\ &= \operatorname{sign}(\tilde{c}(x) - \tilde{z}), \end{aligned}$$

where equality $(\star)$ follows from Lemma 8.15.

It remains to show the upper bound on the encoding length of I'. Each of the tuples $(c, z), (g_1, b_1), \ldots, (g_m, b_m)$ can be encoded with

$$\mathcal{O}\big(r^4 + r^3 d \log(ru) + rd \log(nd)\big)$$

bits. The variables d and u can be encoded with $\mathcal{O}(\log d + \log u)$ bits. In total, this yields the desired upper bound on the kernel size. ◀

This way, Theorem 8.16 extends an earlier result by Granot and Skorin-Karpov [58] who considered the restricted variant of $d = 2$.

As r is bounded from above by $\mathcal{O}((n + d)^d)$, Theorem 8.16 yields a polynomial kernel for the combined parameter (n, m, u) for constant dimensions d. In particular, Theorem 8.16 provides a polynomial kernel for Integer Linear Programming for combined parameter (n, m, u). This provides a sharp contrast to the result by Kratsch [77] that Integer Linear Programming does not admit a polynomial kernel for combined parameter (n, m) unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

9 Conclusions and Open Problems

We used smoothed analysis as an attempt to explain the good practical performance of local search for the Max-Cut problem as well as for some scheduling problems. Furthermore we derived kernels with a linear number of vertices for Max-Cut Above Edwards-Erdős Bound as well as for many generalizations. Finally we obtained polynomial kernels for several weighted optimization problems. In the following we summarize some of the remaining open problems related to our work.

9.1 Smoothed Analysis of Local Search for Max-Cut

We showed that the smoothed running time of the FLIP algorithm for the Max-Cut Problem is polynomially bounded in $n^{\log n}$ and ϕ. For this purpose we introduced the analysis of $\Theta(n)$ consecutive improvement steps, whereas former analyses only looked at a constant number – normally one – of consecutive improvement steps. However, our approach is not suitable to show polynomial bounds. This is because Angel et al. [3] proved that there are sequences of length $\Omega(n)$ in which for every subsequence of length ℓ' only $O(\ell'/\log(n))$ vertices flip twice. This means that for $k = o(\log n)$ every block sequence of linear length is k-repeating.

Instead we hope to trigger future research similar to Arthur and Vassilvitskii's paper about the k-means method [5]. They showed the non-polynomial bound $n^{O(k)}$ which inspired further research leading to a polynomial bound by Arthur et al. [4]. We also hope that local search algorithms for other PLS-hard problems can be analyzed in a similar manner, especially for problems arising from algorithmic game theory. A first step to improve our quasi-polynomial bound is the result by Angel et al. [3] that FLIP terminates with high probability after polynomially many steps on complete graphs. We conjecture that a polynomial bound holds for arbitrary instances.

A different perspective on the Max-Cut problem is given by our smoothed analysis of complete instances with squared Euclidean distances. This setting is motivated by clustering applications. Our Maximum-Cut setting corresponds to min-sum 2-clustering with the popular choice of squared Euclidean distances as distance measure. Instead of perturbing edge weights independently of each other, we perturb the positions of the vertices in a d-dimensional real vector space. This matches our intuition for geometrical and clustering applications. We proved an upper bound on the smoothed number of steps of the FLIP algorithm that is polynomial in n, σ^{-1}, and 2^d. We do not know the worst-case behavior on complete graphs, but as a contrast we showed that restricting the Max-Cut problem to squared Euclidean distances on non-complete graphs does not suffice to guarantee a sub-exponential number of steps in the worst case.

Manthey and Röglin [84] were able to extend their smoothed analysis of k-means clustering from squared Euclidean distances to arbitrary Bregman divergences. Due to the similarities to our setting, which can also be interpreted as a clustering setting, we believe that also our results should be extendible in a similar way. Of course we conjecture also in this model that the smoothed number of steps of the FLIP algorithm is polynomial also in the dimension d.

A version of the k-means method that works rather well in experiments is Hartigan's method [102]. Telgarsky and Vattani conjecture that the smoothed running time of this algorithm is polynomial [102]. However, so far this conjecture could not be proven and it seems rather challenging. As Hartigan's method has some similarities with the FLIP algorithm for the Max-Cut problem for squared Euclidean distances, we believe that our proof technique might also be helpful for proving Telgarsky and Vattani's conjecture.

9.2 Scheduling

We have shown several bounds for the convergence times of local search regarding three different coordination mechanisms on rational inputs. The choice of the right pivot rule decides in the Shortest Job First model between linear and exponential convergence times. The FIFO model is new but we believe that it is a realistic choice for many different real-life applications. We were able to show that every pivot rule converges in this model in linear time on identical machines and a large class of reasonable pivot rules converges in smoothed polynomial time on related machines. An interesting observation is that the machine speeds do not occur in any bound. We leave it as a conjecture that every pivot rule converges in polynomial time in the FIFO model. Another interesting open problem is whether the Best Improvement pivot rule in the Makespan model converges in smoothed or even deterministic polynomial time on related machines. We were only able to show that this happens with high probability when the input is perturbed.

9.3 Max-Cut Above Edwards-Erdős Bound

For the classical (Signed) Max-Cut problem, and its wide generalization to strongly λ-extendible properties, parameterized above the classical Poljak-Turzík bound, we improved the running time analysis for a known fixed-parameter algorithm to $8^k \cdot O(m)$. We further improved all known kernels with $O(k^3)$ vertices for these problems to asymptotically optimal $O(k)$ vertices. We did not try to optimize the hidden constants, as the analysis is already quite cumbersome.

A natural question to ask is whether this problem admits faster algorithms and smaller kernels, say with running time $2^k \cdot O(m)$ and $2k$ vertices respectively, or whether such results can be ruled out assuming a standard hypothesis.

It remains an interesting question whether all positive results presented here extend to edge-weighted graphs, where each edge receives a positive integer weight and the number m of edges in the Edwards-Erdős bound (1) is replaced by the total sum of the edge weights.

Further, Mnich et al. [87] showed fixed-parameter tractability of Above Poljak-Turzík(Π) for *all* strongly λ-extendible properties Π. However, the polynomial kernelization results by Crowston et al. [26] as well as our results do not seem to apply to the special case of non-hereditary $\frac{1}{2}$-extendible properties. Such properties exist; e.g., $\Pi = \{G \in \mathcal{G} \mid C \ncong K_3 \text{ for all 2-connected components } C \text{ of } G\}$. Also, for $\frac{1}{2}$-extendible properties on labelled graphs we only showed a polynomial kernel for the special case of Signed Max-Cut. It would be desirable to avoid these restrictions.

9.4 Polynomial Kernels for Weighted Problems

We obtained polynomial kernels for the Knapsack problem parameterized by the number of items. We further provide polynomial kernels for weighted versions of a number of fundamental combinatorial optimization problems, as well as integer polynomial programs with bounded range. Our small kernels are built on a seminal result by Frank and Tardos about compressing large integer weights to smaller ones. Therefore, a natural research direction to pursue is to improve the compression quality provided by the Frank-Tardos algorithm.

For the weighted problems we considered here, we obtained polynomial kernels whose sizes are generally larger by some degrees than the best known kernel sizes for the unweighted counterparts of these problems. It would be interesting to know whether this increase in kernel size as compared to unweighted problems is actually necessary (it could be that we

need more space for objects but also due to space for encoding the weights), or whether the kernel sizes of the unweighted problems can be matched.

References

1 Emile HL Aarts and Jan Karel Lenstra. *Local search in combinatorial optimization*. Princeton University Press, 1997.

2 Heiner Ackermann, Heiko Röglin, and Berthold Vöcking. On the impact of combinatorial structure on congestion games. *Journal of the ACM*, 55(6), 2008.

3 Omer Angel, Sébastien Bubeck, Yuval Peres, and Fan Wei. Local max-cut in smoothed polynomial time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 429–437. ACM, 2017.

4 David Arthur, Bodo Manthey, and Heiko Röglin. Smoothed analysis of the k-means method. *Journal of the ACM*, 58(5):19, 2011.

5 David Arthur and Sergei Vassilvitskii. Worst-case and smoothed analysis of the ICP algorithm, with an application to the k-means method. *SIAM Journal on Computing*, 39(2):766–782, 2009.

6 James Aspnes, Yossi Azar, Amos Fiat, Serge Plotkin, and Orli Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *J. ACM*, 44(3):486–504, 1997. URL: `http://dx.doi.org/10.1145/258128.258201`, `doi:10.1145/258128.258201`.

7 Baruch Awerbuch, Yossi Azar, Amir Epstein, Vahab S. Mirrokni, and Alexander Skopalik. Fast convergence to nearly optimal solutions in potential games. In *ACM Conference on Electronic Commerce (EC)*, pages 264–273, 2008.

8 Francisco Barahona. On the computational complexity of Ising spin glass models. *J. Phys. A*, 15(10):3241–3253, 1982.

9 Rene Beier and Berthold Vöcking. Random knapsack in expected polynomial time. *J. Comput. System Sci.*, 69(3):306–329, 2004. URL: `http://dx.doi.org/10.1016/j.jcss.2004.04.004`, `doi:10.1016/j.jcss.2004.04.004`.

10 Pavel Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, USA, 2002.

11 Anand Bhalgat, Tanmoy Chakraborty, and Sanjeev Khanna. Approximating pure nash equilibrium in cut, party affiliation, and satisfiability games. In *Proceedings of the 11th ACM Conference on Electronic Commerce (EC)*, pages 73–82, 2010.

12 Hans L. Bodlaender, Fedor V. Fomin, and Saket Saurabh. Open problems posed at WORKER 2010. `http://fpt.wdfiles.com/local--files/open-problems/open-problems.pdf`. Accessed: 2015-12-21.

13 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM J. Discrete Math.*, 28(1):277–305, 2014.

14 Béla Bollobás and Alexander Scott. Better bounds for Max Cut. In Béla Bollobás, editor, *Contemporary Combinatorics*, volume 10 of *Bolyai Soc. Math. Studies*, pages 185–246, 2002.

15 Peter Brucker, Johann Hurink, and Frank Werner. Models and algorithms for planning and scheduling problems improving local search heuristics for some scheduling problems. part ii. *Discrete Applied Mathematics*, 72(1):47 – 69, 1997. URL: `http://www.sciencedirect.com/science/article/pii/S0166218X96000364`, `doi:http://dx.doi.org/10.1016/S0166-218X(96)00036-4`.

16 T. Brunsch, H. Röglin, C. Rutten, and T. Vredeveld. Smoothed performance guarantees for local search. *Math. Program.*, 146(1-2, Ser. A):185–218, 2014. URL: `http://dx.doi.org/10.1007/s10107-013-0683-7`, `doi:10.1007/s10107-013-0683-7`.

17 Tobias Brunsch, Michael Etscheid, and Heiko Röglin. Bounds for the convergence time of local search in scheduling problems. In *International Conference on Web and Internet Economics*, pages 339–353. Springer, 2016.

18 C. Chiang, A. B. Kahng, S. Sinha, X. Xu, and A. Z. Zelikovsky. Fast and efficient bright-field AAPSM conflict detection and correction. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 26(1):115–126, 2007.

19 Miroslav Chlebík and Janka Chlebíková. Crown reductions for the minimum weighted vertex cover problem. *Discrete Appl. Math.*, 156(3):292–312, 2008.

20 George Christodoulou, Elias Koutsoupias, and Akash Nanavati. Coordination mechanisms. volume 410, pages 3327–3336. 2009. URL: `http://dx.doi.org/10.1016/j.tcs.2009.01.005`, `doi:10.1016/j.tcs.2009.01.005`.

21 George Christodoulou, Vahab S. Mirrokni, and Anastasios Sidiropoulos. Convergence and approximation in potential games. *Theoretical Computer Science*, 438:13–27, 2012.

22 R. Crowston, M. Fellows, G. Gutin, M. Jones, E. J. Kim, F. Rosamond, I. Z. Ruzsa, S. Thomassé, and A. Yeo. Satisfying more than half of a system of linear equations over GF(2): a multivariate approach. *J. Comput. System Sci.*, 80(4):687–696, 2014.

23 R. Crowston, G. Gutin, M. Jones, and G. Muciaccia. Maximum balanced subgraph problem parameterized above lower bound. *Theoret. Comput. Sci.*, 513:53–64, 2013.

24 Robert Crowston, Gregory Gutin, and Mark Jones. Directed acyclic subgraph problem parameterized above the Poljak-Turzík bound. In *Proc. FSTTCS 2012*, volume 18 of *Leibniz Int. Proc. Informatics*, pages 400–411, 2012.

25 Robert Crowston, Mark Jones, and Matthias Mnich. Max-cut parameterized above the Edwards-Erdős bound. *Algorithmica*, 72(3):734–757, 2015.

26 Robert Crowston, Mark Jones, Gabriele Muciaccia, Geevarghese Philip, Ashutosh Rai, and Saket Saurabh. Polynomial kernels for λ-extendible properties parameterized above the Poljak-Turzík bound. In *Proc. FSTTCS 2013*, volume 24 of *Leibniz Int. Proc. Informatics*, pages 43–54, 2013.

27 Marek Cygan, Fedor V. Fomin, Bart M. P. Jansen, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, MichałPilipczuk, and Saket Saurabh. Open problems for FPT school 2014. `http://fptschool.mimuw.edu.pl/opl.pdf`. Accessed: 2015-12-21.

28 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 3. Springer, 2015.

29 Marek Cygan, Łukasz Kowalik, and Marcin Pilipczuk. Open problems from workshop on kernels (2013). `http://worker2013.mimuw.edu.pl/slides/worker-opl.pdf`. Accessed: 2015-12-21.

30 Artur Czumaj and Berthold Vöcking. Tight bounds for worst-case equilibria. *ACM Trans. Algorithms*, 3(1):Art. 4, 17, 2007. URL: `http://dx.doi.org/10.1145/1219944.1219949`, `doi:10.1145/1219944.1219949`.

31 Frederic Dorn. Planar subgraph isomorphism revisited. In *Proc. STACS 2010*, volume 5 of *Leibniz Int. Proc. Informatics*, pages 263–274, 2010.

32 R.G. Downey and M.R. Fellows. Fixed-parameter intractability. In *Proc. 7th Annual Structure in Complexity Theory Conference*, pages 36–49, Jun 1992.

33 Rodney G Downey and Michael R Fellows. *Fundamentals of parameterized complexity*, volume 4. Springer, 2013.

34 Zdeněk Dvořák and Matthias Mnich. Large Independent Sets in Triangle-Free Planar Graphs. *SIAM J. Discrete Math.*, 31(2):1355–1373, 2017.

35 C. S. Edwards. Some extremal properties of bipartite subgraphs. *Canad. J. Math.*, 25:475–485, 1973.

36 C. S. Edwards. An improved lower bound for the number of edges in a largest bipartite subgraph. In *Recent Advances in Graph Theory*, pages 167–181, 1975.

37 Robert Elsässer and Tobias Tscheuschner. Settling the complexity of local max-cut (almost) completely. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 171–182, 2011.

38 Matthias Englert, Heiko Röglin, and Berthold Vöcking. Worst case and probabilistic analysis of the 2-Opt algorithm for the TSP. *Algorithmica*, 68(1):190–264, 2014.

39 Paul Erdős and Richard Rado. Intersection theorems for systems of sets. *J. London Math. Soc.*, 35:85–90, 1960.

40 Michael Etscheid. Performance guarantees for scheduling algorithms under perturbed machine speeds. *Discrete Appl. Math.*, 195:84–100, 2015. URL: `http://dx.doi.org/10.1016/j.dam.2014.05.041`, `doi:10.1016/j.dam.2014.05.041`.

41 Michael Etscheid, Stefan Kratsch, Matthias Mnich, and Heiko Röglin. Polynomial kernels for weighted problems. In *Mathematical foundations of computer science 2015. Part II*, volume 9235 of *Lecture Notes in Comput. Sci.*, pages 287–298. Springer, Heidelberg, 2015. URL: `https://doi.org/10.1007/978-3-662-48054-0_24`.

42 Michael Etscheid, Stefan Kratsch, Matthias Mnich, and Heiko Röglin. Polynomial kernels for weighted problems. *J. Comput. System Sci.*, 84:1–10, 2017. URL: `https://doi.org/10.1016/j.jcss.2016.06.004`.

43 Michael Etscheid and Matthias Mnich. Linear kernels and linear-time algorithms for finding large cuts. In *Proc. ISAAC 2016*, volume 64 of *Leibniz Int. Proc. Informatics*, pages 31:1–31:13, 2016.

44 Michael Etscheid and Matthias Mnich. Linear kernels and linear-time algorithms for finding large cuts. *Algorithmica*, Oct 2017. URL: `https://doi.org/10.1007/s00453-017-0388-z`, `doi:10.1007/s00453-017-0388-z`.

45 Michael Etscheid and Heiko Röglin. Smoothed analysis of local search for the maximum-cut problem. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 882–889, 2014.

46 Michael Etscheid and Heiko Röglin. Smoothed analysis of the squared euclidean maximum-cut problem. In *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA)*, pages 509–520, 2015.

47 Michael Etscheid and Heiko Röglin. Smoothed analysis of local search for the maximum-cut problem. *ACM Trans. Algorithms*, 13(2):Art. 25, 12, 2017. URL: `https://doi.org/10.1145/3011870`.

48 E. Even-Dar, A. Kesselman, and Y. Mansour. Convergence time to nash equilibria. In *Proc. of ICALP 2003*, pages 502–513. 2003. URL: `http://dx.doi.org/10.1007/3-540-45061-0_41`, `doi:10.1007/3-540-45061-0_41`.

49 Alex Fabrikant, Christos H. Papadimitriou, and Kunal Talwar. The complexity of pure nash equilibria. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 604–612, 2004.

50 Rainer Feldmann, Martin Gairing, Thomas Lücking, Burkhard Monien, and Manuel Rode. Nashification and the coordination ratio for a selfish routing game. In *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings*, pages 514–526, 2003. URL: `http://dx.doi.org/10.1007/3-540-45061-0_42`, `doi:10.1007/3-540-45061-0_42`.

51 Michael R. Fellows, Serge Gaspers, and Frances A. Rosamond. Parameterizing by the number of numbers. *Theory Comput. Syst.*, 50(4):675–693, 2012.

52 Michael R. Fellows, Jiong Guo, Dániel Marx, and Saket Saurabh. Data reduction and problem kernels (dagstuhl seminar 12241). *Dagstuhl Reports*, 2(6):26–50, 2012.

53 Greg Finn and Ellis Horowitz. A linear time approximation algorithm for multiprocessor scheduling. *BIT*, 19(3):312–320, 1979. URL: `http://dx.doi.org/10.1007/BF01930985`, `doi:10.1007/BF01930985`.

54 András Frank and Éva Tardos. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.

55 M. R. Garey and D. S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM J. Comput.*, 4(4):397–411, 1975. URL: `http://dx.doi.org/10.1137/0204035`, `doi:10.1137/0204035`.

56 Paul W. Goldberg. Bounds for the convergence rate of randomized local search in a multiplayer load-balancing game. In *Proc. of PODC 2004*, pages 131–140, 2004. `doi:10.1145/1011767.1011787`.

57 R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal, The*, 45(9):1563–1581, Nov 1966. `doi:10.1002/j.1538-7305.1966.tb01709.x`.

58 Frieda Granot and Jadranka Skorin-Kapov. On simultaneous approximation in quadratic integer programming. *Oper. Res. Lett.*, 8(5):251–255, 1989.

59 Eitan M. Gurari. *An Introduction to the Theory of Computation*. Computer Science Press, 1989.

60 Gregory Gutin and Anders Yeo. Note on maximal bisection above tight lower bound. *Inform. Process. Lett.*, 110(21):966–969, 2010.

61 Armin Haken and Michael Luby. Steepest descent can take exponential time for symmetric connection networks. *Complex Systems*, 2(2):191–196, 1988.

62 Frank Harary. On the notion of balance of a signed graph. *Michigan Math. J.*, 2:143–146 (1955), 1953–54.

63 Frank Harary. On the measurement of structural balance. *Behavioral Sci.*, 4:316–323, 1959.

64 Frank Harary, Meng-Hiot Lim, and Donald C. Wunsch. Signed graphs for portfolio analysis in risk management. *IMA J. Mgmt. Math.*, 13(3):201–210, 2002.

65 Danny Harnik and Moni Naor. On the compressibility of NP instances and cryptographic applications. *SIAM J. Comput.*, 39(5):1667–1713, 2010.

66 Falk Hüffner, Nadja Betzler, and Rolf Niedermeier. Separator-based data reduction for signed graph balancing. *J. Comb. Optim.*, 20(4):335–360, 2010.

67 C. A. J. Hurkens and T. Vredeveld. Local search for multiprocessor scheduling: how many moves does it take to a local optimum? *Oper. Res. Lett.*, 31(2):137–141, 2003. URL: `http://dx.doi.org/10.1016/S0167-6377(02)00212-2`, `doi:10.1016/S0167-6377(02)00212-2`.

68 Nicole Immorlica, Li Li, Vahab S. Mirrokni, and Andreas S. Schulz. Coordination mechanisms for selfish scheduling. *Theoret. Comput. Sci.*, 410(17):1589–1598, 2009. URL: `http://dx.doi.org/10.1016/j.tcs.2008.12.032`, `doi:10.1016/j.tcs.2008.12.032`.

69 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. System Sci.*, 63(4):512–530, 2001.

70 Yoichi Iwata, Magnus Wahlström, and Yuichi Yoshida. Half-integrality, LP-branching, and FPT algorithms. *SIAM J. Comput.*, 45(4):1377–1411, 2016.

71 Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *J. Comput. Syst. Sci.*, 79(1):39–49, 2013.

72 David S. Johnson and Lyle A. McGeoch. The traveling salesman problem: A case study in local optimization. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*. John Wiley and Sons, 1997.

73 Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.

74 Jon M. Kleinberg and Éva Tardos. *Algorithm design*. Addison-Wesley, 2006.

75 Timo Knuutila. Re-describing an algorithm by Hopcroft. *Theoret. Comput. Sci.*, 250(1-2):333–363, 2001.

76 Dénes König. Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. *Mathematische Annalen*, 77(4):453–465, 1916.

77 Stefan Kratsch. On polynomial kernels for integer linear programs: Covering, packing and feasibility. In *Proc. ESA 2013*, volume 8125 of *Lecture Notes Comput. Sci.*, pages 647–658. 2013.

78 H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983.

79 Daniel Lokshtanov, M. S. Ramanujan, and Saket Saurabh. Linear time parameterized algorithms for subset feedback vertex set. In *Proc. ICALP 2015*, volume 9134 of *Lecture Notes Comput. Sci.*, pages 935–946. 2015.

80 George S. Lueker. Exponentially small bounds on the expected optimum of the partition and subset sum problems. *Random Structures Algorithms*, 12(1):51–62, 1998. URL: `http://dx.doi.org/10.1002/(SICI)1098-2418(199801)12:1<51::AID-RSA3>3.3.CO;2-E`, `doi:10.1002/(SICI)1098-2418(199801)12:1<51::AID-RSA3>3.3.CO;2-E`.

81 Meena Mahajan and Venkatesh Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. Technical Report TR97-033, Electronic Colloquium on Computational Complexity, 1997. `http://eccc.hpi-web.de/report/1997/033/`.

82 Meena Mahajan, Venkatesh Raman, and Somnath Sikdar. Parameterizing above or below guaranteed values. *J. Comput. System Sci.*, 75(2):137–153, 2009.

83 Bodo Manthey and Heiko Röglin. Smoothed analysis: Analysis of algorithms beyond worst case. *it - Information Technology*, 53(6):280–286, 2011.

84 Bodo Manthey and Heiko Röglin. Worst-case and smoothed analysis of k-means clustering with bregman divergences. *Journal of Computational Geometry*, 4(1):94–132, 2013. URL: `http://jocg.org/index.php/jocg/article/view/39`.

85 Bodo Manthey and Rianne Veenstra. Smoothed analysis of the 2-opt heuristic for the TSP: polynomial bounds for gaussian noise. In *Proceedings of the 24th International Symposium on Algorithms and Computation (ISAAC)*, pages 579–589, 2013.

86 Dániel Marx and László A. Végh. Fixed-parameter algorithms for minimum cost edge-connectivity augmentation. In *Proc. ICALP 2013*, volume 7965 of *Lecture Notes Comput. Sci.*, pages 721–732. 2013.

87 Matthias Mnich, Geevarghese Philip, Saket Saurabh, and Ondřej Suchý. Beyond Max-Cut: λ-extendible properties parameterized above the Poljak-Turzík bound. *J. Comput. System Sci.*, 80(7):1384–1403, 2014.

88 Jesper Nederlof, Erik Jan van Leeuwen, and Ruben van der Zwaan. Reducing a target interval to a few exact queries. In *Proc. MFCS 2012*, volume 7464 of *Lecture Notes Comput. Sci.*, pages 718–727. 2012.

89 N. V. Ngdoc and Zsolt Tuza. Linear-time approximation algorithms for the max cut problem. *Combin. Probab. Comput.*, 2(2):201–210, 1993.

90 Svatopluk Poljak and Daniel Turzík. A polynomial time heuristic for certain subgraph optimization problems with guaranteed worst case bound. *Discrete Math.*, 58(1):99–104, 1986.

91 Svatopluk Poljak and Zsolt Tuza. Maximum cuts and large bipartite subgraphs. In *Combinatorial optimization (New Brunswick, NJ, 1992–1993)*, volume 20 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 181–244. 1995.

92 Venkatesh Raman and Saket Saurabh. Parameterized algorithms for feedback set problems and their duals in tournaments. *Theoret. Comput. Sci.*, 351(3):446–458, 2006.

93 Venkatesh Raman and Saket Saurabh. Improved fixed parameter tractable algorithms for two "edge" problems: MAXCUT and MAXDAG. *Inform. Process. Lett.*, 104(2):65–72, 2007.

94 Heiko Röglin. *The Complexity of Nash Equilibria, Local Optima, and Pareto-Optimal Solutions.* PhD thesis, RWTH Aachen University, 2008.

95 Robert W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *Internat. J. Game Theory*, 2:65–67, 1973. URL: `http://dx.doi.org/10.1007/BF01737559`, `doi:10.1007/BF01737559`.

96 Arvind Sankar, Daniel A. Spielman, and Shang-Hua Teng. Smoothed analysis of the condition numbers and growth factors of matrices. *SIAM Journal on Matrix Analysis Applications*, 28(2):446–476, 2006.

97 Alejandro A. Schäffer and Mihalis Yannakakis. Simple local search problems that are hard to solve. *SIAM Journal on Computing*, 20(1):56–87, 1991.

98 Leonard J. Schulman. Clustering for edge-cost minimization. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 547–555, 2000.

99 Petra Schuurman and Tjark Vredeveld. Performance guarantees of local search for multiprocessor scheduling. *INFORMS J. Comput.*, 19(1):52–63, 2007. URL: `http://dx.doi.org/10.1287/ijoc.1050.0152`, `doi:10.1287/ijoc.1050.0152`.

100 Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.

101 Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis: An attempt to explain the behavior of algorithms in practice. *Communications of the ACM*, 52(10):76–84, 2009.

102 Matus Telgarsky and Andrea Vattani. Hartigan's method: k-means clustering without voronoi. In *Proceedings of the 13th*, pages 820–827, 2010.

103 René van Bevern. *Fixed-Parameter Linear-Time Algorithms for* NP*-hard Graph and Hypergraph Problems Arising in Industrial Applications*. PhD thesis, TU Berlin, 2014.

104 Andrea Vattani. k-means requires exponentially many iterations even in the plane. *Discrete and Computational Geometry*, 45(4):596–616, 2011.