Constrained Clustering Problems and Parity Games

Clemens Rösner

geboren in Ulm

Dissertation

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

Bonn 2019

1. Gutachter: Prof. Dr. Heiko Röglin
2. Gutachterin: Prof. Dr. Anne Driemel

Tag der mündlichen Prüfung: 05. September 2019
Erscheinungsjahr: 2019

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Abstract

Clustering is a fundamental tool in data mining. It partitions points into groups (clusters) and may be used to make decisions for each point based on its group. We study several clustering objectives. We begin with studying the Euclidean k-center problem. The k-center problem is a classical combinatorial optimization problem which asks to select k centers and assign each input point in a set P to one of the centers, such that the maximum distance of any input point to its assigned center is minimized. The Euclidean k-center problem assumes that the input set P is a subset of a Euclidean space and that each location in the Euclidean space can be chosen as a center. We focus on the special case with $k = 1$, the smallest enclosing ball problem: given a set of points in m-dimensional Euclidean space, find the smallest sphere enclosing all the points. We combine known results about convex optimization with structural properties of the smallest enclosing ball to create a new algorithm. We show that on instances with rational coefficients our new algorithm computes the exact center of the optimal solutions and has a worst-case run time that is polynomial in the size of the input. We use the new algorithm to show that we can solve the Euclidean k-center problem in polynomial time for constant k and dimension m.

The general unconstrained clustering problems are mostly very well studied. The k-center problem for example allows for elegant 2-approximation algorithms [47, 53]. However, the situation becomes significantly more difficult when constraints are added to the problem. We first look at the fair clustering. The fairness constraint is motivated by the fact that the general process of computing a clustering may harm protected (minority) classes if the clustering algorithm does not adequately represent them in desirable clusters – especially if the data is already biased.

At NIPS 2017, Chierichetti et al. [29] proposed a model for *fair clustering* requiring the representation in each cluster to (approximately) preserve the global fraction of each protected class. Restricting to two protected classes, they developed both a 4-approximation algorithm for the fair k-center problem and an $\mathcal{O}(t)$-approximation algorithm for the fair k-median problem, where t is a parameter for the fairness model. For multiple protected classes, the best known result is a 14-approximation algorithm for fair k-center [90].

We extend and improve the known results. Firstly, we give a 5-approximation algorithm for the fair k-center problem with multiple protected classes. Secondly, we propose a relaxed fairness notion under which we can give bicriteria constant-factor approximation algorithms for the fair version of all of the classical clustering objectives (k-center, k-supplier, k-median, k-means and facility location). The latter approximation algorithms are achieved by a framework that takes an arbitrary existing unfair (integral) solution and a fair (fractional) LP solution and combines them into an essentially fair clustering with a weakly supervised rounding scheme. In this way, a fair clustering can be established belatedly, in a situation where for example the centers are already fixed.

The second clustering constraint we study is privacy: Here, we are asked to only open a center when at least ℓ points will be assigned to it. We raise the question whether a general method can be derived to turn an approximation algorithm for a clustering problem with some constraints into an approximation algorithm that additionally respects privacy. We show how to combine privacy with several other constraints and obtain approximation algorithms for the k-center problem with several combinations of constraints.

In this dissertation we also study parity games, a two player game played on a directed graph. We study the case in which one of the two players controls only a small number k of nodes and the other player controls the $n-k$ other nodes of the game. Our main result is a fixed-parameter-tractable algorithm that solves bipartite parity games in time $k^{O(\sqrt{k})} \cdot O(n^3)$, and general parity games in time $(p+k)^{O(\sqrt{k})} \cdot O(pnm)$, where p is the number of distinct priorities and m is the number of edges. For all games with $k = o(n)$ this improves the previously fastest algorithm by Jurdziński, Paterson, and Zwick [64].

We also obtain novel kernelization results and an improved deterministic algorithm for parity games on graphs with small average node-degree.

Zusammenfassung

Clustering ist ein grundlegendes Werkzeug im Data Mining. Es unterteilt Punkte in Gruppen (Cluster) und kann verwendet werden, um Entscheidungen für jeden Punkt basierend auf seiner Gruppe zu treffen. Wir untersuchen verschiedene Clustering Modelle. Als erstes untersuchen wir das euklidische k-Center Problem. Das k-Center Problem ist ein klassisches kombinatorisches Optimierungsproblem, bei dem wir k Zentren auswählen und jeden Punkt aus der Eingabemenge P einem der Zentren zuweisen sollen, sodass die maximale Entfernung eines Punktes zu seinem zugewiesenen Zentrum minimiert wird. Das euklidische k-Center Problem geht davon aus, dass die Eingabemenge P eine Teilmenge eines euklidischen Raums ist und dass jeder mögliche Ort im euklidischen Raum als Zentrum gewählt werden darf.

Wir konzentrieren uns auf den Sonderfall mit $k = 1$, das Smallest Enclosing Ball Problem: Gegeben sei eine Menge von Punkten im m-dimensionalen euklidischen Raum, finde die kleinste Kugel, die alle diese Punkte enthält. Wir kombinieren bekannte Ergebnisse der konvexen Optimierung mit strukturellen Eigenschaften des Smallest Enclosing Balls, um einen neuen Algorithmus zu entwerfen. Wir zeigen, dass unser neuer Algorithmus bei Instanzen mit rationalen Koeffizienten das exakte Zentrum der optimalen Lösungen berechnet und eine Worst-Case-Laufzeit aufweist, die polynomiell in der Eingabegröße ist. Wir verwenden den neuen Algorithmus, um zu zeigen, dass wir das euklidische k-Center Problem in Polynomialzeit für konstante k und Dimension m lösen können.

Ohne Nebenbedingungen sind die allgemeinen Clustering-Probleme zumeist sehr gut untersucht. Für das k-Center Problem gibt es beispielsweise elegante 2-Approximationsalgorithmen [47, 53]. Die Situation wird jedoch erheblich schwieriger, wenn wir fordern, dass zusätzliche Nebenbedingungen erfüllt werden. Als erstes betrachten wir faires Clustering. Die Fairness-Nebenbedingung wird durch die Tatsache motiviert, dass die allgemeine Berechnung guter Clusterings geschützte (Minderheits-) Klassen schädigen kann, wenn das Ergebnis sie nicht in den bevorzugten Clustern angemessen repräsentiert, insbesondere wenn die Daten bereits voreingenommen sind.

Auf der NIPS 2017 haben Chierichetti et al. [29] ein Modell für *faires Clustering* vorgeschlagen, das einfordert, dass die Repräsentation jedes Clusters (ungefähr) den globalen Anteil jeder geschützten Klasse widerspiegelt. Sie beschränkten sich auf zwei geschützte Klassen und entwickelten sowohl einen 4-Approximationsalgorithmus für das faire k-Center Problem als auch einen $\mathcal{O}(t)$-Approximationsalgorithmus für das faire k-Median Problem, wobei t ein Parameter des Fairness-Modells ist. Bei mehreren geschützten Klassen ist das beste bekannte Ergebnis ein 14-Approximationsalgorithmus für das faire k-Center Problem [90].

Wir erweitern und verbessern die bekannten Ergebnisse. Zum einen geben wir einen 5-Approximationsalgorithmus für das faire k-Center Problem mit mehreren geschützten Klassen an. Zum anderen schlagen wir eine abgeschwächte Fairness Bedingung vor, unter der wir für die faire Version von allen klassischen Clustering-Problemen (k-Center, k-Supplier, k-Median, k-Means und Facility Location) einen bikriteriellen Approximationsalgorithmus mit konstantem Approximationsfaktor angeben können. Die letzteren Approximationsalgorithmen werden durch ein Rahmenwerk erreicht, das eine beliebige gegebene unfaire (ganzzahlige) Lösung und eine faire (fraktionelle) LP-Lösung verwendet und diese mittels eines schwach überwachten Rundungsschema zu einem im Wesentlichen fairen Clustering kombiniert. Auf diese Weise kann Fairness nachträglich zu einem Clustering, bei dem zum Beispiel die Zentren bereits festgelegt sind, hinzugefügt werden.

Die zweite untersuchte Nebenbedingung für Clustering-Probleme ist Privacy: Hier werden wir aufgefordert, ein Zentrum nur dann zu öffnen, wenn ihm mindestens ℓ Punkte zugewiesen werden. Wir stellen die Frage, ob es eine allgemeine Methode gibt, um einen Approximationsalgorithmus für ein Clustering Problem mit einigen Nebenbedingungen in einen Approximationsalgorithmus umzuwandeln, der zusätzlich Privacy erfüllt. Wir zeigen, wie man Privacy mit mehreren anderen Nebenbedingungen kombinieren kann und erhalten Approximationsalgorithmen für das k-Center Problem mit mehreren Kombinationen von Nebenbedingungen.

In dieser Dissertation untersuchen wir auch Paritätsspiele, ein 2-Spieler-Spiel auf einem gerichteten Graph. Wir untersuchen die eingeschränkte Variante, bei der einer der beiden Spieler nur eine kleine Anzahl k der Knoten kontrolliert und der andere Spieler die $n - k$ restlichen Knoten des Spiels kontrolliert. Unser Hauptergebnis ist ein Fest-Parameter-Algorithmus, der bipartite Paritätsspiele in $k^{O(\sqrt{k})} \cdot O(n^3)$ Zeit und allgemeine Paritätsspiele in $(p + k)^{O(\sqrt{k})} \cdot O(pnm)$ Zeit löst, hierbei bezeichnet p die Anzahl der unterschiedlichen Prioritäten und m die Anzahl der Kanten. Für alle Spiele mit $k = o(n)$ verbessert dies den zuvor schnellsten Algorithmus von Jurdziński, Paterson und Zwick [64].

Wir erhalten auch ein neuartiges Kernelisierungsergebnis und einen verbesserten deterministischen Algorithmus für Paritätsspiele in Graphen mit geringem durchschnittlichen Knotengrad.

Acknowledgments

I would like to express my gratitude towards several people who made the last couple of years an exciting time, supported me, and encouraged me with the research.

First and foremost, I thank my advisor, Heiko Röglin, without him it would not have been possible for me to write this thesis. I am very grateful to Heiko for his support, guidance, and all of his invaluable ideas. Being part of his research group and all of the interesting discussions within, have been an unforgettable experience.

Besides Heiko, I also thank all of my other coauthors Ioana Bercea, Martin Groß, Samir Khuller, Aounon Kumar, Matthias Mnich, Daniel Schmidt, and especially Melanie Schmidt for their valuable contributions. In particular, I am grateful to Melanie, not only because of her essential input to large parts of my research, but also for our interesting discussions about non research related topics.

I thank all of my colleagues in the department of theoretical computer science at the university of Bonn. Special thanks go to Michael Etscheid and Carsten Fischer who always had an open ear for my problems and were part of a lot of interesting discussions.

Last but not least, I thank my family for their support and encouragement.

Contents

1 Introduction **1**

1.1 Approximation Algorithms and Fixed Parameter Tractability 3

1.2 Clustering 4

1.2.1 The Euclidean k-Center Problem 6

1.2.2 Fair Clustering 8

1.2.3 Privacy Preserving Clustering 12

1.2.4 Preliminaries 13

1.3 Parity Games 16

1.3.1 Our contributions 17

1.3.2 Detailed comparison with previous work 18

1.4 Outline and Bibliographical Notes 19

2 Clustering **20**

2.1 The Euclidean k-Center Problem 20

2.1.1 The Smallest Enclosing Ball Problem 22

2.2 Fair Clustering 31

2.2.1 Computing Fair Clusterings via Fairlets 32

2.2.2 (I)LP Formulations for Fair Clustering Problems 44

2.2.3 Black-Box Algorithms for Essentially Fair Clustering 46

2.2.4 True Approximation Algorithms for Fair k-Center and k-Supplier 51

2.2.5 NP-Hardness of the Fair Assignment Problem for k-Center 55

2.2.6 Integrality Gap of the Canonical Clustering LP 56

2.3 Privacy Preserving Clustering 58

2.3.1 The Private k-Center Problem with Outliers 59
2.3.2 Combining Privacy with other Constraints 67
2.3.3 Strongly Private k-Center 75
2.3.4 (Metric) Facility Location 78

3 Parity Games 81

3.1 Fundamental Properties and Notation of Parity Games 81
3.2 Kernelization of Parity Games 84
3.2.1 General Parity Games 84
3.2.2 Bipartite Parity Games 86
3.3 A Simple Exponential-Time Algorithm 90
3.4 Overview of the New Algorithms 90
3.5 Finding Small Dominions 91
3.6 New Algorithms for Solving Parity Games 94
3.7 Out-Degree based Algorithm 94
3.8 Pseudocode for Algorithms **new-win** 96
3.9 Analysis of the Run Time 98
3.9.1 Recurrence Relation Computations 100

4 Conclusion and Open Problems 105

4.1 Clustering 105
4.1.1 The Euclidean k-Center Problem 105
4.1.2 Fair Clustering 106
4.1.3 Private Clustering 106
4.2 Parity Games 107

A Approximation Algorithm for the k-Center Problem [47] 108

B Facts about the k-Means Cost Function 110

C Fair k-Center Analysis by Chierichetti et al. [29] 111

D Notation 113

Bibliography 115

Index 125

List of Figures

2.1 Example of a fairlet decomposition 32
2.2 Flow Network for rounding the x-variables 51
2.3 Reduction from Exact Cover by 3-sets to the fair assignment problem . 55
2.4 Integrality gap example for the fair clustering LP 57

C.1 Example for the fairlet decomposition analysis 112

List of Tables

1.1 Essentially fair approximation ratio 11
1.2 Overview of constrained clustering approximation results 16

D.2 Objects relevant for our Parity Games results 113
D.1 Objects relevant for our Clustering results 114

Chapter 1

Introduction

A pivotal part of theoretical computer science is the development and analysis of algorithms for different computational tasks. We categorize computational tasks into three categories: *decision problems*, *search problems*, and *optimization problems*. Decision problems are all problems that can be stated as a yes-no question. Problems that require us to find one (of possibly multiple) feasible solutions for the input are called search problems. Similar to search problems an optimization problem asks to find a feasible solution for the input. In addition we assume that each possible feasible solution has some value associated with it and the task is to find a feasible solution with minimal (or maximal) value among all feasible solutions.

For a given computational task we would like to develop an algorithm which solves the problem and is preferably fast. Here we say that an algorithm is fast, if it always finishes in polynomial time, depending on the size of the input. Unfortunately for many problems it is impossible or unknown, if they can be solved in polynomial time. To deal with such cases there are several different approaches, including approximation algorithms for optimization problems and parameterized algorithms. For optimization problems we can relax the optimality requirement of the solution, as, instead of taking a long time to find a solution with the best value, it can, in some cases, be better to compute a solution with a good value in only a short amount of time. An approximation algorithm for an optimization problem is exactly that, it is an algorithm that takes polynomial time and computes a feasible solution, whose value is, in some sense, close to the value of the optimal solution. Parameterized algorithms on the other hand are not restricted to optimization problems. They are algorithms that still compute a solution as originally required and might take a long time for some input instances. Yet the run time of a parameterized algorithm does not only depend on the input size, but also on some additional parameter of the input, and, in case this parameter is small, the run time of the algorithm should be polynomial in the input size.

In this dissertation we take a look at different computational tasks for which we develop and analyze polynomial time algorithms, approximation algorithms and parameterized algorithms.

The computational tasks in this dissertation come from two different problem concepts.

In the first part of this dissertation we deal with several clustering problems. A clustering problem asks us to partition a set of objects into clusters, such that the objects in each cluster are similar to each other. This is a very common task in many different fields, which include machine learning, bio-informatics, and pattern recognition. Computation of a clustering for a given set of objects has two major aspects: How to quantify the similarity (or difference or distance) of two objects and how to judge the quality of the partitioning of the original set into subsets, i.e., decide how to compute the value of a clustering.

We will not deal with quantifying the similarity between objects, as it depends on the specific type of objects and the context, with respect to which they are supposed to be clustered. We will assume that the similarity of two objects is known and can be quantified by a distance function on the set of objects. Given such a function, that describes the similarity between pairs of objects, there are numerous ways to judge the quality of a clustering, including, in particular, *k-center/k-supplier*, *k-median*, *k-means*, and *facility location*. They all specify the clustering problem as a minimization problem and either restrict the number of clusters or impose an additional cost for each cluster.

Specific clustering problems often require that the computed solution satisfies additional constraints. We start with the *fairness* constraint. Requiring fairness in a clustering can help to reduce a possible bias in the distance function or ensure that each cluster has a beneficial composition of different types of objects. We study fair clustering with respect to k-center/k-supplier, k-median, k-means, and facility location. Our results include true and bicriteria approximation algorithms as well as hardness results. We further consider *private* clustering, where each cluster must contain at least a specified number of points. We study the private k-center/k-supplier problem and try to combine privacy with other constraints, like *outliers*, *capacities*, and fairness. We take an existing approximation algorithm for one of the other constraints and show how we can use it and obtain an approximation algorithm, which satisfies privacy as well as the other constraint.

In the second part of the dissertation we study *parity games*, a two player game of perfect information played on a directed graph. Solving parity games has various applications in, for example, the theory of formal languages and automata. Among others the *model-checking* problem for the *modal μ-calculus* can be reduced to solving parity games. We study parity games with respect to several parameters. These parameters include the minimum number of nodes owned by one of the players, the number of different priorities, as well as several parameters based on the out-degrees of the nodes in the graph. Our first result looks at the combination of two parameters, the number of different priorities and the minimum number of nodes owned by one of the players. We obtain a kernelization result and a fixed-parameter-tractable parity games solver for the combined parameter. For the out-degree based parameters, we obtain a parity games solver, whose run time is polynomial if the parameter is bounded by a fixed constant.

1.1 Approximation Algorithms and Fixed Parameter Tractability

Before we give a more detailed introduction to clustering and parity games, we give a small excursion into different types of algorithms.

When developing algorithms for a computational task, we prefer algorithms with a small run time. An algorithm is commonly considered fast, if its worst-case run time is polynomial in the size of the input. Unfortunately there are a lot of computational tasks, for which probably no algorithms with worst-case run time polynomial in the input size exist.

There are several ways to deal with the fact that it might be impossible for a problem to be solved in polynomial time. We will describe two concepts, which can help us to create algorithms which are still relatively fast and find relatively good solutions. First we take a look at approximations and after that we look at parameterized run times.

Given that the computational task at hand is an optimization problem we can, instead of trying to find an optimal feasible solution, try to find any feasible solution, whose value is close to the best possible value.

Definition 1. *Given an optimization problem $\mathcal{P}$ and an instance I of $\mathcal{P}$, let* $\mathsf{opt}(I)$ *denote the value of an optimal feasible solution for I. We say that, for $\alpha \geq 1$, a feasible solution S is an* α-approximation *for I, if $val(S)$, the value of S, is within a factor α to* $\mathsf{opt}(I)$*, i.e., if $\mathcal{P}$ is a minimization problem, we must have $val(S) \leq \alpha \cdot \mathsf{opt}(I)$, and, if $\mathcal{P}$ is a maximization problem, we must have $val(S) \geq \frac{\mathsf{opt}(I)}{\alpha}$.*

Definition 2. *Given an optimization problem $\mathcal{P}$ and $\alpha \geq 1$, an algorithm A is considered an α-*approximation algorithm *for $\mathcal{P}$, if for any given instance I, A returns an α-approximation for I in time polynomial in the size of I. We say that algorithm A has an* approximation ratio *of α.*

In some cases it even seems hard to find an α-approximation algorithm for a problem $\mathcal{P}$ and a reasonably small α. In such cases we might be able to try to relax the definition of what we consider a feasible solution and also consider almost feasible solutions. In that case we also need to extend the value function to almost feasible solutions. How such a relaxation works depends on the specific problem. We call an algorithm that returns, for any instance I, a feasible or almost feasible solution, whose value is within a factor α of $opt(I)$, a bicriteria α-approximation algorithm.

Another approach is to develop an algorithm that does not always have a run time polynomial in the input size, but is fast, given that certain circumstances are met. One way to use such an approach is to analyze the run time, not only with respect to the input size, but also with respect to some other parameters of the instance.

Definition 3. *A* parameterized problem *is a tuple $(\mathcal{P}, k)$. The first element of a parameterized problem is a computational task $\mathcal{P}$. Let Σ^* denote the set of feasible input instances for $\mathcal{P}$, then the second element of the parameterized problem is a parameter function $k : \Sigma^* \to \mathbb{N}$, that assigns a parameter value to every instance of the problem $\mathcal{P}$.*

What kind of parameters are reasonable and useful will depend on the problem, a parameter for problems on a graph could for example be the maximum or average node degree. Given a parameterized problem $(\mathcal{P}, k)$ and an algorithm A for $(\mathcal{P}, k)$ we analyze the worst-case run time of A with respect to both, the input size and the parameter of the input instance, i.e., we want to show that for every feasible instance I for $\mathcal{P}$ A has a run time of $O(f(size(I), k(I)))$, where $size(I)$ denotes the input size of I. When $f(size(I), k(I))$ is a polynomial in case $k(I)$ is bounded by a fixed constant, we have found an algorithm which is fast on all instances with a small parameter. A special class of such algorithms are the fixed-parameter-tractable algorithms whose run time depends only polynomial on the size of the input and may depend super polynomially only on the value of the parameter.

Definition 4. *A parameterized problem $(\mathcal{P}, k)$ is called* fixed-parameter-tractable*, if there exists an algorithm A, that solves any instance I of $\mathcal{P}$ in time $O(f(k(I)) \cdot g(size(I)))$, where $f : \mathbb{N} \to \mathbb{N}$ is an arbitrary function depending only on $k(I)$ and $g : \mathbb{R} \to \mathbb{R}$ is a polynomial function. We call A a* fixed-parameter-tractable algorithm *for $(\mathcal{P}, k)$.*

1.2 Clustering

Clustering problems deal with the following fundamental task in, among others, unsupervised learning: Given a set of objects, partition them into clusters, such that objects within a cluster are well matched, while objects from different clusters have something that clearly differentiates them.

The classical clustering objectives studied in combinatorial optimization are *k-center/k-supplier*, *k-median*, *k-means* and *facility location*. Given a point set P, k-center/k-supplier, k-median and k-means ask for a set of k centers and an assignment of the points in P to the selected centers that minimize an objective. For k-center/k-supplier, the objective is the maximum distance of any point to its assigned center. The difference between k-center and k-supplier is that k-supplier distinguishes between the set of points P and the set of possible center locations L, while classical k-center assumes that $P = L$. For k-median, the objective is the sum of the distances of all points to their assigned center (this is called connection cost) and for k-means, it is the sum of the squared distances of all points to their assigned center. Facility location does not restrict the number of centers. Instead, every center (here called facility) has an opening cost. The goal is to find a set of centers such that the connection cost plus the opening cost of all chosen facilities is minimized. In the general version it is an NP-hard problem to find the best solution for any of these clustering problems. For the unconstrained versions it is easy to find, for a given set of centers, an optimal assignment of the points to the centers, as each point can be assigned to its closest opened center.

With the addition of constraints simply assigning every point to its closest center will often not be a feasible assignment and a different assignment is necessary in order to satisfy the additional constraints. For several constraints, including capacities, lower bounds, and outliers, an optimal feasible assignment to a given set of centers can be computed through a network flow computation. For the fairness constraint however, we will show that it is NP-hard to compute an optimal assignment to a given set of centers. A lot of research has been devoted to developing approximation algorithms for these objectives. The earliest success story is that of k-center: Gonzalez [47] as well as Hochbaum and Shmoys [53] gave a 2-approximation algorithm for the problem, while Hsu and Nemhauser [54] showed that finding a better approximation is NP-hard. The k-supplier problem can be 3-approximated [53], which is also tight [53].

Since then, much effort has been made to approximate the other objectives. Typically, facility location will be first, and transferring new techniques to k-median poses additional challenges. Significant techniques developed over the last decades are LP rounding techniques [25, 93], greedy and primal dual methods [58, 59], local search algorithms [10, 69], and, more recently, the use of pseudo-approximation [79]. The currently best approximation ratio for facility location is 1.488 [76], while the best lower bound is 1.463 [50]. For k-median, the currently best approximation algorithm achieves a ratio of $2.675+\varepsilon$ [22], while the best lower bound is $1 + \frac{2}{e} \approx 1.736$ [58]. A recent breakthrough gives a 6.357-approximation for k-means [5], but the newest hardness result is marginally above 1 [11, 72].

While the basic approximability of the objectives is well studied, a lot less is known once constraints are added to the picture. Constraints come naturally with many applications of clustering, and since machine learning and unsupervised learning methods become more and more popular, there is an increasing interest in this research topic. One of the troubles with approximation algorithms is that often they cannot easily be adapted to a different scenario. This task seems to be easier for simple heuristics for the problem, which are easier to understand and implement in the first place. Indeed, it turns out that adding constraints to clustering often requires fundamentally different techniques for the design of approximation algorithms and is a new challenge altogether.

A good example is the *capacity* constraint: Each center c is now equipped with a capacity $u(c)$, and can only serve $u(c)$ points, i.e., a feasible clustering must assign no more than $u(c)$ points to c. This natural constraint is notoriously difficult to cope with; indeed, the standard LP formulations for the problems have an unbounded integrality gap [30, 73, 9, 78]. Capacitated k-center was first approximated with uniform upper bounds [14, 66]. Local search provides a way out for facility location, leading to 3- and 5-approximation algorithms for uniform [2] and non-uniform capacities [13], and preprocessing together with involved rounding proved sufficient for k-center to obtain a 9-approximation [30, 8]. The algorithm in [8] can also be applied to the k-supplier problem and yields an 11-approximation. However, the choice of techniques that turned out to work for capacitated clustering problems is still very limited, and indeed *no* constant-factor approximation algorithms are known to date for k-median and k-means.

And all the while, new constraints for clustering problems are proposed and studied. The k-center problem allows for constant-factor approximation algorithms for many useful constraints. In *private* clustering [3], we demand a lower bound on the number of points assigned to a center. As stated in [4, 3] this ensures a certain anonymity and is motivated through the need to obtain data privacy. The more general form where each cluster has an individual lower bound is called clustering *with lower bounds* [7]. *Fair* clustering [29] assumes that points have a protected feature (like gender), modeled by a color, and that we want clusters to be fair in the sense that the ratios between points of different colors are similar in every cluster. Clustering *with outliers* [26, 31, 24] allows us to ignore a fixed number of objects and searches for a solution where a prespecified number of points may be excluded from the cost computation. Other constraints include diversity [74], fault tolerance [65], matroid or knapsack constraints [27], must-link and cannot-link constraints [98], diversity [74] and chromatic clustering constraints [33, 34].

Facility location also allows for constant-factor approximation algorithms with capacities [2, 9, 13], uniform lower bounds [6, 95], and outliers [26]. Much less is known for k-median and k-means. True constant-factor approximation algorithms so far exist only for the outlier constraint [28, 70]. A major problem for obtaining constant-factor approximation algorithms is that the natural LP has an unbounded integrality gap, which is mostly also true for the LP with additional constraints. Bicriteria approximation algorithms are known that either violate the capacity constraints [75, 77, 78] or the cardinality constraint [1].

Relatively little is known about approximation algorithms for the combination of constraints. Cygan and Kociumaka [31] give a 25-approximation algorithm for the capacitated k-center problem with outliers. Aggarwal et al. [3] give a 4-approximation algorithm for the private k-center problem with outliers. Ahmadian and Swamy [7] consider the combination of k-supplier with outliers with (non-uniform) lower bounds and derive a 5-approximation algorithm. The paper also studies the k-supplier problem with outliers (without lower bounds), and the min-sum-of-radii problem with lower bounds and outliers. Their algorithms are based on the Lagrangian multiplier preserving primal dual method due to Jain and Vazirani [59].

Ding et al. [32] study the combination of capacities and lower bounds as well as capacities, lower bounds and outliers by generalizing the LP algorithms from [8] and [31] to handle lower bounds. They give results for several variations, including a 6-approximation algorithm for private capacitated k-center and a 9-approximation algorithm for private capacitated k-supplier.

Friggstad, Rezapour, and Salavatipour [42] consider the combination of uniform capacities and non-uniform lower bounds for *facility location* and obtain bicriteria approximation algorithms.

1.2.1 The Euclidean k-Center Problem

The Euclidean k-center problem assumes that the set of points is, for some $m \in \mathbb{N}$, given as a subset P of $\mathbb{R}^m$, the m-dimensional Euclidean space. In contrast to the classical k-center and k-supplier problems, the set of possible center locations is not restricted to a finite set L, but instead each point $j \in \mathbb{R}^m$ can be chosen as a center.

With the infinite set of possible center locations, computing the optimal solution, even with exponential time, is not a trivial task. The difficulty seams to strongly depend on the dimension m. In the one dimensional case the problem is easy and can be solved in $O(n \log n)$ time [82], but for all $m \geq 2$ it is an NP-hard problem [84] to compute an optimal solution. For $m = 2$ Drezner showed that it is possible to compute an optimal solution in time $O(n^{2k+1} \log n)$ [35]. Hwang, Lee and Chang [55] improved the run time to $O(n^{O(\sqrt{k})})$.

As the general problem is difficult to solve we also look at the special case with $k = 1$.

The Euclidean 1-center Problem, is also known as the *smallest enclosing ball* (SEB) problem or the minimum bounding sphere problem, as it reduces the k-center problem to: Given a set of points P in m-dimensional Euclidean space, compute a ball of minimum radius, which contains P and output its center.

The smallest enclosing ball problem

The SEB problem is a well-studied problem with applications in numerous clustering problems [12, 15, 21] and nearest neighbor search [46].

There are algorithms for the SEB problem which for a fixed constant dimension run in linear time [83, 40, 80]. For moderately high dimensions there exist exact algorithms with good behavior in practice [41, 44, 45] as well as a $(1+\varepsilon)$-approximation algorithm [12, 71] which needs a polynomial number of arithmetic operations.

The first exact algorithm is due to Megiddo [83]. Megiddo's algorithm runs in time polynomial in the number of points, but exponential in the dimension.

Welzl [99] proposed a simpler randomized algorithm based on the extension of Seidel's algorithm [92] for linear programming. Welzl's algorithm has an expected subexponential run time. However, implementations based on Welzl's algorithm can only reasonably handle point sets with small dimension m [44].

The algorithm by Gärtner and Schönherr [45] is based on quadratic programming; in practice it runs in time polynomial in m. However, it critically requires arbitrary-precision linear algebra to avoid robustness issues, which limits the tractable dimensions.

In contrast, the algorithm by Fischer et al. [41] is a combinatorial algorithm. Their algorithm does not have polynomial run time in the worst-case, but seems to perform well in practice in moderately low dimensions and does not exhibit numerical stability problems.

Badoiu et al. present an $(1+\varepsilon)$-approximation algorithm [12], which has been improved by Kumar et al. [71]. These algorithms both use core sets and have a run time polynomial in n, m and $\frac{1}{\varepsilon}$.

As of now, no exact algorithm with polynomial run time is known for the SEB problem.

Our results

We describe the SEB problem as an optimization problem over a convex body and use known results about optimization over convex bodies [49] to show that we can obtain a $(1+\varepsilon)$-approximation in time and space polynomial in the encoding size of the instance and $\log \frac{1}{\varepsilon}$. We then show structural properties of the SEB problem which, given a good enough approximation, allow us to compute the exact center of the optimal solution. Combining both results we show that, given all coefficients of the instance are rational numbers, we can find the optimal solutions in time and space polynomial in the input size.

Theorem 5. *Instances of the SEB problem with rational coefficients can be solved in time polynomial in the input size.*

In the analysis we focus on instances with integral coefficients. Thereafter, we show how to transform instances with rational coefficients into equivalent integral instances with only a polynomial increase of the instance size.

Our focus is to show structural properties which instances of the SEB problem with integral coefficients have. We show an upper bound on how far away the center of a $(1+\varepsilon)$-approximation can be from the center of the optimal solution. In addition we show that on instances with integral coefficients each point in the input set either lies on the boundary of the optimal solution or has a significant distance to it. With the right choice of $\varepsilon > 0$ we combine these two properties. Then we can, given the distance between the approximated center and a point j of the input set, decide if j lies on the boundary of the optimal solution. Knowing the set of points on the boundary it is easy to compute the exact center of the optimal solution.

Given that we can compute the optimal center for a fixed set of points, we can solve the Euclidean k-center problem by finding the optimal partition of the points into clusters and computing the optimal center for each of the partitions.

We show how to compute a set of $O(n^{mk})$ partitionings, which contains an optimal partitioning, where n is the number of points and m is the dimension. If we test all of these partitionings and compute the optimal center for each partition we can solve the Euclidean k-center problem. If k and the dimension m are constants this only takes polynomial time.

Theorem 6. *An optimal Euclidean k-center solution of a rational instance $P \subseteq \mathbb{Q}^m$ can be computed in $O(n^{(m+1)k} poly(enc))$ time, where $n = |P|$ and enc denotes the input size of the instance.*

1.2.2 Fair Clustering

Suppose we are to reorganize school assignments in a big city. Given a long list of children starting school next year and a short list of all available teachers, the goal is to assign the students-to-be to (public) schools such that the maximum distance to the school is small. The school capacity is given by the number of its teachers: For each teacher, s students can be admitted.

This challenge is in fact an instance of the capacitated (metric) k-center problem. A naïve solution may, however, result in some schools having an excess of boys while others might have a surplus of girls. We would prefer an assignment where the classes are more balanced. Thus a new challenge arises: Assign the children such that the ratio is (approximately) 1:1 between boys and girls, and minimize the maximum distance under this condition.[1] This can be modeled by the following combinatorial optimization problem: Given a point set, half of the points are red, the other half is blue. Compute a clustering where each cluster has an equal number of red and blue points, and minimize the maximum radius.

In this form, our example is a special case of the *fair k-center* problem, as proposed by Chierichetti et al. [29] in the context of maintaining fairness in *unsupervised* machine learning tasks. Their model is based on the concept of *disparate impact* [89] (and the p%-rule). The input points are assumed to have a binary sensitive attribute modeled by two colors, and discrimination based on this attribute is to be avoided. Since preserving exact balance in each cluster may be very costly or even be impossible[2], the idea is to ensure that at least $1/t$ of the points of each cluster are of the minority color, where t is a parameter. A cluster with this property is called *fair*, and the fairness constraint can now be added to any clustering problem, giving rise to fair k-center, fair k-median, etc. Chierichetti et al. develop a 4-approximation algorithm for fair k-center and a $(t + 1 + \sqrt{3} + \varepsilon)$-approximation algorithm for fair k-median.

The fair clustering model as proposed by Chierichetti et al. can also be used to incorporate other aspects into our school assignment example: For example, we might want to mitigate effects of gentrification or segregation. For these use cases, we need multiple colors. Then, in each cluster, the ratio between the number of points with one specific color and the total number of points shall be in some given range. If the allowed range is $[0.20, 0.25]$ for red points, we require that in each cluster, at least a fifth and at most a fourth of the points are red. This models well established notions of fairness (statistical parity, group fairness), which require that each cluster exhibits similar compositional makeup as the overall data with respect to a given attribute. One downside of this notion is that a malicious user could create an illusion of fairness by including proxy points: If we wanted to create a boy-heavy school in our above example, we could still achieve the desired parity by assigning only girls that are very unlikely to attend. Thus, instead of enforcing *equal representation* in the above sense, one could also ask for *equal opportunity* as proposed by Hardt et al. [52] for the case where we take binary decisions (i.e., $k = 2$) and have access to a labeled training set. This approach, however, raises the philosophical question if this equality of opportunity is a sufficient condition for the absence of discrimination. Rather than delving into this complex and much debated issue, we refer to the excellent surveys by Romei and Ruggieri [89] and Žliobaitė et al. [96] that systematically discuss different forms of discrimination and how they can be detected. We assume that it is the intent of the user to achieve a truly fair solution.

Finding fair clusterings turns out to be an interesting challenge from the point of view of combinatorial optimization. As other clustering problems with side constraints, it loses the property that points can be assigned locally. But while many other constrained problems at least allow polynomial algorithms that assign points to given centers

[1] Or, incorporating the capacities, ensure that the teacher:boys:girls ratio is 1:$\frac{s}{2}$:$\frac{s}{2}$.

[2] Imagine a point set with 49 red and 51 blue points: This cannot at all be divided into true subsets with the exact same ratio.

optimally, we show that even this restricted problem is NP-hard in the case of fair k-center.

Chierichetti et al. tackle fair clustering problems by a two-step procedure: First, they compute a micro clustering into so-called *fairlets*, which are groups of points that are fair and cannot be split further into true subsets that are also fair. Secondly, representative points of the fairlets are clustered by an approximation algorithm for the unconstrained problem. Consider the special case of a point set with 1:1 ratio of red and blue points. Then a fairlet is a pair of one red and one blue point, and a good micro clustering can be found by computing a suitable bipartite matching between the two color classes.

The problem of computing good fairlets gets increasingly difficult when considering more general variants of the problem. For multiple colors and the special case of exact ratio preservation (i.e., for all colors, the allowed range for its ratio is one specific number), the fairlet computation problem can be reduced to a capacitated clustering problem. This is used in [90] to obtain a 14(15)-approximation algorithm for fair k-center(k-supplier) problem with multiple colors and exact ratio preservation.

In Section 2.2.1, we give an extensive overview of the existing results and further the fairlet approach in order to explore its applicability for different variants of fair clustering. Two major issues arise: Firstly, capacitated clustering is not solved for all clustering objectives; indeed, finding a constant-factor approximation algorithm for capacitated k-median is a long-standing open problem. Secondly, (even for k-center) it is unclear how fairlets even look like when we have multiple colors and want to allow ranges for the ratios. In this situation, subsets of very different size and composition may satisfy the desired ratio.

Our main contribution with regards to fair clustering is a very different approach. We start with a solution to the unconstrained problem. Based on the given solution, we derive a fair clustering solution with the same centers. That is achieved by a technique that we call *weakly supervised LP rounding*: We solve an LP for the fair clustering problem and then combine it with the integral unfair solution by careful rounding. We use this method to prove the following statements.

Theorem 7. *There exists a* 5(7)*-approximation algorithm for the fair k-center(k-supplier) problem with exact preservation of ratios.*

Essentially fair refers to our notion of bicriteria approximation: A cluster C is *essentially fair* if there exists a fractional fair cluster C', such that the mass of points in C' differs from the number of points in C by *at most* 1 and for each color h the number of color h points in C differs by *at most* 1 from the mass of color h points in C'.

Theorem 8. *Given any set of centers S, there exists an assignment ϕ' which is essentially fair, and which incurs a cost that is linear in the cost of S for the unconstrained problem and the cost of an optimal fractional fair clustering of P, for all objectives k-center, k-supplier, k-median, k-means, and facility location.*

Corollary 9. *There exist essentially fair approximation algorithms for fair clustering problems with approximation ratios as stated in Table 1.1.*

Fair Clustering Problem	**Essentially fair approximation ratio**
k-center	3
k-supplier	5
facility location	3.488
k-median	4.675
k-means	62.856

Table 1.1: Essentially fair approximation ratios for fair clustering problems.

An essentially fair cluster can be viewed as a cluster with a small additive fairness violation. For large clusters, an additive violation of 1 will translate into a negligible multiplicative violation. Of course if a cluster is very small or a color has a very small ratio, then a violation of 1 is large in multiplicative terms. However, consider an example with one majority color having 500 points and 50 colors having 10 point each. Then our model means that in a cluster of 100 points, because the *overall* amount of violation is at most one point, the majority color can at most have 51 points, and at least 49 of the points belong to minority colors. Although it is possible that some minorities will not be represented at all, no minority would be represented by more than 2 points, which implies that at least half of the minorities are represented.

For the bicriteria approximation algorithms in Theorem 8 and Corollary 9, we can start with *any* unconstrained starting solution. We thus say that our algorithm is a *black-box* approximation algorithm. For Theorem 7, this is not the case; here we need to use a specific starting solution. We prove Theorem 8 in Section 2.2.3 and Theorem 7 in Section 2.2.4.

Our results have two advantages. Firstly, we get results for a wide range of clustering problems, and these results improve previous results. For example, we get a 5-approximation algorithm for the fair k-center problem with exact ratio preservation, where the best known guarantee was 14. All our bicriteria results work for multiple colors and approximate ratio preservation, a case for which no previous algorithm was known. As for the quality of the guarantees, compare the 4.675-approximation algorithm for essentially fair k-median clusterings with the best previously known $\Theta(t)$-approximation algorithm, which is only applicable to the case of two colors. Notice that a similar result can *not* be achieved by using bicriteria approximation algorithms for capacitated clustering. The reduction from capacitated clustering only works when the capacities are not violated.

Secondly, the black-box approach has the advantage that fairness can be established belatedly, in a situation where the centers are already given [36, 101]. Consider our school example and notice that the location of the schools cannot be chosen. Our result says that if we are alright with essentially fair clusterings, we get a clustering which is not much more expensive than a fair clustering where the centers were chosen without the fairness constraint at hand.

1.2.3 Privacy Preserving Clustering

The abundance of constraints and the difficulty to adjust methods for all of them individually asks for ways to *add* a constraint to an approximation algorithm in an oblivious way. Instead of adjusting and reproving known algorithms, we would much rather like to take an algorithm as a black-box and ensure that the solution satisfies one more constraint in addition. This is a challenging request. We start the investigation of such add-on algorithms by studying private clustering in more detail. Indeed, we develop a method to add the privacy constraint to approximation algorithms for constrained k-center problems. That means that we use an approximation algorithm as a subroutine and ensure that the final solution will additionally respect a given lower bound. The method has to be adjusted depending on the constraint, but it is oblivious to the underlying approximation algorithm used for that constraint.

This works for the basic k-center problem (giving an algorithm for the private k-center problem), but we also show how to use the method when the underlying approximation algorithm is for the k-center problem with outliers, the fair k-center problem, the capacitated k-center problem and the fair capacitated k-center problem. We also demonstrate that our method suffices to approximate the *strongly private* k-center problem, where we assume a protected feature like in fair clustering, but instead of fairness, now demand that a minimum number of points of each color is assigned to each open center to ensure anonymity for each class individually.

Our technique

The general structure of the algorithm is based on standard thresholding [53], i.e., the algorithm tests all possible thresholds and chooses the smallest for which it finds a feasible solution. For each threshold, it starts with the underlying algorithm and computes a non-private solution. Then it builds a suitable network to shift points to satisfy the lower bounds. The approximation ratio of the method depends on the underlying algorithm and on the structure of this network.

The shifting does not necessarily work right away. If it does not produce a feasible solution, then using the max-flow min-cut theorem, we obtain a set of points for which we can show that the clustering uses too many clusters (and can thus not satisfy the lower bounds). We then recompute the solution in this part. Depending on the objective function, we have to overcome different hurdles to ensure that the recomputation works in the sense that it a) makes sufficient progress towards finding a feasible solution and b) does not increase the approximation ratio. The process is then iterated until we find a feasible solution.

Our results

We obtain the following results for multiple combinations of privacy with other constraints. Our reductions can handle the general case; whether the resulting algorithm is then for k-center or k-supplier thus depends on the evoked underlying algorithm.

- We obtain a 4-approximation algorithm for private k-center with outliers (5 for the supplier version). This matches the best known bounds [3] ([7] for the supplier version (this also holds for non-uniform lower bounds)).
- We obtain an 14-approximation algorithm for private and uniform capacitated k-center (i.e., centers have a uniform lower bound *and* a uniform upper bound), and an 8-approximation algorithm for the private capacitated k-center problem with soft uniform capacities. The best known bounds for these two problems is 6 [32]. For the supplier version we obtain a 15-approximation algorithm (The best known bound is 9 [32]).
- We achieve constant-factor approximation algorithms for private fair capacitated/uncapacitated k-center/k-supplier clustering. The approximation ratio depends on the *balance* of the input point set and the type of upper bounds, it ranges between 8 in the uncapacitated case where for each color h the number of points with color h is an integer multiple of the number of points with the rarest color and 195 in the general supplier version with non-uniform upper bounds. To the best of our knowledge, all these combinations have not been studied before.
- Finally, we propose the *strongly private k-center problem*. As in the fair clustering problem, the input here has a protected feature like gender, modeled by colors. Now instead of a fair clustering, we aim for anonymity for each color, meaning that we have a lower bound for each color. Each open center needs to be assigned this minimum number of points for each color. To the best of our knowledge, this problem has not been studied before; we obtain a 4-approximation algorithm as well as a 5-approximation algorithm for the supplier version.

Since our method does not require knowledge of the underlying approximation algorithm, the approximation guarantees improve if better approximation algorithms for the underlying problems are found. There is also hope that our method could be used for new, not yet studied constraints, with not too much adjustment.

1.2.4 Preliminaries

Points and locations. Let (X, d) be a semi-metric space, i.e., X is a set and $d : X \times X \to \mathbb{R}_{\geq 0}$ is a semi-metric. A semi metric is a non-negative function that fulfills

$$\begin{aligned} d(x,y) &= 0 \Leftrightarrow x = y, \\ d(x,y) &= d(y,x) \quad \text{for all } x, y \in X \end{aligned}$$

and a *β-relaxed triangle inequality*

$$d(x,z) \leq \beta(d(x,y) + d(y,z)) \quad \text{for all } x, y, z \in X \tag{1.1}$$

for some $\beta \geq 1$. We use $d(x,T) = \min_{y \in T} d(x,y)$ for the smallest distance between $x \in X$ and a set $T \subseteq X$. For two sets $S, T \subseteq X$, we use $d(S,T) = \min_{x \in S, y \in T} d(x,y)$ for the smallest distance between any pair $x \in S, y \in T$. We are given a set of n

points $P \subseteq X$ and a set of potential locations $L \subseteq X$. We allow L to be infinite when $X = \mathbb{R}^m$ for some $m \in \mathbb{N}$ and $L = \mathbb{R}^m$, otherwise we assume that L is a finite set. The task is to open a subset $S \subseteq L$ of the locations and to assign each point in P to an open location via a mapping $\phi : P \to S$. We refer to the set of all points assigned to a location $i \in S$ by $P(i) := \phi^{-1}(i)$. The assignment incurs a cost governed by the semi-metric $d : (P \cup L) \times (P \cup L) \to \mathbb{R}_{\geq 0}$

Additionally, we may have opening costs $f_i \geq 0$ for every potential location $i \in L$ or a maximum number of centers $k \in \mathbb{N}$, i.e., we demand $|S| \leq k$. We will use n to denote the number of points $|P|$.

Objectives. We consider versions of several classical clustering problems. An instance is given by $I := (P, L, d, f, k)$, where $P \subseteq X$ denotes a set of points, $L \subseteq X$ denotes the set of possible locations, d denotes the distance function on X, $f : L \to \mathbb{R}$ denotes the opening cost, and $k \in \mathbb{N}$ denotes the maximum number of clusters allowed. Our goal is to choose a solution (S, ϕ), with $S \subseteq L$, $|S| \leq k$, and $\phi : P \to S$ according to one of the following objectives.

- **k-center**, **k-supplier** and **Euclidean k-center**: minimize the maximum distance between a point and its assigned location: Minimize $\max_{j \in P} d(j, \phi(j))$. In these problems, we have $f \equiv 0$ and d is a metric. Furthermore, in k-center, $L = P$, whereas in k-supplier, $L \neq P$ is some finite set, and in Euclidean k-center, $X = \mathbb{R}^m$, $L = \mathbb{R}^m$ for some $m \in \mathbb{N}$ and $d(x, y) = ||y - x|| = \sqrt{\sum_{i=1}^{m}(y_i - x_i)^2}$ is the Euclidean distance on $\mathbb{R}^m$.
- **k-median**: minimize $\sum_{j \in P} d(j, \phi(j))$, $f \equiv 0$, d is a metric and $L \subseteq P$.
- **k-means**: minimize $\sum_{j \in P} d(j, \phi(j))$, $f \equiv 0$, where $X = \mathbb{R}^m$ for some $m \in \mathbb{N}$, $L = \mathbb{R}^m$ and $d(x, y) = ||y - x||^2$, the squared Euclidean distance between x and y, is a semi-metric for $\beta = 2$.
- **facility location**: minimize $\sum_{j \in P} d(j, \phi(j)) + \sum_{i \in S} f_i$, where $k = n$, d is a metric and L is a finite set.

In constrained versions of the above mentioned clustering problems, an instance additionally contains parameters describing the constraints, and a solution (S, ϕ) must satisfy the corresponding property as described for each constraint.

Colors and fairness. In fair clustering we are additionally given a set of *colors* Col, and a coloring $col : P \to Col$ that assigns a color to each point $j \in P$. For any set of points $P' \subseteq P$ and any color $h \in Col$ we define $col_h(P') = \{j \in P' \mid col(j) = h\}$ to be the set of points colored with h in P'. We call $r_h(P') := \frac{|col_h(P')|}{|P'|}$ the *ratio* of h in P'. If an implicit assignment ϕ is clear from the context, we write $col_h(i)$ to denote the set of all points of a color $h \in Col$ assigned to an $i \in S$, i.e., $col_h(i) = col_h(P(i))$.

A set of points $P' \subseteq P$ is *exactly fair* if P' has the same ratio for every color as P, i.e., for each $h \in Col$ we have $r_h(P') = r_h(P)$. We say that P' is *ℓ, u-fair* or just *fair* for some $\ell = (\ell_h \mid h \in Col)$ and $u = (u_h \mid h \in Col)$ if we have $r_h(P') \in [\ell_h, u_h]$ for every color $h \in Col$.

Additional constraints.

- **Fairness:** In fair clustering problems, we want to preserve the ratios of colors found in P in our clusters. We distinguish two cases:
  - **exact fairness:** For the exact preservation of ratios, we ask that all clusters are exactly fair, i.e., $P(i)$ is exactly fair for all $i \in S$.
  - **relaxed fairness:** For the relaxed preservation of ratios, we are given lower and upper bounds $\ell = \{\ell_h \in \mathbb{Q} \mid h \in Col\}$ and $u = \{u_h \in \mathbb{Q} \mid h \in Col\}$ on the required ratio of colors in each cluster and ask that all clusters are ℓ, u*-fair*.

  The exact case is a special case of the relaxed case where we set $\ell_h = u_h = r_h(P)$ for every color $h \in Col$. *Essentially fair* clusterings are defined in Section 2.2.2 (see Definition 36).
- **Privacy:** In private clustering we are given a lower bound $\ell \in \mathbb{N}$ and demand $\ell \leq |P(i)|$ for every selected center $i \in S$.
- **Strong privacy:** In *strongly private* clustering the input contains a coloring $col : P \to Col$ of points as well as a lower bound $\ell_h \in \mathbb{N}$ for each color $h \in Col$. Now the assignment is restricted to ensure that it satisfies the lower bound for the points of each color, i.e., $\ell_h \leq |col_h(i)|$ for each $i \in S$ and $h \in Col$.
- **Capacity:** The *capacity* constraint comes with an upper bound function $u : L \to \mathbb{N}$ for which we demand $|P(i)| \leq u(i)$ for every selected center $i \in S$. When we have $u(x) = u$ for all $x \in L$ and some $u \in \mathbb{N}$, then we say that the capacities are *uniform*, otherwise, we say they are *non-uniform*. The *soft* capacity constraint is a version of the capacity constraint, which allows a clustering to open multiple centers at the same location $i \in L$. The number of points assigned to each of these centers can be at most $u(i)$.
- **Outlier:** An instance of a clustering problem with *outliers* additionally contains a parameter $o \in \mathbb{N}$ for the maximum number of outliers. A solution is then allowed to ignore up to o points. Therefore the problem is to compute a set of centers $S \subseteq L$ and an assignment $\phi : P \to S \cup \{out\}$, with $|P(out)| \leq o$, which assigns each point to a center in S or to be an outlier. In the computation of the cost, we assume $d(j, out) = 0$ for all $j \in P$.

The fair assignment problem. For all the clustering objectives mentioned above, we call the subproblem of computing a cost-minimal fair assignment of points to a given set $S \subseteq L$ of centers the *fair assignment problem.* We show the following theorem in Section 2.2.5.

Theorem 10. *Finding an α-approximation for the fair assignment problem for k-center is* NP*-hard for any $\alpha < 3$.*

Previous results used as black-box algorithms We make use of known results for several constraints. We state the best known bounds and their references in Table 1.2.

	Vanilla	Capacities		Outlier	Fair	
		uniform	non-uniform		$\frac{r}{b} \in \mathbb{N}$	general
k-center	2 [53]	6 [66]	9 [8]	2 [24]	4 [29]	5 (Thm. 44)
k-supplier	3 [53]		11 [8]	3 [26]		7 (Thm. 45)

Table 1.2: An overview on the approximation results that we combine with privacy.

1.3 Parity Games

A parity game [37] is a two-player game of perfect information played on a directed graph G by two players, *even* and *odd*, who move a token from node to node along the edges of G so that an infinite path is formed. The nodes of G are partitioned into two sets V_0 and V_1; the even player moves if the token is at a node in V_0 and the odd player moves if the token is at a node in V_1. The nodes of G are labeled by a *priority function* $p : V \to \mathbb{N}_0$, and the players compete for the parity of the highest priority occurring infinitely often on the infinite path $v_0, v_1, v_2 \ldots$ describing a play: the even player wins if $\limsup_{i\to\infty} p(v_i)$ is even, and the odd player wins if it is odd.

The winner determination problem for parity games is the algorithmic problem to determine for a given parity game $G = (V_0 \uplus V_1, E, p)$ and an initial node $v_0 \in V_0 \cup V_1$, whether the even player has a winning strategy in the game if the token is initially placed on node v_0. We say that an algorithm for this problem *solves* parity games. Parity games have various applications in computer science and the theory of formal languages and automata in particular. They are closely related to other games of infinite duration, such as mean payoff games, discounted payoff games, and stochastic games [61]. Solving parity games is linear-time equivalent to the model checking problem for the modal μ-calculus [94]. Hence, any parity game solver is also a model checker for the μ-calculus (and vice versa).

Many algorithms have been suggested for solving parity games [20, 64, 97, 103], yet none of them is known to run in polynomial time. McNaughton [81] showed that the winner determination problem belongs to the class $\mathsf{NP} \cap \mathsf{coNP}$, and Jurdziński [61] strengthened this to $\mathsf{UP} \cap \mathsf{coUP}$. It is a long-standing open question whether parity games can be solved in polynomial time. The first quasi-polynomial algorithm is due to Calude et al. [23] and has a run time of $O(n^{O(\log n)})$. Later Jurdziński and Lazić [63] showed a quasi-polynomial algorithm which reduced the space requirement from quasi-polynomial to quasi-linear.

As a polynomial-time algorithm for solving parity games has remained elusive, researchers have started to consider which restrictions on the game allow for polynomial-time algorithms. One such well-studied restriction is the treewidth t of the underlying undirected graph G of the game. Obdržálek [86] found an algorithm solving parity games on n nodes in time $n^{O(t^2)}$. Later, Fearnley and Lachish [38] gave an algorithm solving parity games in time $n^{O(t \log n)}$. Another well-studied parameter for parity games is the number p of distinct priorities by which the nodes of the game are labeled. The progress-measure lifting algorithm by Jurdziński [62] solves parity games in time

$O(pm(2n/p)^{p/2})$, where m denotes the number of edges of G. This run time has been improved by Schewe [91] to $O(m((2e)^{3/2}n/p)^{p/3})$. The first fixed-parameter-tractable algorithm for the parameter p is by Calude et al. [23] and has a run time of $O(2^p n^4)$. Fearnley and Schewe [39] presented an algorithm for solving parity games with run time $O(n(t+1)^{t+5}(p+1)^{3t+5})$, assuming that a tree decomposition of G with width t is given.

For a given parameter κ, one usually aims for *fixed-parameter-tractable algorithm* algorithms, i.e., algorithms that run in time $f(\kappa) \cdot n^c$ for some computable function f and some constant c that is independent of κ. Such an algorithm can be practical for large instances if f grows moderately and c is small. From the previously mentioned algorithms only the algorithm by Fearnley and Schewe [39] is a fixed-parameter-tractable algorithm for the combined parameter (t, p). It is not known if fixed-parameter-tractable algorithms exist for the parameter t or the parameter p alone.

Further parameters for which polynomial-time algorithms for parity games have been suggested include DAG-width [17], clique-width [87], and entanglement [19]; none of these are fixed-parameter-tractable algorithms.

1.3.1 Our contributions

We study as parameter the number k of nodes that belong to the player who controls the smaller number of nodes in the parity game. Our first result is a *subexponential* fixed-parameter-tractable algorithm for solving general parity games for parameters p and k and for parameter only k for bipartite parity games (where players alternate between their moves).

Theorem 11. *There exists a deterministic algorithm that solves any parity game G on n nodes and m edges in time $(p+k)^{O(\sqrt{k})} \cdot O(pnm)$, where k denotes the minimum number of nodes owned by one of the players and p denotes the number of distinct priorities. If G is bipartite, the algorithm runs in time $k^{O(\sqrt{k})} \cdot O(n^3)$.*

Thus, our algorithm is particularly efficient if the game is unbalanced, in the sense that one player owns only k nodes and the other player owns the remaining $n - k \gg k$ nodes.

Let us remark that it is not very hard to show fixed-parameter-tractability for parameter $p + k$; indeed McNaughton's algorithm [81] can be shown to run in time $p^k \cdot n^{O(1)}$, and this was improved to $p^{\log k} \cdot 4^k \cdot n^{O(1)}$ by Gajarský et al. [43]. Our key contribution here is to reduce the dependence of k to a *subexponential* function. Indeed, this improvement allows us to derive the following immediate corollary of Theorem 11 to expedite the run time for solving *general* parity games.

Corollary 12. *There exists a deterministic algorithm that solves parity games in time $n^{O(\sqrt{k})}$.*

Our algorithm is asymptotically always at least as fast as the fastest known deterministic parity game solver by Jurdziński, Paterson, and Zwick [64], which runs in time $n^{O(\sqrt{n})}$.

For the case $k = o(n)$, our algorithm is asymptotically faster than theirs and constitutes the fastest known deterministic solver for such games.

We also prove the existence of a small kernel, as our second result. For a parameterized problem, a *kernelization algorithm* takes as input an instance x with parameter κ and computes in time $(|x| + \kappa)^{O(1)}$ an equivalent instance x' with parameter κ' (a *kernel*) with size $|x'| \leq g(\kappa)$, for some computable function g; here, equivalent means that an optimal solution for x can be derived in polynomial time from an optimal solution of x'.

Theorem 13. *Parity games can be kernelized in time $O(pmn)$ to an instance with at most $(p+1)^k + (p+1)k$ nodes, and bipartite parity games can be kernelized in time $O(n^3)$ to an instance with at most $k + 2^k \cdot \min\{k, p\}$ nodes and at most $k2^k \cdot \min\{k, p\}$ edges.*

This kernelization result is not only interesting for its own sake, but it is also an ingredient in the proof of Theorem 11.

As our third result, we generalize the algorithm by Jurdziński, Paterson, and Zwick [64] for parity games with maximum out-degree 2 to arbitrary out-degree Δ.

Theorem 14. *There is a deterministic algorithm that solves parity games on n nodes out of which s_j nodes have out-degree at most j in time*

$$n^{O\left(\min_{1 \leq j \leq n}\left\{\sqrt{n - s_j} + \sqrt{\frac{s_j}{\log_j s_j}}\right\}\right)}.$$

Corollary 15. *There is a deterministic algorithm that solves parity games on n nodes with maximum out-degree $\Delta_{\max}$ in time $n^{O(\sqrt{\log(\Delta_{\max}) \cdot n / \log(n)})}$ and parity games on n nodes with average out-degree Δ_{avg} in time $n^{O(\sqrt{\log(\log(n)\Delta_{\text{avg}}) \cdot n / \log(n)})}$.*

1.3.2 Detailed comparison with previous work

Let us discuss in detail how our results compare to previous work. It is well-known (cf. [68, Lemma 3.2]) and easy to prove that the treewidth of a complete bipartite graph equals the size of the smaller side. Since the treewidth of a graph can only decrease when deleting edges, the graph underlying a bipartite parity game in which one player owns k nodes has a treewidth of at most k. However, as it is not known if there exists a fixed-parameter-tractable algorithm for parameter treewidth, the result in Theorem 11 for the bipartite case does not follow from previous work about parity games with bounded treewidth. As a parity game in which one player owns k nodes can have up to n different priorities, also the fixed-parameter-tractable algorithm for the combined parameter (t, p) by Fearnley and Schewe [39] does not imply our result. As the exponent of our run time does not depend on p the fixed-parameter-tractable algorithm for parameter p by Calude et al. [23] also does not imply our result.

The algorithm of Jurdziński, Paterson, and Zwick [64] for parity games with maximum out-degree two with run time $n^{O(\sqrt{n/\log n})}$ can easily be generalized to arbitrary parity games at the expense of its run time. For this, one only needs to observe that every parity game can be transformed into a game with maximum out-degree two by replacing each node with a higher out-degree by an appropriate binary tree. This transformation increases the number of nodes from n to $\Theta(m)$ where m denotes the number of edges in the original parity game. Hence, the run time becomes $m^{O(\sqrt{m/\log m})} = n^{O(\sqrt{m/\log n})}$. For graphs with average out-degree $\Delta = \omega(\log\log n)$ the resulting run time of $n^{O(\sqrt{\Delta n/\log n})}$ is asymptotically worse than the run time we obtain in Corollary 15

For graphs in which the variance of the out-degrees is large, our algorithm can even be better than stated in Corollary 15. If, for example, there are $n^{1-\varepsilon}$ nodes with an arbitrary out-degree for some $\varepsilon > 0$ and all remaining nodes have constant out-degree at most c then our algorithm has a run time of $n^{O(\sqrt{\frac{n}{\log n}})}$ (the minimum in Theorem 14 is assumed for $j = c$). This matches the best known bound for randomized algorithms.

Gajarský et al. [43] present an algorithm that solves parity games in time $w^{O(\sqrt{w})} \cdot n^{O(1)}$, where w denotes the modular width of G. Since the modular width of a bipartite graph can be exponential in the size of the smaller side, Theorem 11 does not follow from this result.

1.4 Outline and Bibliographical Notes

In the following chapters, we prove the results about clustering (Chapter 2) and parity games (Chapter 3) stated in the introduction. In Chapter 4 we draw conclusions and discuss open questions.

The results stated in this dissertation are based on the following work.

- The results shown in Section 2.1 are based on currently unpublished joint work with Matthias Mnich and Heiko Röglin.
- The results shown in Section 2.2 are based on joint work with Ioana Bercea, Martin Groß, Samir Khuller, Aounon Kumar, Daniel Schmidt, and Melanie Schmidt [16].
- The results shown in Section 2.3 are based on joint work with Melanie Schmidt [90].
- The results in shown in Chapter 3 are based on joint work with Matthias Mnich and Heiko Röglin [85].

Chapter 2

Clustering

In this chapter, we consider different types of clustering problems and discuss the results stated in Section 1.2. The first problem is the Euclidean k-center problem in Section 2.1. We specifically study its special case for $k = 1$, the smallest enclosing ball problem. We continue with fair clustering in Section 2.2, where we discuss fairness for several clustering objectives. We conclude the chapter with a discussion about combining privacy with other constraints in Section 2.3.

2.1 The Euclidean k-Center Problem

Before we explain how we can solve the Euclidean k-center problem, we define the smallest enclosing ball problem.

For $c \in \mathbb{R}^m$ and $r \in \mathbb{R}_{\geq 0}$ let $B(c, r) = \{x \in \mathbb{R}^m \mid d(x, c) \leq r\}$ denote the m-dimensional sphere with center c and radius r. Let $\delta(B(c, r)) := \{x \in \mathbb{R}^m \mid d(x, c) = r\}$ denote the boundary of $B(c, r)$. For a finite and non-empty set $P \subseteq \mathbb{R}^m$ and a point $c \in \mathbb{R}^m$, let $B(c, P)$ denote the sphere $B(c, \max_{j \in P} d(j, c))$, i.e., the smallest sphere with center c that encloses the points in P. The smallest enclosing ball $B(P)$ of a finite and non-empty point set $P \subseteq \mathbb{R}^m$ is defined as the sphere of minimal radius which contains the points in P, i.e., the sphere $B(c, P)$ of smallest radius over all $c \in \mathbb{R}^m$. The existence and uniqueness of $B(P)$ are well-known [99].

We now formally define the smallest enclosing ball problem which is equivalent to the Euclidean k-center problem for $k = 1$:

SMALLEST ENCLOSING BALL (SEB) PROBLEM
Input: A finite and non-empty set $P \subseteq \mathbb{R}^m$.
Task: Find $B(P)$, i.e., find $c \in \mathbb{R}^m$ such that $B(P) = B(c, P)$.

Any feasible solution (S, ϕ) of the Euclidean k-center problem can be described as a combination of a partitioning $\bigcup_{i \in S} P(i)$ of P into $|S| \leq k$ subsets and the feasible solution $B(i, P(i))$ of the SEB problem on $P(i)$ for each $i \in S$. Since it cannot destroy the optimality of a solution, we assume that for each point $j \in P$ $\phi(j)$ is the closest point to j in S.

Also keeping the same partitioning $\bigcup_{i \in S} P(i)$ and replacing every $i \in S$ by the unique $i' \in \mathbb{R}^m$ with $B(i', P(i)) = B(P(i))$ can only improve a given solution.

This implies that we can take any optimal solution (S, ϕ), look a the partitioning $\bigcup_{i \in S} P(i)$ it induces and optimally solve the SEB problem on $P(i)$ for every $i \in S$ to obtain another optimal solution (S', ϕ').

As Theorem 5 states that we can solve the SEB problem optimally on rational instances this reduces the Euclidean k-center problem on rational instances to finding a partitioning of P, which is induced by an optimal solution.

As the number of possible ways to partition a set of n points into at most k sets is $O(k^n)$ this implies the following statement.

Corollary 16. *The optimal Euclidean k-center solution of a rational instance $P \subseteq \mathbb{Q}^m$ can be computed in $O(k^n poly(enc))$ time, where enc denotes the input size of the instance.*

Fortunately we do not need to check all possible ways to partition P into k sets, as a lot of the possible partitionings cannot be obtained by selecting up to k centers and assigning every point in P to its closest center. To reduce the number of relevant partitionings we take a look at Voronoi diagrams in $\mathbb{R}^m$.

Definition 17. *Let $S \subseteq \mathbb{R}^m$ be a finite set of points. For a point $s \in S$ the Voronoi region R_s associated with s is the set of all points in $\mathbb{R}^m$ whose distance to s is not greater than their distance to any other point $s' \in S$, i.e., $R_s = \{x \in \mathbb{R}^m \mid d(x, s)\ led(x, s')$ for all $s' \in S\}$. We call s the center of R_s. The Voronoi diagram of S is the tuple of the Voronoi regions $(R_s)_{s \in S}$.*

Notice that a point $x \in \mathbb{R}^m$ can only be part of more than one Voronoi region if it lies on the boundary of each such region and has the same distance to each of their centers. For each set $S \subset \mathbb{R}^m$ the Voronoi diagram naturally partitions P into at most $|S|$ sets (with points that are part of multiple Voronoi regions assigned arbitrarily). We call such a partitioning a *Voronoi partitioning* of P by the Voronoi diagram $(R_s)_{s \in S}$. By definition each partitioning of P obtained by selecting up to k centers and assigning every point in P to its closest center is a Voronoi partitioning of P by Voronoi diagram of the set of selected centers. It is therefore enough to only look at partitionings of P that are Voronoi partitionings of P by the Voronoi diagram of a set $S \subseteq \mathbb{R}^m$ with $|S| = k$. Imai, Inaba, and Katoh showed that the number of Voronoi partitionings of n points by the Voronoi diagram generated by k points in the m-dimensional Euclidean space is $O(n^{mk})$ [57] and Imai and Inaba showed that they can be enumerated in $O(n^{(m+1)k})$ time [56]. We therefore need to test a most $O(n^{mk})$ different partitionings and together with Theorem 5 proves Theorem 6.

It is now left to show how the SEB problem can be solved and prove Theorem 5 .

2.1.1 The Smallest Enclosing Ball Problem

Before we explain the algorithm to solve the SEB problem, we introduce more notation. Given a subset $T \subseteq P$ of at most $m+1$ affine independent points, we denote by $B_\Delta(T)$ the smallest sphere which has all points in T on its boundary, i.e., $B_\Delta(T) := B(c,r)$ for

$$\begin{aligned} r &= \min\{r' \in \mathbb{R}^+ \mid \exists\, c' \in \mathbb{R}^m : \; d(j,c') = r' \text{ for all } j \in T\}, \\ c &\in \{c' \in \mathbb{R}^m \mid d(j,c') = r \text{ for all } j \in T\} \ . \end{aligned}$$

We will call the center of $B_\Delta(T)$ the circumcenter of T and denote it by $cc(T)$. Note that $B_\Delta(T)$ will always be the unique sphere with center in the affine hull $\mathsf{aff}(T)$ and T on its boundary. A non-empty affinely independent set $T \subseteq P$ will be called the *support set* of $B_\Delta(T)$.

The Algorithm

We now describe the algorithm for instances with integral coefficients and later explain how we can transform instances with rational coefficients into instances with integral coefficients without increasing the input size too much.

First, we design the SEB problem as an optimization problem over a convex body and show that our description satisfies certain criteria stated in [49] which imply that a version of the ellipsoid method can compute a $(1+\varepsilon)$-approximation in time polynomial in the encoding size of the instance and $\log \frac{1}{\varepsilon}$.

Proposition 18. *There exists an algorithm* $\boldsymbol{seb_{approx}}$ *that, for any given* $\varepsilon > 0$ *and point set* $P \in \mathbb{Z}^m$*, computes a* $(1+\varepsilon)$*-approximation of* $B(P)$ *in time polynomial in the encoding size of* P *and* $\log \frac{1}{\varepsilon}$*.*

The SEB problem has been described as an optimization problem over a convex body before. Zhou, Toh, and Sun for example describe the SEB problem as a second-order cone program [102], while Gärtner and Schönherr describe it as a quadratic program [45]. As far as we know, for all previous descriptions of the SEB problem as an optimization problem over a convex body, the analysis focused on the number of arithmetic operations a solver would need in order to find a solution and did not go into detail about the necessary precision and the encoding length needed during the computations.

For a given set $P \subseteq \mathbb{Z}^m$, let $c_{\max} = \max\{|j_i| \mid j \in P \text{ and } 1 \le i \le m\}$ denote the largest absolute value over all coordinates of points in P, and let

$$\varepsilon_{m,c_{\max}} = \max_{i\in\mathbb{N}} \left\{ \frac{1}{2^{2i+2}} \;\middle|\; \frac{1}{2^{2i}} < \frac{1}{25m^2(2c_{\max})^{8m+12}} \right\} \ . \tag{2.1}$$

Our extended algorithm, named $\mathbf{seb}^+$, takes the set $P \subseteq \mathbb{Z}^m$ as input and computes a $(1+\varepsilon_{m,c_{\max}})$-approximation with $\mathbf{seb_{approx}}$ to obtain a sphere B with center c and radius r'. We let $r = \min_{i\in\mathbb{N}} \{i \cdot \varepsilon_{m,c_{\max}} \mid i \cdot \varepsilon_{m,c_{\max}} \ge r'\}$. In all non-trivial integral instances we have $r_{\mathsf{opt}} \geq \frac{1}{2}$. Therefore we can assume $r \leq (1 + 3\varepsilon_{m,c_{\max}})r_{\mathsf{opt}}$. Given c, and r, we then show that the support T_{opt} of the optimal solution contains exactly those points $j \in P$ for which $d(c,j) \geq (1 - 2\sqrt{\varepsilon_{m,c_{\max}}})\frac{r}{1+\varepsilon_{m,c_{\max}}}$. Once we have T_{opt}, we can easily compute c_{opt}.

The pseudo-code of algorithm $\mathbf{seb}^{+}$ is given as Algorithm 1.

Algorithm 1 $\mathbf{seb}^{+}(P)$

```
Input: A finite, non-empty set $P \subseteq \mathbb{Z}^m$ of points.
Output: A sphere with minimal radius containing $P$.
1: Set $\varepsilon = \varepsilon_{m,c_{\max}}$;
2: Compute $B(c, r') = \mathbf{seb}_{\mathbf{approx}}(P, \varepsilon)$;
3: Let $r = \min_{i \in \mathbb{N}} \{i\varepsilon \mid i\varepsilon > r'\}$
4: Set $T = \{p \in P \mid d(p, c) \geq (1 - 2\sqrt{\varepsilon})\frac{r}{1+\varepsilon}\}$;
5: return $B_{\Delta}(T)$
```

SEB as Optimization over a Convex Body

In order to apply previous results by Yudin and Nemirovskii [100] to prove Proposition 18, we now describe the SEB problem as an optimization problem over a convex body

Let $P \subseteq \mathbb{Z}^m$ be an instance of the SEB problem with $c_{\max} = \max\{|j_i| \mid j \in P \text{ and } 1 \leq i \leq m\}$. Every feasible ball of the SEB problem for P consists of a center $c \in \mathbb{R}^m$ and a radius $r \in \mathbb{R}$ s.t. $r \geq \|j - c\|$ for all $j \in P$. The general set of feasible balls can therefore be described as $\mathit{Sol}(P) = \{(r, c) \in \mathbb{R} \times \mathbb{R}^m \mid \forall j \in P : r \geq \|j - c\|\}$.

Corollary 19. *$\mathit{Sol}(P)$ is an unbounded convex set.*

Proof. For any $(r, c) \in \mathit{Sol}(P)$ and any $r' \geq r$ we have $r' \geq r \geq \|j - c\|$ for all $j \in P$, which implies $(r', c) \in \mathit{Sol}(P)$. Since $\mathit{Sol}(P)$ is obviously not empty it has to be unbounded.

Let $(r_1, c_1), (r_2, c_2) \in \mathit{Sol}(P)$ and let $\lambda \in [0, 1]$. We need to show that $(\lambda r_1 + (1 - \lambda)r_2, \lambda c_1 + (1 - \lambda)c_2) \in \mathit{Sol}(P)$, i.e., for each point $j \in P$ we have $\lambda r_1 + (1 - \lambda)r_2 \geq \|j - (\lambda c_1 + (1 - \lambda)c_2)\|$.

Let $v \in \mathbb{R}^m$ be a vector orthogonal to $c_1 - c_2$ with $\|v\| = \|c_1 - c_2\|$, such that the plane $E = c_2 + \mu(c_1 - c_2) + \nu v$ contains j. Let $c = \|c_1 - c_2\|$ and $j = c_2 + a(c_1 - c_2) + bv$ for some $a, b \in \mathbb{R}$, then we have

$$\begin{aligned}
&r_2 \geq \|j - c_2\| = \sqrt{a^2c^2 + b^2c^2} = c\sqrt{a^2 + b^2},\\
&r_1 \geq \|j - c_1\| = \sqrt{(a-1)^2c^2 + b^2c^2} = c\sqrt{(a-1)^2 + b^2} \text{ and}\\
&\|j - (\lambda c_1 + (1 - \lambda)c_2)\| = \sqrt{(a-\lambda)^2c^2 + b^2c^2} = c\sqrt{(a-\lambda)^2 + b^2}.
\end{aligned}$$

It is left to show

$$c\sqrt{(a-\lambda)^2+b^2} \leq \lambda c\sqrt{(a-1)^2+b^2} + (1-\lambda)c\sqrt{a^2+b^2}$$

to conclude the proof. As we know $c \geq 0$, it is enough to show

$$\sqrt{(a-\lambda)^2+b^2} \leq \lambda\sqrt{(a-1)^2+b^2} + (1-\lambda)\sqrt{a^2+b^2}.$$

We have

$$\begin{aligned}
&0 \leq b^2 \\
\Leftrightarrow &a^4 - 2a^3 + 2a^2b^2 + a^2 - 2ab^2 + b^4 \leq a^4 - 2a^3 + 2a^2b^2 + a^2 - 2ab^2 + b^4 + b^2 \\
\Leftrightarrow &(a^2+b^2-a)^2 \leq (a^2+b^2)((a-1)^2+b^2).
\end{aligned}$$

As $(a^2+b^2)((a-1)^2+b^2)$ is non-negative it follows

$$\begin{aligned}
&(a^2+b^2-a) \leq \sqrt{(a^2+b^2)((a-1)^2+b^2)} \\
\Leftrightarrow &2(\lambda-\lambda^2)(a^2+b^2-a) \leq 2(\lambda-\lambda^2)\sqrt{(a^2+b^2)((a-1)^2+b^2)} \\
\Leftrightarrow &-2\lambda a \leq -2a\lambda^2 + (2\lambda^2-2\lambda)(a^2+b^2) + 2(\lambda-\lambda^2)\sqrt{(a^2+b^2)((a-1)^2+b^2)} \\
\Leftrightarrow &a^2 - 2\lambda a + \lambda^2 + b^2 \\
&\quad\leq \lambda^2(-2a+1) + (1-2\lambda+2\lambda^2)(a^2+b^2) + 2(\lambda-\lambda^2)\sqrt{(a^2+b^2)((a-1)^2+b^2)} \\
\Leftrightarrow &(a-\lambda)^2 + b^2 \\
&\quad\leq \lambda^2((a-1)^2+b^2) + (1-\lambda)^2(a^2+b^2) + 2\lambda(1-\lambda)\sqrt{(a^2+b^2)((a-1)^2+b^2)} \\
\Leftrightarrow &(a-\lambda)^2 + b^2 \leq \left(\lambda\sqrt{(a-1)^2+b^2} + (1-\lambda)\sqrt{a^2+b^2}\right)^2 \\
\Leftrightarrow &\sqrt{(a-\lambda)^2+b^2} \leq \lambda\sqrt{(a-1)^2+b^2} + (1-\lambda)\sqrt{a^2+b^2} \\
\Leftrightarrow &c\sqrt{(a-\lambda)^2+b^2} \leq \lambda c\sqrt{(a-1)^2+b^2} + (1-\lambda)c\sqrt{a^2+b^2}. \\
\Rightarrow &c\sqrt{(a-\lambda)^2+b^2} \leq \lambda r_1 + (1-\lambda)r_2.
\end{aligned}$$

□

For every $x \in \mathbb{R}^+$ we define $Sol_x(P) := \{(r,c) \in Sol(P) \mid r \leq x\}$ as the set of feasible balls with radius at most x. $Sol_x(P)$ obviously is a convex set. $x \geq r_{\mathsf{opt}}$ implies that $Sol_x(P)$ contains the optimal solution $(r_{\mathsf{opt}}, c_{\mathsf{opt}})$ and $(r_{\mathsf{opt}}, c_{\mathsf{opt}}) = \arg\min_{(r,c)\in Sol_x(P)} r$. Since, by definition of $c_{\max}$ we have $P \subseteq B(0^m, m \cdot c_{\max})$, we know $r_{\mathsf{opt}} \leq m \cdot c_{\max}$ and therefore $Sol_{m\cdot c_{\max}}(P)$ contains $(r_{\mathsf{opt}}, c_{\mathsf{opt}})$. As $Sol_{m\cdot c_{\max}}(P)$ is a convex set we can apply known results for convex optimization problems in order to obtain a good approximation to the SEB problem.

Before we state the theorem that goes back to Yudin and Nemirovskii [100], which will allow us to prove Proposition 18, we introduce the following definitions.

Definition 20. *A centered convex body is a quintuple $(K; m, R, r', a_0)$, where $K \subseteq \mathbb{R}^m$ is a compact and full dimensional convex set, $m \in \mathbb{N}$, $R, r' \in \mathbb{Q}$ and $a_0 \in \mathbb{Q}^m$ and we have $B(a_0, r') \subseteq K \subseteq B(0, R)$.*

Definition 21. *For any set $K \subseteq \mathbb{R}^m$ and any real number $\delta > 0$ let*

- $S(K, \delta) := \{x \in \mathbb{R}^m \mid \|x - y\| \leq \delta \text{ for some } y \in K\}$ *and*
- $S(K, -\delta) := \{x \in K \mid S(\{x\}, \delta) \subseteq K\}$.

Note for a sphere B we have $S(B, -\delta) = \{q \in B \mid \inf_{p \notin B} d(p, q) \geq \delta\}$.

The Weak Membership Problem (WMEM)
Input: A convex body K, a vector $y \in \mathbb{Q}^m$ and a rational number $\gamma > 0$.
Task: Assert that $y \in S(K, \gamma)$, or assert that $y \notin S(K, -\gamma)$.

We can now state the following theorem which goes back to Yudin and Nemirovskii [100]. For a more detailed description and proofs we refer to [49].

Theorem 22. *Given a rational number $\varepsilon' > 0$, a centered convex body $(K; m, R, r', a_0)$ given by an oracle for the weak membership problem, and a convex function $f : \mathbb{R}^m \to \mathbb{R}$ given by an oracle that, for every $x \in \mathbb{Q}^m$ and $\gamma > 0$, returns a rational number t such that $|f(x) - t| \leq \gamma$, we can compute a vector $y \in S(K, \varepsilon')$ such that $f(y) \leq f(x) + \varepsilon'$ for all $x \in S(K, -\varepsilon')$ in oracle polynomial time.*

Here *oracle polynomial time* means that the number of calls to the oracles as well as the time needed for all additional computations is polynomial in the encoding sizes of ε and $(K; m, R, r', a_0)$.

With Theorem 22 we can now prove Proposition 18.

Proof (Proposition 18). Theorem 22 needs a centered convex body and we need to be able to easily check for any point that is not close to its boundary if the point is inside or outside the convex body. We create our centered convex body as follows. Let $K = Sol_{2m \cdot c_{\max}}(P)$, $m' = m + 1$, $R = 4m \cdot c_{\max}$, $r' = \frac{1}{4}m \cdot c_{\max}$ and $a_0 = (1.5m \cdot c_{\max}, 0, \ldots, 0)$. To show that $(K; m', R, r', a_0)$ is a centered convex body we need to show $B(a_0, r') \subseteq K \subseteq B(0, R)$. Note that $B(a_0, r')$ and $B(0, R)$ are spheres in $\mathbb{R}^{m+1}$, the solution space of the SEB problem, and not in $\mathbb{R}^m$, the space of the original instance. To make the distinction easier we will use $\mathcal{B}$ instead of B for spheres in $\mathbb{R}^{m+1}$. Let $a = (r_a, c_a) \in \mathbb{R} \times \mathbb{R}^m$ and $r \in \mathbb{R}$ then $\mathcal{B}(a, r)$ contains all points $(r_1, c_1) \in \mathbb{R} \times \mathbb{R}^m$ with $\|(r_1 - r_a, c_1 - c_a)\| = \sqrt{(r_1 - r_a)^2 + \|c_1 - c_a\|^2} \leq r$. Given $r_a \geq r \geq 0$ the sphere $\mathcal{B}(a, r)$ is equivalent to the set of spheres in $\mathbb{R}^m$, which contains a sphere $B(c_1, r_1)$ if and only if we have $\sqrt{(r_1 - r_a)^2 + \|c_1 - c_a\|^2} \leq r$. We then abuse notation and say that $\mathcal{B}(a, r)$ contains $B(c_1, r_1)$ and also write $B(c_1, r_1) \in \mathcal{B}(a, r)$. We must have $B((0, \ldots, 0), 1.5m \cdot c_{\max}) \in K$ as $1.5m \cdot c_{\max} \leq 2m \cdot c_{\max}$ and $B((0, \ldots, 0), 1.5m \cdot c_{\max})$ contains P. Let $B(c_1, r_1) \in \mathcal{B}(a_0, r')$ then we know $\sqrt{(r_1 - 1.5m \cdot c_{\max})^2 + \|c_1 - (0, \ldots, 0)\|^2} \leq \frac{1}{4}m \cdot c_{\max}$. This implies $r_1 \in [1.25m \cdot c_{\max}, 1.75m \cdot c_{\max}]$ and $\|c_1\| \leq \frac{1}{4}m \cdot c_{\max}$. Since the distance between c_1 and any point in P is no more than $\|c_1\| + m \cdot c_{\max} \leq 1.25m \cdot c_{\max}$, we know $B(c_1, r_1) \in Sol_{2m \cdot c_{\max}}(P)$. As K contains exactly the spheres in $\mathbb{R}^m$, which contain P and have a radius of at most $2m \cdot c_{\max}$, and we know that P is non-empty and contains only points $p \in \mathbb{R}^m$ with $\|p\| \leq m \cdot c_{\max}$, we know for any sphere $B(c, r) \in K$ that we must have $\|c\| \leq 3m \cdot c_{\max}$. This implies $\sqrt{r^2 + \|c\|^2} \leq \sqrt{(2m \cdot c_{\max})^2 + (3m \cdot c_{\max})^2} = \sqrt{13}m \cdot c_{\max} \leq 4m \cdot c_{\max}$. We then have $K \subseteq \mathcal{B}(0, R)$ because $\mathcal{B}(0, R)$ contains all spheres $B(c, r) \in \mathbb{R}^m$ with $\sqrt{(r-0)^2 + \|c - (0, \ldots, 0)\|^2} = \sqrt{r^2 + \|c\|^2} \leq 4m \cdot c_{\max} = R$.

Therefore $(K; m', R, r', a_0)$ is a centered convex body that contains the optimal solutions $B(c_{\mathsf{opt}}, r_{\mathsf{opt}}) = \arg\min_{B(c,r) \in Sol_{2m \cdot c_{\max}}(P)} r$.

In order to check for a given $(r, c) \in \mathbb{Q} \times \mathbb{Q}^m$ if we have $B(c, r) \in Sol_{2m \cdot c_{\max}}(P)$ we need to check if we have $r \leq 2m \cdot c_{\max}$ and $\|p_i - c\| \leq r$ for all $i \in \{1, \ldots, n\}$. This can obviously be done in time polynomial in n, m and the encoding size of r, c and $c_{\max}$.

This gives us an oracle to the weak membership problem with polynomial run time.

Let $f((r, c)) := r$. For every $x = (r_x, c_x) \in \mathbb{Q}^{m+1}$ and $\gamma > 0$ it is easy to return $t = r_x$ such that $|f(x) - t| = 0 \leq \gamma$ (Note that in order to minimize memory it would also be sufficient to let $\alpha = \max_{i \in \mathbb{N}}\{\frac{1}{2^i} \mid \frac{1}{2^i} \leq \gamma\}$ and let $f(x)$ be equal to r_x rounded to the nearest multiple of α).
The choice of $(K; m', R, r', a_0)$ ensures that for every $B(c, r) \in S(K, \varepsilon')$ we have $\|j - c\| \leq r + \varepsilon'$ for all $j \in P$. Together with the choice of f and $\gamma = \varepsilon'$ it ensures $\min\{f(x) + \varepsilon' \mid x \in S(K, -\varepsilon')\} \leq r_{\mathsf{opt}} + 2\varepsilon'$. Let $B(c, r) \in S(K, \varepsilon')$ be a sphere such that $r \leq f(x) + \varepsilon'$ for all $x \in S(K, -\varepsilon')$ then $B(c, r') = r + \varepsilon'$ is a feasible solution with $r' \leq r_{\mathsf{opt}} + 3\varepsilon'$.
Given the fact that we have $r_{\mathsf{opt}} \geq \frac{1}{2}$ for every non-trivial integral instance $B(c, r')$ is a $(1 + 6\varepsilon')$-approximation. Setting $\varepsilon' = \frac{1}{6}\varepsilon$ concludes the proof of Proposition 18. □

Analysis of Algorithm 1

Overview of the analysis We start by showing for any $P \subseteq \mathbb{Z}^m$ a lower bound on the minimal non-zero distance $\gamma > 0$ between any point in $j \in P$ and the boundary of the sphere $B_\Delta(T)$, for any affine set $T \subseteq P$. Then we show the following "separation property": for any point $j \in P$, its distance $d(c_{\mathsf{opt}}, j)$ to the center c_{opt} of the optimal solution is either equal to the radius r_{opt} of the optimal sphere, or is at most $r_{\mathsf{opt}} - \gamma$.

Next, we use algorithm $\mathbf{seb}_{\mathbf{approx}}$ to obtain a feasible solution $B(c, r)$ with radius $r \leq (1 + \varepsilon) r_{\mathsf{opt}}$ and center c. We show $\|c - c_{\mathsf{opt}}\| \leq \sqrt{3\varepsilon} r_{\mathsf{opt}}$, which in turn implies that

$$\begin{aligned} d(c,j) &\in ((1-\sqrt{3\varepsilon})r_{\mathsf{opt}}, r) \subseteq ((1-2\sqrt{\varepsilon})\frac{r}{1+\varepsilon}, r) \text{ for all } j \in T_{\mathsf{opt}}; \\ d(c,j) &\leq (1+\sqrt{3\varepsilon})r_{\mathsf{opt}} - \gamma \leq (1+2\sqrt{\varepsilon})r - \gamma \text{ for all } j \in P \setminus T_{\mathsf{opt}}\ . \end{aligned}$$

Choosing ε according to (2.1), we obtain $(1 + 2\sqrt{\varepsilon})r - \gamma < (1 - 2\sqrt{\varepsilon})\frac{r}{1+\varepsilon}$. This means that all points $j \in P$ with $d(c, j) \geq (1 - 2\sqrt{\varepsilon})\frac{r}{1+\varepsilon}$ must be in T_{opt}. Finally, given T_{opt} one easily obtains $B(P) = B_\Delta(T_{\mathsf{opt}})$.

Correctness of Algorithm 1 Let $\varepsilon > 0$, $m \in \mathbb{N}$ and $r \in \mathbb{R}^+$. Let $B \subseteq \mathbb{R}^m$ be a m-dimensional sphere with radius r. We denote by $X(B, \varepsilon)$ the set of points inside B that are ε-close to the boundary of B, i.e.,

$$X(B, \varepsilon) = \{q \in B \setminus \delta(B) \mid \inf_{p \in \delta(B)} d(p, q) \leq \varepsilon\} \ .$$

Definition 23. *For $\gamma > 0$, a set $P \subseteq \mathbb{R}^m$ is* γ-sphere-separated *if for each affine independent subset $T \subseteq P$, the sets P and $X(B_\Delta(T), \gamma)$ are disjoint.*

Lemma 24. *Let $P \subseteq \mathbb{Z}^m$ be a set of points and let $c_{\max} := \max_{j \in P, i \in \{1,\ldots,m\}} |j_i|$ be the maximum absolute value of any coordinate of a point in P. Then P is γ-sphere-separated for $\gamma = \frac{1}{2\sqrt{m}(2c_{\max})^{4m+5}}$.*

Proof. At first we will show, that for any affine subset $T \subseteq P$ of points the coordinates of the circumcenter $cc(T)$ are rational numbers with a common denominator of at most $(2c_{\max})^{2m+2}$. This allows us to see that for any affine subset $T \subseteq P$ there is some $c \in \mathbb{N}$ with $c \leq (2c_{\max})^{2m+2}$ such that the finite grid

$$G = \{(a_1, \ldots, a_m) \mid a_i = b_i/c \text{ for some } b_i \in \mathbb{Z} \text{ with } |b_i/c| \leq c_{\max}\} \tag{2.2}$$

contains the circumcenter $cc(T)$ as well as P. We then show how to compute a lower bound—depending on c and $c_{\max}$—on the minimal non-zero difference in distance between two pairs of points on the grid; that will conclude the proof.

Let us start by considering the circumcenter $cc(T)$. It is known that for any affine set $T \subseteq P$, $cc(T)$ is the unique point in the affine hull of T which has the same distance to all points in T. That is, $cc(T) = \sum_{j \in T} \alpha_j j$ for some $\alpha = (\alpha_j \mid j \in T) \in \mathbb{R}^{|T|}$ with $\sum_{j \in T} \alpha_j = 1$, and there exists some $r \in \mathbb{R}$ such that

$$d(cc(T), j) = r \quad \text{for each } j \in T. \tag{2.3}$$

Condition (2.3) is equivalent to

$$\sum_{i=1}^{m} (cc(T)_i^2 - 2j_i cc(T)_i + j_i^2) = |cc(T) - j|^2 = r^2 \text{ for each } j \in T. \tag{2.4}$$

Since the terms $cc(T)_i^2$ appear in this equation for every $j \in T$, condition (2.4) is equivalent to the existence of some $r' \in \mathbb{N}$ such that

$$\sum_{i=1}^{m} (-2j_i cc(T)_i + j_i^2) = r' \text{ for all } j \in T \ .$$

Treating $\alpha_i, i = 1, \ldots, k$ and $cc(T)_j, j = 1, \ldots, m$ as variables, we obtain a set of $m + k + 1 \leq 2m + 2$ linear equations with integral coefficients of value at most $2c_{\max}$. As $cc(T)$ is the unique point in the affine hull of T which has the same distance to all points in T, this system of linear equations admits a unique solution. We can obtain this solution via Gaussian elimination and therefore all variables in the solution have a rational value with a common denominator $c \leq (2c_{\max})^{2m+2}$. Therefore, $cc(T)$ and P are contained in the finite grid G defined by (2.2). The distance between two points of the grid is equal to the square root of an integer multiple of c^{-2}, and thus equal $\frac{\sqrt{x}}{c}$ for some integer x. The distances between two separate pairs of points are therefore either the same or differ by at least

$$\frac{1}{c}\left(\sqrt{x}-\sqrt{x-1}\right) \tag{2.5}$$

with $x \leq m(2c_{\max})^{4m+6}$. As $(\sqrt{x}+\frac{1}{2\sqrt{x}})^2 = x-1+\frac{1}{4x} > x-1 = (\sqrt{x-1})^2$ the value of (2.5) is at least $1/(2c\sqrt{x}) \geq 1/(2c\sqrt{m(2c_{\max})^{4m+6}}) = 1/(2c\sqrt{m}(2c_{\max})^{2m+3}) \geq 1/(2\sqrt{m}(2c_{\max})^{4m+5})$.

□

We will now show an upper bound for the distance between c_{opt} and the center c of any $(1+\varepsilon)$-approximation for the smallest enclosing ball of P. Combined with Lemma 24 this will allow us to determine the points in T_{opt} through their distance to c.

We use the following fact, which goes back to an idea of Seidel and can be proved by the Karush-Kuhn-Tucker optimality conditions for constrained optimization [88].

Proposition 25. *Let T be a set of points on the boundary of some sphere B with center c. Then $B = B(T)$ if and only if $c \in conv(T)$, where $conv(T)$ denotes the convex hull of T.*

Lemma 26. *For any $\varepsilon \in (0,1]$ and any $(1+\varepsilon)$-approximation $B(c,r)$ of $B(P) = B(c_{\mathsf{opt}}, r_{\mathsf{opt}})$, it holds $d(c,c_{\mathsf{opt}}) \leq \sqrt{3\varepsilon}r_{\mathsf{opt}}$.*

Proof. Let T_{opt} be the support set of $B(P)$ and let $c' \neq c_{\mathsf{opt}} \in \mathbb{R}^m$ be a point with $r' = \max_{j\in P} d(c',j) \leq (1+\varepsilon)r_{\mathsf{opt}}$. By Proposition 25, c_{opt} is part of the convex hull of T_{opt}; therefore, there is some $j \in T_{\mathsf{opt}}$ for which the angle $\angle c'c_{\mathsf{opt}}j$ has value at least 90°. We thus have

$$\begin{aligned} r_{\mathsf{opt}}^2 + d(c',c_{\mathsf{opt}})^2 &= d(c_{\mathsf{opt}},j)^2 + d(c',c_{\mathsf{opt}})^2 \\ &\leq d(c',j)^2 \\ &\leq ((1+\varepsilon)r_{\mathsf{opt}})^2 \ . \end{aligned}$$

Where the last inequality comes from the fact that $B(c',r')$ is a $(1+\varepsilon)$-approximation of $B(P)$ and we must have $\max_{j\in P} d(c,j) \leq (1+\varepsilon)r_{\mathsf{opt}}$. Therefore, $d(c',c_{\mathsf{opt}})^2 \leq (2\varepsilon+\varepsilon^2)r_{\mathsf{opt}}^2 \leq 3\varepsilon r_{\mathsf{opt}}^2$ and hence $d(c',c_{\mathsf{opt}}) \leq \sqrt{3\varepsilon}r_{\mathsf{opt}}$.

□

Combining Lemma 26 with Lemma 24 we can show the following lemma:

Lemma 27. *Let $\varepsilon \in (0,1]$ and let $B = B(c,r)$ be a $(1+\varepsilon)$-approximation of $B(P) = B(c_{\mathsf{opt}}, r_{\mathsf{opt}})$. Let T_{opt} be the support set of $B(P)$. Then $d(c,j) \in ((1-2\sqrt{\varepsilon})\frac{r}{1+\varepsilon}, r)$ for all $j \in T_{\mathsf{opt}}$ and $d(c,j) < (1+2\sqrt{\varepsilon})r - \frac{1}{2\sqrt{m}(2c_{\max})^{4m+5}}$ for any $j \in P \setminus T_{\mathsf{opt}}$.*

Proof. By Lemma 24 we know $d(c_{\mathsf{opt}}, j) < r_{\mathsf{opt}} - \frac{1}{2\sqrt{m}(2c_{\max})^{4m+5}}$ for every $j \in P \setminus T_{\mathsf{opt}}$. With Lemma 26 we then have $d(c,j) < d(c, c_{\mathsf{opt}}) + d(c_{\mathsf{opt}}, j) < (1+2\sqrt{\varepsilon})r_{\mathsf{opt}} - \frac{1}{2\sqrt{m}(2c_{\max})^{4m+5}} \leq (1+2\sqrt{\varepsilon})r - \frac{1}{2\sqrt{m}(2c_{\max})^{4m+5}}$.

For $j \in T_{\mathsf{opt}}$ we know $d(c_{\mathsf{opt}}, j) = r_{\mathsf{opt}}$ which implies $d(c,j) > d(c_{\mathsf{opt}}, j) - d(c, c_{\mathsf{opt}}) \geq (1-\sqrt{3\varepsilon})r_{\mathsf{opt}}$. By definition of r we know $r \leq r' + \varepsilon$ and since $\mathbf{seb}_{\mathbf{approx}}$ is a $(1+\varepsilon)$-approximation algorithm we know $r' \leq (1+\varepsilon)r_{\mathsf{opt}}$ and therefore $r \leq (1+\varepsilon)r_{\mathsf{opt}} + \varepsilon$. Since $r_{\mathsf{opt}} \geq 1/2$ and $\varepsilon < 1/4$, we have

$$\begin{aligned}
&\frac{\varepsilon(1-2\sqrt{\varepsilon})}{(2-\sqrt{3})\sqrt{\varepsilon}(1+\varepsilon)} < \frac{\sqrt{\varepsilon}}{2-\sqrt{3}} \leq \frac{1}{2} \leq r_{\mathsf{opt}} \\
\Leftrightarrow &\frac{\varepsilon}{1+\varepsilon} < \frac{(2-\sqrt{3})\sqrt{\varepsilon}}{1-2\sqrt{\varepsilon}} r_{\mathsf{opt}} \\
\Leftrightarrow & r_{\mathsf{opt}} + \frac{\varepsilon}{1+\varepsilon} < \frac{1-\sqrt{3\varepsilon}}{1-2\sqrt{\varepsilon}} r_{\mathsf{opt}} \\
\Leftrightarrow &(1+\varepsilon) r_{\mathsf{opt}} + \varepsilon < (1+\varepsilon)\frac{1-\sqrt{3\varepsilon}}{1-2\sqrt{\varepsilon}} r_{\mathsf{opt}} \\
\Rightarrow &(1-2\sqrt{\varepsilon})\frac{r}{1+\varepsilon} < (1-\sqrt{3\varepsilon}) r_{\mathsf{opt}} \quad .
\end{aligned}$$

□

To show that Algorithm 1 returns the optimal solution, we want to ensure

$$(1+2\sqrt{\varepsilon})r - \frac{1}{2\sqrt{m}(2c_{\max})^{4m+5}} < (1-2\sqrt{\varepsilon})\frac{r}{1+\varepsilon}. \tag{2.6}$$

In that case we have $T_{\mathsf{opt}} = \{j \in P \mid d(c,j) \in ((1-2\sqrt{\varepsilon})\frac{r}{1+\varepsilon}, r)\}$. Thus, it suffices to show that the choice of $\varepsilon = \varepsilon_{m,c_{max}}$ in (2.1) is small enough. To this end, we first observe that the relation $r^2 \leq mc_{\max}^2$ yields

$$\varepsilon = \varepsilon_{m,c_{\max}} < \frac{1}{25m^2(2c_{\max})^{8m+12}} < \frac{1}{100r^2m(2c_{\max})^{8m+10}} \quad .$$

Together with the condition $0 < \varepsilon \leq 1$, we obtain

$$\begin{aligned}
&\varepsilon < \frac{1}{100r^2m(2c_{\max})^{8m+10}} \\
\Leftrightarrow &5\sqrt{\varepsilon} < \frac{1}{2r\sqrt{m}(2c_{\max})^{4m+5}} \\
\Rightarrow &2\sqrt{\varepsilon} + \frac{\varepsilon+2\sqrt{\varepsilon}}{1+\varepsilon} < \frac{1}{2r\sqrt{m}(2c_{\max})^{4m+5}} \\
\Leftrightarrow &\frac{\varepsilon + 2\varepsilon\sqrt{\varepsilon} + 4\sqrt{\varepsilon}}{1+\varepsilon} < \frac{1}{2r\sqrt{m}(2c_{\max})^{4m+5}} \\
\Leftrightarrow &1 + 2\sqrt{\varepsilon} - \frac{1-2\sqrt{\varepsilon}}{1+\varepsilon} < \frac{1}{2r\sqrt{m}(2c_{\max})^{4m+5}}
\end{aligned}$$

$$\Leftrightarrow (1+2\sqrt{\varepsilon})r - (1-2\sqrt{\varepsilon})\frac{r}{1+\varepsilon} < \frac{1}{2\sqrt{m}(2c_{\max})^{4m+5}}$$
$$\Leftrightarrow (1+2\sqrt{\varepsilon})r - \frac{1}{2\sqrt{m}(2c_{\max})^{4m+5}} < (1-2\sqrt{\varepsilon})\frac{r}{1+\varepsilon} \ .$$

Hence, we have shown the following:

Lemma 28. *Algorithm 1 is correct and returns the smallest enclosing ball $B(P)$.*

Next we will take a look at the encoding length of the objects used during Algorithm 1. Together with Proposition 18 it will then be easy to see that all steps of Algorithm 1 can be performed with time and space polynomial in the encoding size of P.

Length of encoding the solution As shown in the proof of Lemma 24, the coordinates of the circumcenter $cc(T)$ of any any affine subset $T \subseteq P$ are rational numbers with common denominator at most $c_{\max}^{2m+2}$. Since the absolute value of each coordinate is at most $c_{\max}$, the numerators have absolute value at most $c_{\max}^{2m+3}$. Both denominator and numerator can thus be encoded using $O(\log(c_{\max}^{2m+3})) = O(m\log(c_{\max}))$ bits.

Note that in many cases the radius of the smallest enclosing ball is irrational. But the proof of Lemma 24 showed it to be the square root of an integer multiple $x \cdot c^{-2}$ of some c^{-2}, $c \in \mathbb{N}$ with $c \leq c_{\max}^{2m+2}$ and $x \leq 4mc_{\max}^{2(2m+3)}$. Similarly, the radius of the approximation obtained in line 3 is an integer multiple $y \cdot \varepsilon_{m,c_{\max}}$ of $\varepsilon_{m,c_{\max}}$ with $y \leq \frac{m2c_{\max}}{\varepsilon_{m,c_{\max}}}$. Instead of computing the actual radii, we can use their squares, which are rational numbers, for all necessary computations. Finally, the values of x, y and c can be encoded with $O(m\log(c_{\max}))$ bits.

Polynomial run time of Algorithm 1 The choice of $\varepsilon = \varepsilon_{m,c_{\max}}$ and Proposition 18 show that steps 1 and 2 of Algorithm 1 need time polynomial in the encoding size of P and $\log\frac{1}{\varepsilon}$, i.e., polynomial in n, m and $\log c_{\max}$.

By the choice of $\varepsilon = \varepsilon_{m,c_{\max}}$ and the choice of r in line 3 the value of $(1-2\sqrt{\varepsilon})\frac{r}{1+\varepsilon}$ in line 4 has to be rational number smaller than $2m \cdot c_{\max}$. We now show that its denominator is smaller than $2829m^3(2c_{max})^{12m+18}$. As $r = j\varepsilon$ and $\varepsilon = \frac{1}{2^{2i+2}}$ for some $i, j \in \mathbb{N}$ we know that $\frac{r}{1+\varepsilon} = j\varepsilon\frac{1}{1+\varepsilon} = j\varepsilon\frac{\varepsilon^{-1}}{1+\varepsilon^{-1}} = \frac{j}{\varepsilon^{-1}+1} = \frac{j}{2^{2i+2}+1}$. We then have $(1-2\sqrt{\varepsilon})\frac{r}{1+\varepsilon} = (1-\frac{2}{2^{i+1}})\frac{j}{2^{2i+2}+1} = \frac{2^{i+1}-2}{2^{i+1}}\frac{j}{2^{2i+2}+1} = \frac{(2^{i+1}-2)j}{2^{3i+3}+2^{i+1}}$. By the definition of $\varepsilon_{m,c_{\max}}$ we know $2^{2i-1} \leq 25m^2(2c_{max})^{8m+12}$ and therefore $2^{3i+3} + 2^{i+1} \leq 2^{4.5} \cdot 2^{2i-1} \cdot 2^{i-0.5} + 2^{1.5}2^{i-0.5} \leq 2^{4.5} \cdot 25m^2(2c_{max})^{8m+12} \cdot 5m(2c_{max})^{4m+6} + 2^{1.5}5m(2c_{max})^{4m+6} \leq 2829m^3(2c_{max})^{12m+18}$. Therefore steps 3 and 4 can also be performed in time polynomial in n, m and $\log c_{\max}$.

Transformation of rational instances It is easy to retrieve the smallest enclosing ball of a set of points from the smallest enclosing ball of a scaled version of the same points. For a set P of points and a constant $k > 0$ let $P_k = \{k \cdot j \mid j \in P\}$ be the set P scaled by k, where for any $j \in \mathbb{R}^m$ the point $k \cdot j$ has coordinates $(k \cdot j)_i := k \cdot j_i$.

Lemma 29. *Let $P \subseteq \mathbb{R}^m$ be a set of points and let $B(c,r)$ be the smallest enclosing ball of P. Then for any number $k > 0$, $B(kc, kr)$ is the smallest enclosing ball of P_k.*

Proof. We first show that $B(kc, kr)$ is an enclosing ball for P_k. For every point $j \in P_k$ we have $\frac{1}{k}j \in P$. As $B(c,r)$ is an enclosing ball for P, we have $d(kc, j) = k \cdot d(c, \frac{1}{k}j) \leq k \cdot r$.

Now suppose, for sake of contradiction, that there exists an enclosing ball $B(c', r')$ for P_k with $r' < kr$. For every point $j \in P$ we then have $d(\frac{1}{k}c', j) = \frac{1}{k}d(c', kj) \leq \frac{1}{k}r' < r$. Thus, $B(\frac{1}{k}c', \frac{1}{k}r')$ would be an enclosing ball for P smaller than $B(c,r)$—a contradiction.

□

Let $P \subseteq \mathbb{Q}^m$ be a finite instance of the SEB problem where all coordinates have rational values. Let $cd(P)$ denote the common denominator of coordinates of the points in P, i.e., the common denominator of the set $\bigcup_{j \in P}\{j_1, \dots, j_m\}$. Then $P_{cd(P)}$ is an instance with integral coordinates. Let $denom_{\max}$ be the largest denominator of any coordinate of any point in P and let $nomin_{\max}$ be the largest absolute value of any nominator of any coordinate of any point in P:

$$\begin{aligned}
&denom_{\max} = \\
&\max\{x \mid \exists j \in P, 1 \leq i \leq m \text{ s.t. } j_i = \frac{y}{x} \text{ for some } x, y \in \mathbb{Z} \text{ with } gcd(x,y) = 1\}, \\
&nomin_{\max} = \\
&\max\{y \mid \exists j \in P, 1 \leq i \leq m \text{ s.t. } j_i = \frac{y}{x} \text{ for some } x, y \in \mathbb{Z} \text{ with } gcd(x,y) = 1\}.
\end{aligned}$$

Then $cd(P) \leq denom_{\max}^{nm}$, as there are at most nm different coordinates. Hence, the maximum absolute value of any coordinate in $P_{cd(P)}$ is at most $nomin_{\max} denom_{\max}^{nm}$. Thus the space needed to encode an integral equivalent to P is in $O(nm(nm \log(denom_{\max}) + \log(nomin_{\max})))$.

This completes the proof of Theorem 5. □

2.2 Fair Clustering

In this section we discuss several types of fair clustering. First we explain the fairlet approach, which has been used for the fair k-center problem in previous work. We extend the fairlet approach to more cases of fair clustering, discuss its advantages and drawbacks, and show in which cases it already leads to approximation algorithms. Then, in order to compute results in some of the cases, for which it is still unknown how to compute good fairlets in polynomial time, and to improve the approximation ratio for the fair k-center/k-supplier problem, we explain our LP-based approaches. We first explain the approach for essentially fair clustering and then show our improved result for the fair k-center/k-supplier problem. We conclude the section with some insight into the difficulties of fair clustering. We show NP-hardness for the fair assignment problem as well as the unbounded integrality gap of the clustering LP.

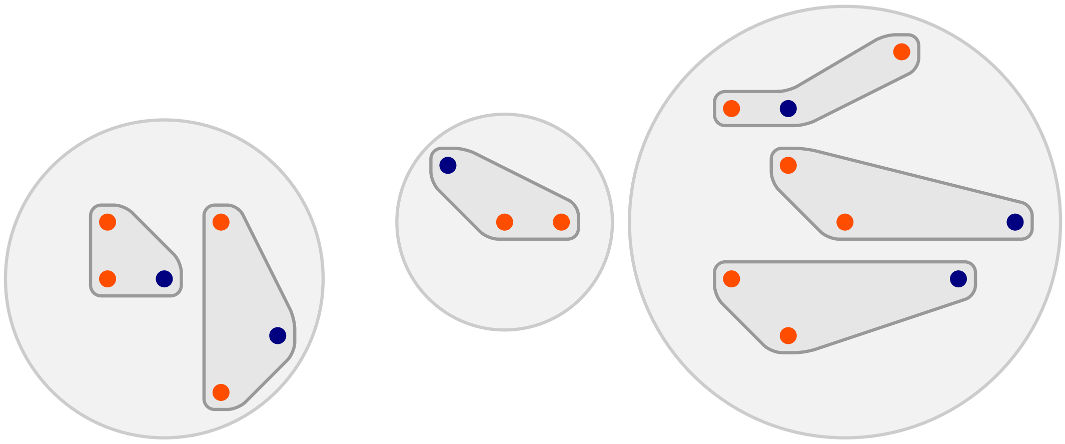

Figure 2.1: A decomposition of a fair clustering into fairlets with two red points (●) and one blue point (●).

2.2.1 Computing Fair Clusterings via Fairlets

Fairlets and fairlet decompositions Chierichetti et al. [29] make an important observation: Any cluster of a fair clustering is composed of *atomic* subclusters. For instance, if we are to achieve a ratio of 2/1 of red to blue points, then a cluster with two red and one blue point cannot be divided into fair subclusters. Moreover, any fair cluster with an overall balance ratio of 2/1 consists of an integer number of atomic subclusters – each having two red and one blue point. Figure 2.1 visualizes this concept. We follow [29] and call these atomic microclusters *fairlets*. The concept naturally generalizes to clusterings under relaxed fairness.

Definition 30. *A set of points $F \subseteq P$ is called a* fairlet *if F is fair and cannot be partitioned into multiple, non-empty fair sets. Given $\ell = (\ell_h \mid h \in Col)$ and $u = (u_h \mid h \in Col)$ a set $F \subseteq P$ is called an ℓ, u-fairlet if F is ℓ, u-fair and cannot be partitioned into multiple, non-empty ℓ, u-fair sets.*

Note that, compared to [29], we use a different definition for fairlets. For the case of exact fairness covered in [29] both definitions refer to the same objects and our definition gives a natural generalization to relaxed fairness. When we refer to fair clusters and fairlets we also mean ℓ, u-fair and ℓ, u-fairlets (unless stated otherwise) as most of the proofs are analogous. As observed above, every fair cluster can be partitioned into fairlets. The converse is also true: The union of disjoint fairlets is a fair set of points.

Definition 31. *Let $P = \bigcup_{i \in I} F_i$ be a partitioning of P such that $F_i \subseteq P$ is a fairlet (an ℓ, u-fairlet) for each $i \in I$. Then we call $\mathcal{F} = \{F_i \mid i \in I\}$ a* fairlet decomposition *(an ℓ, u-fairlet decomposition).*

An intuitive way to look at a fairlet decomposition $\mathcal{F}$ is to view it as a fair clustering of P with $k' = |I|$ clusters (where the clusters have no centers). Before assigning a cost to $\mathcal{F}$, we assign a center $c_i \in L$ to each fairlet $F_i \in \mathcal{F}$. We then assign a cost to $\mathcal{F}$ in the following way

$$cost(\mathcal{F}) = \begin{cases} \max_{i\in I} \max_{j\in F_i} d(j, c_i) & k\text{-center}/k\text{-supplier problem} \\ \sum_{i\in I} \sum_{j\in F_i} d(j, c_i) & \text{facility location, } k\text{-median, } k\text{-means problem.} \end{cases}$$

We call the problem to find a fairlet decomposition of minimal cost the *fairlet decomposition problem*. This assignment of costs is natural: The cost of $\mathcal{F}$ is exactly the cost of the corresponding k'-clustering. Notice that assigning a center $c_i \in L$ to each fairlet is part of the fairlet decomposition problem. It follows that the cost $\mathsf{opt}_{\mathsf{fd}}$ of an optimum fairlet decomposition – i.e., the cost of an optimum fair clustering of P with at most $k' \geq k$ centers – cannot be larger than the cost opt of an optimum fair clustering with k centers. In other words, we have $\mathsf{opt}_{\mathsf{fd}} \leq \mathsf{opt}$ for the fair k-center/k-supplier problem, the fair k-median problem, the fair k-means problem, and the fair facility location problem.

The general fairlet approach for different problems To see why fairlets help us to compute fair clusterings let us first consider a simple example. Suppose we are looking for an optimum fair k-center clustering of a point set P and denote the cost of this clustering by opt. Also suppose that we know an optimum fairlet decomposition $\mathcal{F} := \{F_i \mid i \in I\}$ of P with cost $\mathsf{opt}_{\mathsf{fd}}$ that assigns a center c_i to each $F_i \in \mathcal{F}$. How can we use this knowledge to find a good clustering of P? Since the union of disjoint fairlets yields a fair set of points, it seems a good idea to contract each fairlet to a single point and to then compute a colorblind clustering. This idea works indeed and was first proposed in [29]: We interpret c_i as a representative of F and compute a β-approximate colorblind k-center clustering $\mathcal{C}' = (S', \phi')$ of the representatives $P' = \{c_i \mid i \in I\}$. Then, we merge the fairlets, whose representatives ϕ assigns to the same center, into a fair cluster: We obtain a fair clustering $\mathcal{C}$ by assigning the points $j \in F_i$ of each fairlet $F_i \in \mathcal{F}$ to $\phi'(c_i)$. As a union of disjoint fairlets each cluster of $\mathcal{C}$ must be fair. Also, by our previous observation that $\mathsf{opt}_{\mathsf{fd}} \leq \mathsf{opt}$, each fairlet must have a radius of at most opt. Likewise, we can bound costs of the colorblind clustering $\mathcal{C}'$ by opt: An optimum colorblind clustering cannot be more expensive than an optimum fair clustering. Thus, the radius of each cluster in $\mathcal{C}$ can at most be $(1 + \beta) \cdot \mathsf{opt}$. Hence we have found a $(1 + \beta)$-approximation!

In the remainder of this section, we explore what kind of approximation ratios we can expect when we cluster fairlets with a colorblind algorithm. To do so, we formalize the approach and generalize it to the k-supplier, the k-median, the facility location and the k-means problem where we will need more advanced techniques for the analysis.

Using Fairlets to obtain Clusterings

As before, we look for clusterings of a point set P and denote the cost of an optimum (exact or relaxed) fair k-clustering by opt (we rely on the context to make clear whether this means k-center, k-median, facility location or k-means). We let $\mathsf{opt}_{\mathsf{fd}}$ denote the cost of an optimum decomposition of P into fairlets and we denote the cost of an optimum colorblind k-clustering of P by $\mathsf{opt}_{\mathsf{cb}}$. Since any fair k-clustering yields a valid fairlet decomposition we have $\mathsf{opt}_{\mathsf{fd}} \leq \mathsf{opt}$. Likewise, any fair k-clustering is a valid colorblind k-clustering and $\mathsf{opt}_{\mathsf{cb}} \leq \mathsf{opt}$ follows as well; just as in the above example.

In the sequel, we suppose that we can compute an α-approximate fairlet decomposition $\mathcal{F} = \{F_i \mid i \in I\}$ of P, i.e., we suppose that $\mathsf{cost}(\mathcal{F}) \leq \alpha\,\mathsf{opt}_{\mathsf{fd}} \leq \alpha\,\mathsf{opt}$. We also suppose that we can compute a β-approximate colorblind clustering $\mathcal{C}'$, i.e., we ask that $\mathsf{cost}(\mathcal{C}') \leq \beta\,\mathsf{opt}_{\mathsf{cb}} \leq \beta\,\mathsf{opt}$. The current state-of-the-art does not provide non-trivial approximation algorithms in all cases, but we postpone this issue until the end of this section. For now, let us assume that we have the necessary black-box algorithms and explore the resulting guarantees of the fairlet approach.

k-center. We repeat the argument from the introductory example: Given an α-approximate fairlet decomposition of P that assigns a center $c_i \in P$ to each F_i in $\mathcal{F}$, we compute a β-approximate colorblind k-center solution $\mathcal{C}' = (S', \phi')$ of $P' := \{c_i \mid i \in I\}$. We obtain a fair clustering $\mathcal{C} = (S', \phi'')$ of P by setting $\phi''(j) = \phi'(c_i)$ for all $j \in F_i$. We can now bound the distance of any point $j \in F_i$ to its center $\phi'(c_i)$ in $\mathcal{C}$ as $d(j, \phi'(c_i)) \leq d(j, c_i) + d(c_i, \phi'(c_i))$ since d is a metric. Yet, we have $d(j, c_i) \leq \alpha\,\mathsf{opt}_{\mathsf{fd}} \leq \alpha\,\mathsf{opt}$ and $d(c_i, \phi'(c_i)) \leq \beta\,\mathsf{opt}_{\mathsf{cb}} \leq \beta\,\mathsf{opt}$. It follows that $\mathsf{cost}(\mathcal{C}) \leq (\alpha + \beta)\,\mathsf{opt}$ and our algorithm is a $(\alpha + \beta)$-approximation algorithm.

k-supplier. Given an α-approximate fairlet decomposition of P that assigns a center c_i to each F_i in $\mathcal{F}$, we cannot just compute a β-approximate colorblind k-center solution $\mathcal{C}' = (S', \phi')$ of $P' := \{c_i \mid i \in I\}$ as we might have $P' \subsetneq P$, which would make difficult to compare the cost of a solution on P' to the cost of an optimal solution on P. Instead of c_i we choose an arbitrary point $j_i \in F_i$ as a representative for F_i and compute a β-approximate colorblind k-center solution $\mathcal{C}'$ of $P' := \{j_i \mid i \in I\}$. We obtain a fair clustering $\mathcal{C}$ of P by assigning all $j \in F_i$ to $\phi'(j_i)$. We can now bound the distance of any point $j \in F_i$ to its center $\phi'(j_i)$ in $\mathcal{C}$ as $d(j, \phi'(j_i)) \leq d(j, j_i) + d(j_i, \phi'(j_i))$ since d is a metric. Yet, we have $d(j, j_i) \leq 2\alpha\,\mathsf{opt}_{\mathsf{fd}} \leq 2\alpha\,\mathsf{opt}$ and $d(j_i, \phi'(j_i)) \leq \beta\,\mathsf{opt}_{\mathsf{cb}} \leq \beta\,\mathsf{opt}$. It follows that $\mathsf{cost}(\mathcal{C}) \leq (2\alpha + \beta)\,\mathsf{opt}$ and our algorithm is a $(2\alpha + \beta)$-approximation algorithm. Some of the algorithms for the fairlet decomposition problem will already only assign one of the points in each fairlet as its center. In that case it follows that $\mathsf{cost}(\mathcal{C}) \leq (\alpha + \beta)\,\mathsf{opt}$ and our algorithm is a $(\alpha + \beta)$-approximation algorithm.

k-median. The k-median case is more difficult: Computing representatives and repeating the above analysis yields an additive error of $\alpha\,\mathsf{opt}$ *for each point*, and thus at best a $2 \cdot \alpha \cdot |P|$-approximation. A different technique is needed here.

We start by computing a β-approximate colorblind clustering $\mathcal{C}' = (S', \phi')$ on P. For all $i \in I$ and purely for the analysis we choose the center c_i of F_i in $\mathcal{F}$ as a representative of F_i. We then look for the point

$$j_i := \arg\min_{j \in F_i} \Big[d(c_i, j) + d(j, \phi'(j)) \Big]$$

that among all $j \in F_i$ minimizes the distance from c_i to $\phi'(j)$ via j. We assign all points $j \in F_i$ to this center $\phi'(j_i)$. In this way, we obtain a fair clustering $\mathcal{C}$. Let us first compute an upper bound on the cost of assigning a single point $j \in F_i$ to its center $\phi'(j_i)$ in $\mathcal{C}$. We have

$$d(j, \phi'(j_i)) \leq d(j, c_i) + d(c_i, j_i) + d(j_i, \phi'(j_i)).$$

By our choice of j_i, we have $d(c_i, j_i) + d(j_i, \phi'(j_i)) \leq d(c_i, j) + d(j, \phi'(j))$ and it follows that

$$\begin{aligned}
\mathsf{cost}(\mathcal{C}) &= \sum_{i \in I} \sum_{j \in F_i} d(j, \phi'(j_i)) \\
&\leq \sum_{i \in I} \sum_{j \in F_i} \Big(2d(c_i, j) + d(j, \phi'(j)) \Big) \\
&= 2 \cdot \sum_{i \in I} \sum_{j \in F_i} \Big(d(j, c_i) + \sum_{j \in P} d(j, \phi'(j)) \Big) \\
&\leq 2 \cdot \alpha \, \mathsf{opt}_{\mathsf{fd}} + \beta \, \mathsf{opt}_{\mathsf{cb}} .
\end{aligned}$$

Thus, our algorithm yields a $(2\alpha + \beta)$-approximation.

Facility location. Observe that in the above algorithm, we only open the centers from the colorblind solution; the fairlet decomposition does not incur facility opening costs. By the same reasoning as before we get

$$\begin{aligned}
\mathsf{cost}(\mathcal{C}) &= \sum_{i \in I} \sum_{j \in F_i} d(j, c(j_i)) + \sum_{l \in S_{\mathrm{cb}}} f_l \\
&\leq 2 \cdot \sum_{i \in I} \sum_{j \in F_i} \Big(d(j, c_i) + \sum_{j \in P} d(j, c(j)) + \sum_{l \in S_{\mathrm{cb}}} f_l \\
&\leq 2 \cdot \alpha \, \mathsf{opt}_{\mathsf{fd}} + \beta \, \mathsf{opt}_{\mathsf{cb}} .
\end{aligned}$$

where S_{cb} denotes the set of facilities opened by the colorblind facility location solution. We obtain a $(2\alpha + \beta)$-approximation algorithm in this case as well.

k-means. For the k-means problem, we assume that $P \subseteq \mathbb{R}^m$ and that d is the squared Euclidean distance measure. For each fairlet F_i let $\mu(F_i)$ be its centroid, i.e., $\mu(F_i) = \frac{1}{|F_i|} \sum_{j \in F_i} j$ and let us look at a colorblind β-approximate clustering $\mathcal{C}' = (S', \phi')$ on the instance which contains $|F_i|$ copies of $\mu(F_i)$ for each $i \in I$. Without loss of generality, we assume that for all $i \in I$ all of the copies of $\mu(F_i)$ will be assigned to the same center as otherwise assigning all copies of $\mu(F_i)$ to the nearest opened center would only decrease the cost. We then obtain $\mathcal{C}$ from $\mathcal{C}'$ by assigning all points in F_i to the center c_{F_i} to which the copies of $\mu(F_i)$ were assigned in $\mathcal{C}'$. The following extremely useful observation, which is usually considered folklore, helps us to determine the cost *cost*$(\mathcal{C})$ of this assignment.

Observation 32. *Given a point set $P \subseteq \mathbb{R}^m$ and a point $c \in \mathbb{R}^m$, then the 1-means cost of assigning P to c can be decomposed into*

$$\sum_{j \in P} \|j - c\|^2 = \sum_{j \in P} \|j - \mu\|^2 + |P| \cdot \|\mu - c\|^2$$

where $\mu = \frac{1}{|P|}\sum_{j\in P} j$ *is known as the centroid of* P.

Observation 32 implies that the cost to assign all points $j \in F_i$ to c_{F_i} is equal to

$$\begin{aligned}\mathsf{cost}(\mathcal{C}) &= \sum_{i\in I}\sum_{j\in F_i}\|j - c_{F_i}\|^2\\ &= \sum_{i\in I}\Big[\sum_{j\in F_i}\|j-\mu(F_i)]\|^2 + |F_i|\cdot\|\mu(F_i) - c_{F_i}\|^2\Big]\\ &= \underbrace{\sum_{i\in I}\sum_{j\in F_i}\|j-\mu(F_i)\|^2}_{\le\mathsf{cost}(\mathcal{F})} + \underbrace{\sum_{i\in I}|F_i|\cdot\|\mu(F_i) - c_{F_i}\|^2}_{\le\mathsf{cost}(\mathcal{C}')}\,.\end{aligned}$$

In other words, the cost of $\mathcal{C}$ is at most the joint cost of $\mathcal{F}$ and $\mathcal{C}'$. Since we know that $cost(\mathcal{F})$ is bounded by $\alpha\,\mathsf{opt}$ it remains to show an upper bound on $cost(\mathcal{C}')$, the cost of the clustering on the centroids. To that aim, let us consider an optimum colorblind clustering $\mathcal{C}_{\mathsf{cb}} = (S_{\mathsf{cb}}, \phi_{\mathsf{cb}})$ of P. For any fixed $j \in F_i$ assigning all copies of $\mu(F_i)$ to $\phi_{\mathsf{cb}}(j)$ yields a cost of

$$|F_i|\cdot\|\mu(F_i) - \phi_{\mathsf{cb}}(i)\|^2 \le |F_i|\cdot(2\|\mu(F_i) - j\|^2 + 2\|j - \phi_{\mathsf{cb}}(j)\|^2)$$

by the 2-relaxed triangle inequality. In each fairlet F_i, we now look at those points whose distance to $\mu(F_i)$ is at most the median of these distances across F_i. Formally, we define for each $F_i \in \mathcal{F}$:

$$N_i := \left\{ j \in F_i \;\middle|\; \|j-\mu(F_i)\|^2 \le \|j' - \mu(F_i))\|^2 \text{ for at least } \left\lfloor\frac{|F_i|}{2}\right\rfloor \text{ points } j' \ne j \in F_i \right\}$$

and let $j_i := \arg\min_{j\in N_i}\|j - \phi_{\mathsf{cb}}(j)\|^2$ be a point with minimum distance to its center in S_{cb}, among all points in N_i. With that we obtain $|F_i|\cdot\|\mu(F_i) - j_i)\|^2 \le 2\sum_{j\in F_i}\|\mu(F_i) - j\|^2$ and $|F_i|\cdot\|j_i - \phi'(j_i)\|^2 \le 2\sum_{j\in F_i}\|j - \phi_{\mathsf{cb}}(j)\|^2$ for all $i \in I$. This implies that there exists a clustering on the centroids with a cost of at most

$$\sum_{i\in I}\Big(4\cdot\sum_{j\in F_i}\|\mu(F_i) - j\|^2\Big) + \sum_{i\in I}\Big(4\cdot\sum_{j\in F_i}\|j - \phi'(j)\|^2\Big).$$

The first sum is at most 4 times the cost of $\mathcal{F}$ and therefore bounded by $4\alpha\,\mathsf{opt}$. The second sum is at most 4 times the cost of an optimal (colorblind) clustering and is therefore bounded by $4\,\mathsf{opt}$. This shows that there exists a clustering on the centroids with a cost of at most $4(\alpha+1)\,\mathsf{opt}$. The β-approximation therefore has a total cost of at most $\beta\cdot 4(\alpha+1)\,\mathsf{opt}$ and the computed clustering on P has cost of at most $(4\beta(\alpha+1)+\alpha)\,\mathsf{opt}$. Hence, it is a $(4\beta(\alpha+1)+\alpha)$-approximation.

Computing Fairlet Decompositions

Having reviewed fairlet based black-box approximation algorithms for the different clustering objectives in the previous section, we now turn to approaches and problems of computing fairlet decompositions. What concrete approximation ratios do we obtain once we fill in the black-box algorithms with the state-of-the-art approximation algorithms for the fairlet decomposition problem?

The happy world of k-center For the happy world of k-center we review known results, or at least algorithms that achieve the same guarantee as the known results. The colorblind variant of the k-center problem has a 2-approximation algorithm [47, 53] and thus, it remains to find an α-approximate fairlet decomposition: Together with the results from the previous section, we then obtain a $(2+\alpha)$-approximation algorithm for the fair k-center problem. The prominent feature of the k-center problem is the *threshold graph.* Observe that for k-center, the optimum *cost* is a pairwise distance: It is the maximum distance between a center (which is an input point itself) and the furthest away point assigned to it. Thus, we can guess the optimum cost by iterating through all $\Theta(n^2)$ pairwise distances. 'Guessing' means that we try the $\Theta(n^2)$ values, assume in each run that our guess is the optimum and compute a solution based on this assumption. Depending on the actual algorithm, some runs may fail (indicating that we had the wrong value), and the other runs will give solutions of different quality; we can then pick the best solution, which can only be better than the solution of the run where we indeed had guessed the value correctly. This trick is used in many papers on k-center. The usual way to make use of the guessed optimum value – which we name τ – is to build a threshold graph G_τ where points are connected by an edge, if they are at distance $\leq \tau$ and then observe that two points can only be in the same optimum cluster if they are connected by a path of length two in G_τ, i.e., if they are connected in the 2-hop graph G_τ^2 of G_τ.

Two colors. Now let's start with the case that is studied in [29], i.e., we assume that $|Col| = 2$, say $Col = \{red, blue\}$. Instead of the ratios of each color Chierichetti et al. define the *balance* of a set Q as

$$\text{balance}(Q) = \min\left\{\frac{|col_{red}(Q)|}{|col_{blue}(Q)|}, \frac{|col_{blue}(Q)|}{|col_{red}(Q)|}\right\} \in [0,1],$$

and the balance of a clustering as the smallest balance of any cluster in it. Now the main question is what balance we want to achieve. In [29], two cases are distinguished: a) the input P has balance 1, and we want to compute a clustering which also has exactly balance 1 and b) we want a clustering with balance at least $1/t$, but we do not know anything about the balance of P. If balance$(P) < 1/t$, there is no feasible solution, and if balance$(P) > 1/t$, then we allow additional imbalance.

When considering the difficulty of the approximation problem, it makes sense to distinguish different cases. For the purpose of later referencing, we state them in the following definition.

Definition 33. *For the two color case, we distinguish the following cases of a fair clustering. These are the cases where we want a clustering*

1. *with exact fairness, and* balance$(P) = 1$*,*
2. *with exact fairness, and* balance$(P) = 1/t$ *for an integer* t*,*
3. *with exact fairness, and* balance$(P) = s/t$ *for integers* $s \leq t$*,*
4. *with balance* $b \leq$ *balance*(P)*, and* $b = 1/t$ *for an integer* t*, or*

5. *with balance $b \leq balance(P)$ for any $b \in [0,1]$.*

In cases 4 and 5, the target balance b may in particular be smaller than balance(P). Notice that for any finite point set, balance(P) $= s/t$ for some integers s, t, so the real restriction in case 3 is that we want to match balance(P) exactly. The cases 1, 2, 3 involve *exact* fair clustering. Notice that these models force us to exactly match the proportion of red and blue points of P. Under this condition, it might not even be possible to find k clusters; thus, the optimal solution may well just consist of a single cluster which contains every point. This trivial clustering, however, is always a feasible solution, so the problem is well-defined.

A first algorithm for case 1. In case 1, a good fairlet decomposition is particularly easy to obtain: Chierichetti et al. [29] show that it consists of a perfect matching between the red and the blue points. Assume that $\mathcal{C}_{\mathsf{opt}} = (S_{\mathsf{opt}}, \phi_{\mathsf{opt}})$ is the optimal fair clustering of P with radius r. Then in particular for each $i \in S_{\mathsf{opt}}$ the cluster $P(i)$ consists of an equal number of red and blue points. Thus, there exists a bipartite matching between the red and blue points where both endpoints of a matching edge are always in the same optimum cluster.

Chierichetti et al. [29] tried to recover a good fairlet decomposition by computing a perfect matching, in a bipartite graph between the red and the blue points. They used the threshold idea and only connect a red point to a blue point if their pairwise distance is at most the given threshold τ'. Let $G_{rb,\tau'}$ denote the bipartite threshold graph that connects blue points and red points with a pairwise distance of at most τ. They guess the optimum cost τ as described above from one of the pairwise distances. As any two points in the same optimum cluster must share a location at distance at most opt to both of them, their distance is at most $2\,\mathsf{opt}$ and there must be a threshold $\tau \leq 2\,\mathsf{opt}$ such that $G_{rb,\tau}$ contains a perfect matching. If no perfect matching exists in $G_{rb,\tau}$, then $\tau \leq \mathsf{opt}/2$ follows. It suffices to find the smallest τ for which $G_{rb,\tau}$ contains a perfect matching. Let M be a perfect matching in $G_{rb,\tau}$. For every edge $e_i \in M$ let r_i and b_i denote the red and the blue point e_i connects. For each edge $e_i \in M$ let the fairlet $F = \{r_i, b_i\}$ with center r_i be contained in the fairlet decomposition corresponding to M. Since e_i is part of $G_{rb,\tau}$, the distance between r_i and b_i – and thus the radius of F – is at most τ. Testing all reasonable values for τ and taking the best fairlet decomposition yields a 2-approximation since its cost is bounded by $2\,\mathsf{opt}$. In total this approach yields a 4-approximation algorithm for the fair k-center problem. Chierichetti et al. [29] claim a 3-approximation algorithm for k-center. We will explain in Appendix C the small mistake made in their analysis and give an example were the analysis does not hold. We use a different way to compute the cost of a fairlet, where we allow every point to be chosen as the center of a fairlet. We can then make a small adjustment to the algorithm and obtain a 3-approximation algorithm for the special case of the two color fair k-center problem, where each color appears equally often.

An improved algorithm for case 1. Again, partition the points into pairs of one red and one blue point from the same optimal cluster. Let r, b be such a pair. Observe that then there is a point c (a center in the optimum solution) such that $d(r,c) \leq \mathsf{opt}$ and $d(b,c) \leq \mathsf{opt}$. Thus, instead of looking at the pairwise distances between all red and blue points, we do something else. For each pair r, b, we compute the point $x =$

$x(r,b) = \arg\min_{j\in P} \max\{d(r,j), d(b,j)\}$. Also, we set $c(r,b) = \max\{d(r,x), d(b,x)\}$. Then we know that if r and b are in the same optimum clustering, then $c(r,b) \le \mathsf{opt}$. We build the slightly different threshold graph G'_τ, where we use c instead of d as a distance measure and include an edge for a pair r, b, if $c(r,b) \le \tau$. A perfect matching in this graph exists if $\tau \ge \mathsf{opt}$. Thus, we search for the smallest τ such that we find a perfect matching in G'_τ and then know that in this matching, every point is at distance $\le \tau$ from his $x(r,b)$. We again create a fairlet $F = \{r, b\}$ for each edge $e = \{r,b\}$ in the matching; this time, however, we assign $x(r,b)$ as the center of F in the decomposition. Since now every fairlet has a radius of at most $\tau = \mathsf{opt}$, the decomposition is exact and we obtain a 3-approximation overall. We can apply the same approach to the fair k-supplier problem when for each pair r, b, we compute the point $x = x(r,b) = \arg\min_{x\in L} \max\{d(r,x), d(b,x)\}$ and set $c(r,b) = \max\{d(r,x), d(b,x)\}$. Since the computed decomposition is then exact and there exist 3-approximation algorithms for the standard k-supplier problem [53], we obtain a 5-approximation algorithm.

Cases 2 and 4. In the happy k-center world, cases 2 and 4 of Definition 33 can both be 4-approximated: Chierichetti et al. [29] give a 4-approximation algorithm by using a minimum cost flow. We describe a slightly simpler algorithm which assumes that we know the majority color (or compute it as a first step). Without loss of generality, assume that blue is the majority color, i.e., at least half of the points are blue.

The difference to case 1 is that the input can no longer be broken into *pairs* of red and blue points. However, we still know that the points can be partitioned into groups of one red and $\le t$ blue points. More precisely, we know: In any clustering satisfying the fairness constraint, a cluster C with $b(C)$ blue points must have at least $\lceil b(C)/t \rceil$ red points. Thus, we can take the optimum solution and partition each optimum cluster in subgroups of one red point together with $\le t$ blue points. Now these subgroups are exactly the fairlets we want to find.

To obtain the partitioning, we set up a flow network built upon the threshold graph. We start by adding all red points as well as all blue points as nodes. We add threshold edges between red and blue points in the same manner as in the easy algorithm: For every pair r, b, we add an edge if $d(r,b) \le 2\tau$ (i.e., if they are connected in G^2_τ). This edge gets a capacity of one. Next, we add a source and a sink. The source is connected to all red points with an edge of capacity t: This is the maximum number of points it can facilitate. The blue points are connected to the sink by an edge of capacity 1. We then compute a maximum flow in this graph. Let n_b be the number of blue points. If the flow has a value less than n_b, we know that τ is wrong: For $\tau \ge \mathsf{opt}$, we know by the above argumentation that we can group the points accordingly and define a flow where every blue point is facilitated by some red point. So we compute the smallest τ for which the maximum flow has value n_b (it cannot be higher since that is the capacity of the edges going into the sink). Using the red points as centers, this step gives us fairlets with radius $\le 2\,\mathsf{opt}$ and we obtain a 4-approximation algorithm. For k-supplier we can again use the same approach. As we already use some of the points as centers we obtain a 5-approximation algorithm.

Case 3. If balance$(P) = s/t$, then the above maximum flow idea fails since we lose the anchor node that collects the points of each fairlet. However, in the happy world of

k-center, we can use capacitated clustering to fill the gap. The general idea is simple (use capacitated clustering to find fairlets), however, the execution is trickier than one might expect. The following is a summary of the approximation algorithm given in Section 4.2 in [90]. The algorithm in [90] works for multiple colors, but in this section, we only discuss two color variants.

Again, we assume that blue is the majority color, which means that $|col_{red}(P)| = n_f \cdot s$ and $|col_{blue}(P)| = n_f \cdot t$ for some integers n_f, s, t with $gcd(s, t) = 1$. We consider an optimal solution. Then each cluster in this solution has $j \cdot s$ red and $j \cdot t$ blue points for some $j \in \mathbb{N}$, and we can group it into fairlets with s red and t blue points. In particular, this grouping induces a uniform capacitated clustering of the red points into n_f clusters with capacity s, and a uniform capacitated clustering of the blue points into n_f clusters with capacity t, and the radius of both these clusterings is at most opt.

There is one subtlety, though: the centers of these capacitated clusterings are not necessarily of the same color. In the optimum fair clustering, the center of each cluster may be different, sometimes red, sometimes blue. This means that the capacitated clustering problems that are induced by the optimum solution are not uniform capacitated *k-center* solutions, but they are uniform capacitated *k-supplier* solutions:

Thus, we know that the optimum solution to the following uniform capacitated k-supplier problem costs at most opt: Let P be the red points, let L be the union of red and blue points, and set the capacity of every center to s. We will use a subroutine for this problem. In the following, we will be okay if the clustering uses soft capacities, i.e., it may open centers multiple times. A solution for this problem can only be cheaper, and we won't be using the centers as centers in the final solution anyway, so the relaxation does not hurt us.

Khuller and Sussmann [66] give a 5-approximation algorithm for the uniform soft capacitated k-center problem. Unfortunately, we need the k-supplier version. Taking a k-center solution for a k-supplier version can at most double the approximation ratio; thus, we use Khuller and Sussmann's algorithm, but it returns a 10-approximation for us. The smarter way would be to *adapt* Khuller and Sussmann's algorithm to work for the k-supplier variant. We conjecture that this might return a 7-approximation instead of the 10.

We apply the algorithm to the red points and obtain clusters of size s. Now we need to assign t blue points to each red cluster. This, however, can be done by computing a matching. We construct a bipartite graph. On the red side, it contains t copies of each center, i.e., in total, it contains $t \cdot n_f = b(P)$ red nodes. The blue side just consists of the blue points. For the edges, we again use thresholding and select our threshold τ as one of the distances between a red and a blue point. For a threshold τ, we add an edge between a blue point b and a center c if c represents at least one red point r with $d(b, r) \leq 2\tau$. Now we compute the smallest τ for which the resulting graph has a perfect matching. This matching assigns one blue point to every center-copy, resulting in t blue points assigned to every red cluster.

What's the quality of the resulting solution? We know that in the optimum solution, there is some clustering of the red points which is coupled with a clustering of the blue points. Now if we had the correct red clustering, the distance between a blue point and all red points in the same optimum cluster would be at most $2 \cdot \mathsf{opt}$, leading to an overall approximation ratio of 12. We can still get a similar statement by using Hall's theorem. Let B be any subset of the red points. Since we made t copies for every red cluster B has to contain points out of at least $\lceil \frac{|B|}{t} \rceil$ such clusters which represent at least $s\lceil \frac{|B|}{t} \rceil$ red points. Therefore any optimal fairlet decomposition contains at least $\lceil \frac{|B|}{t} \rceil$ many fairlets which contain at least one of the points represented by B. These fairlets in the optimal decomposition then contain at least $t\lceil \frac{|B|}{t} \rceil \geq |B|$ many blue points. Each of these blue points then has a distance of at most $2\,\mathsf{opt}$ to the represented red point and therefore by the triangle inequality a distance of at most $12\,\mathsf{opt}$ to the represented center. Therefore if we choose our threshold $\tau = \mathsf{opt}$ we obtain a Graph in which the neighborhood of any subset B of the red nodes contains at least $|B|$ neighbors. Hall's theorem then shows that this graph must contain a perfect matching.

This finally gives us fairlets with $s + t$ points and a corresponding center. The radius of each fairlet is at most $12 \cdot \mathsf{opt}$ and we obtain a 14-approximation in total. For the k-supplier version we obtain a 15-approximation.

Case 5. Even in the happy k-center world no constant-factor approximation algorithm is known for case 5; in particular, we do not know how to find a good fairlet decomposition. The problem here is that we do not even know that the optimal clustering can be partitioned into small balanced subsets.

Multiple colors Before going through the different cases let us make an observation for exact fairness.

Observation 34. *Let $m = gcd(|col_h(P)| \mid h \in Col) \in \mathbb{N}$ be the greatest common denominator of the different numbers of points of the different colors. Let $b_h = \frac{|col_h(P)|}{m}$ for each $h \in Col$. Then for each cluster $P(i)$ in a fair clustering $\mathcal{C}$ of P with exact preservation of ratios, there exists a positive integer $i' \in \mathbb{N}_{\geq 1}$ such that $P(i)$ contains exactly $i' \cdot b_h$ points with color h for each color $h \in Col$ and $i' \cdot \frac{n}{m}$ total points. Thus every cluster must have at least b_h points of color h for each color $h \in Col$.*

Let $h_1 \in Col$ denote one of the rarest colors, i.e., $|col_{h_1}(P)| \leq |col_h(P)|$ for all $h \in Col$. Case 2 with two colors and $balance(P) = 1/t$ for an integer t can be generalized to an arbitrary number of colors with $\frac{|col_h(P)|}{|col_{h_1}(P)|} \in \mathbb{N}$ for all $h \in Col$. By Observation 34 we know that the points in each optimal cluster can be partitioned into groups of one point with color h_1 and $\frac{|col_h(P)|}{|col_{h_1}(P)|}$ points with color h. The idea is to assign points with color $h \in Col \setminus \{h_1\}$, independently of points with a different color, to the points with color h_1. In the end all points connected to the same point in $col_{h_1}(P)$ build a fairlet. To do so we set up a flow network analogously to case 2: It is built upon the threshold graph for each color $h \in Col \setminus \{h_1\}$ which contains all points with colors h_1 and h. They then choose the smallest threshold for which the corresponding networks for all colors $h \in Col \setminus \{h_1\}$ are successful. This again results in a fairlet decomposition with radius $\leq 2\,\mathsf{opt}$ and a 4-approximation algorithm for the fair k-center problem as well as a 5-approximation algorithm for the fair k-supplier variant.

The case 3 can similarly be generalized to instances with arbitrary many colors [90]. Let $b_h = \frac{|col_h(P)|}{gcd(|col_{h'}(P)||h' \in Col)}$ for each $h \in Col$ as in Observation 34, then we know that the points in each optimal cluster can be partitioned into groups of b_h points with color h for every color $h \in Col$. Then again the approach is to compute a clustering on the points with color h_1 in which every cluster contains exactly b_{h_1} points. Dealing with the other colors independently of each other, the same approach as in case 3 can be used to match b_h points with color h to each of these sets of b_{h_1} points with color h_1. Again, with the 5-approximation algorithm for the capacitated k-center problem [66] this creates a fairlet decomposition with cost at most $12\,\mathsf{opt}$ and results in 14 and 15-approximation algorithms for the fair k-center and the fair k-supplier problem.

Theoretically, case 4 can also be generalized to a case where we require that $\frac{|col_{h_1}(P')|}{|col_h(P')|} \geq 1/t_h$ for some integer t_h for all $h \in Col$. This way every cluster in the optimal solution can be partitioned into fairlets out of which each contains exactly one point with color h_1 and at most t_h points of color h. The generalization of the approach is then identical with the generalization of case 2. However, this generalization cannot be described through our ratio based problem definition with l,u-balanced clusters, as in the ratio based problem definition the points with different colors cannot be treated independently of each other.

Fairlet decompositions for k-median, facility location and the k-means problem We now turn to computing fairlet decompositions for clustering objectives beside k-center. We assume that we want to compute exact fairlets (i.e., not l,u-fairlets) in this section.

Instances with two colors Let us first consider the simple case where we only have two colors, i.e., $Col = \{red, blue\}$. In that case we have balance(P) $= r/b$ for two integers $r,b \in \mathbb{N}$ and $\gcd(r,b) = 1$.

The case $r = b = 1$. If the balance is 1, then each fairlet consists of exactly one blue and one red point. We can therefore model the fairlet decomposition problem as a matching problem as before: We construct a complete bipartite graph in which the red and blue points make up one partition, respectively. The cost of an edge between a red point p and a blue point q is the distance between p and q. For k-median/facility location, this cost is equal to assigning both p and q to p. Assigning p and q to any other center cannot yield lower costs by the triangle inequality. In the k-means case, we know that the best possible center for p and q is the centroid $\mu(\{p,q\}) = \frac{p+q}{2}$. We then define the cost of $\{p,q\}$ as the $\|p - \mu(\{p,q\})\|^2 + \|q - \mu(\{p,q\})\|^2$. We then compute a minimum cost perfect matching to obtain a fairlet decomposition; the cost of the matching is the cost of the decomposition. Unfortunately, this is the only case leading to a constant-factor approximation algorithms so far.

The case $r = 1, b > 1$. If the balance is $1/b$ for an integer b, we can solve the fairlet decomposition problem similarly: Analogously to the k-center case, we start with the n_b many blue points and cluster them into sets with exactly b points each. This, again, could be done with an approximation algorithm for the uniform capacitated problem (i.e., k-median and k-means). Unfortunately, we are not aware of any such algorithm. Still, if we set k to n_b/b and the uniform capacity to b, then any β-approximation to the capacitated problem has cost of at most $\beta\,\mathsf{opt}$, as observed before. We can then collect the center of each of the clusters of b blue points, and similarly to the case with $r = b = 1$ compute a perfect matching between the red points and these centers. Hall's theorem then shows that it is possible to assign one red point r to every blue cluster while guaranteeing that at least one of the blue points is in the same cluster with r in the optimal fair solution. For k-median/facility location the cost incurred by r is then at most its cost in the optimal solution, plus the cost incurred by its assigned blue point in the optimal solution, plus the cost incurred by the assigned blue point in the computed approximation. This gives us a total cost of at most $(2\alpha + 1)\,\mathsf{opt}$. For k-means the cost of the red point is then at most four times its cost in the optimal solution, plus four times the cost of its assigned blue point in the optimal solution, plus four times the cost of the assigned blue point in the computed approximation. This yields a total cost of at most $4(2\alpha + 1)\,\mathsf{opt}$. We could also try to make the assignment the other way around and assign b blue points to each red point. This way we would not have to compute a solution of the uniform capacitated problem. An assignment of b blue points to each red point can easily be found with a network flow. For k-median/facility location the cost incurred by each blue point b is then at most its cost in the optimal solution, plus the cost incurred by its assigned red point in the optimal solution. As this counts the cost of every red point b times, this gives us a total cost of at most $b\,\mathsf{opt}$. For k-means the cost of each blue point is at most four times its cost in the optimal solution, plus four times the cost of its assigned red point in the optimal solution. This yields a total cost of at most $4b\,\mathsf{opt}$. So for a constant $b \in \mathbb{N}$ we obtain a constant-factor approximation.

The case $r > 1, b > 1$. We further extend the previous case. Without loss of generality we assume $b \geq r$, i.e., that blue is the majority color. Using an approximation algorithm for capacitated colorblind clustering, we again cluster the blue points into sets of size b and then create r copies of each center. We can then compute a minimum cost perfect matching between the red points and the centers.

Similar to before Hall's theorem shows that there exists an injective assignment of red points to a blue point in the same optimal fair cluster such that exactly r red points are assigned to the blue points in every blue cluster. For k-median/facility location the cost of the red point is then at most their cost in the optimal solution plus the cost of their assigned blue point in the optimal solution plus the cost of the assigned blue point in the computed approximation. This gives us a total cost of at most $(2\alpha + 1)\,\mathsf{opt}$. For k-means the cost of the red point is then again at most four times this term, which makes the total cost at most $4(2\alpha + 1)\,\mathsf{opt}$.

Instances with arbitrary many colors Let $b_h := \frac{|col_h(P)|}{gcd(|col_{h'}(P)||h' \in Col)}$ be the number of points with color h each fairlet should have. We choose one color $h \in Col$ and again start by clustering the points into sets of size b_h. For each other color $h' \in Col \setminus \{h\}$ we then create $b_{h'}$ copies of each center and compute a minimum cost perfect matching between the points with color h' and these centers. We do this for every color $h \in Col$ and then take the best solution that we obtained. Again Hall's theorem shows that for each color $h' \in Col \setminus \{h\}$ there exists an assignment of the points with a color h' to a point with color h in the same optimal fair cluster with the following property: Exactly $b_{h'}$ points with color h' are assigned to the points with color h in every cluster, and the number of points assigned to a point with color h is either $\left\lfloor \frac{n-|col_h(P)|}{|col_h(P)|} \right\rfloor$ or $\left\lceil \frac{n-|col_h(P)|}{|col_h(P)|} \right\rceil$. For k-median/facility location the cost of a point with a color $h' \in Col \setminus \{h\}$ is then at most its cost in the optimal solution, plus the cost of its assigned point with color h in the optimal solution, plus the cost of the assigned point in the computed approximation. Assuming that h is the color whose points have the smallest average cost in the optimal fair clustering this implies that the total cost is at most $(3\alpha + 3)\,\mathsf{opt}$. For k-means the cost of a point with a color $h' \in Col \setminus \{h\}$ is then at most four times its cost in the optimal solution, plus four times the cost of its assigned point with color h in the optimal solution, plus four times the cost of the assigned point in the computed approximation. We again assume that h is the color whose points have the smallest average cost in the optimal fair clustering which implies that the total cost is at most $4(3\alpha + 3)\,\mathsf{opt}$.

As we have seen, when we can quickly compute a good fairlet decomposition, using these fairlets to compute a fair clustering works well for all clustering objectives mentioned above. For several cases we have shown that computing a good fairlet decomposition is a relatively easy task. The problem is that in many other cases we do not know how to compute a good fairlet decomposition. For all mentioned clustering objectives we are mostly limited to the case of exact fairness. And, for k-median, facility location, and k-means, even in the case of exact fairness, we currently only know how to compute a good fairlet decomposition for very limited types of instances. In order to obtain algorithms for more types of fair clustering problems, we will introduce an LP-based approach and start by describing the different fair clustering problems as an integer linear program.

2.2.2 (I)LP Formulations for Fair Clustering Problems

Let $I = (P, L, Col, col, d, f, k, \ell, u)$ be a problem instance for a fair clustering problem, where (P, L, d, f, k) denotes an instance of the unconstrained clustering problem, Col denotes the set of colors, $col : P \to Col$ the coloring of the points, and ℓ and u denote the ratio bounds required for fairness. Notice that for all clustering problems defined in Section 1.2.4, P and L are finite except for k-means. However, for the k-means problem, we can assume that $L = P$ if we accept an additional factor of 2 in the approximation guarantee (Lemma 116 in Appendix B). Thus, we assume in the following that L and P are finite sets. Indeed, we even assume at least $L \subseteq P$ for all problems except k-supplier and facility location. We introduce a binary variable $y_i \in \{0, 1\}$ for all $i \in L$ that decides if i is part of the set S of opened locations in the solution, i.e., $y_i = 1 \Leftrightarrow i \in S$. Similarly, we introduce binary variables $x_{ij} \in \{0, 1\}$ for all $i \in L, j \in P$ with $x_{ij} = 1$ if j is assigned to i, i.e., $\phi(j) = i$. All ILP formulations have the inequalities (2.7) $\sum_{i \in L} x_{ij} = 1\ \forall j \in P$ saying that every point j is assigned to a center, the inequalities (2.8) $x_{ij} \leq y_i\ \forall i \in L, j \in P$ ensuring that if we assign j to i, then i must be open, and the integrality constraints (2.9) $y_i, x_{ij} \in \{0, 1\}\ \forall i \in L, j \in P$.

We may restrict the number of open centers to k with (2.10) $\sum_{i\in L} y_i \le k$. For k-center and k-supplier, the objective is commonly encoded in the constraints of the problem, and the (I)LP has no objective function. The idea is to guess the optimum value τ. Since there is only a polynomial number of choices for τ, this is easily done. Given τ, we construct a *threshold graph* $G_\tau = (P \cup L, E_\tau)$ on the points and locations, where a connection between $i \in L$ and $j \in P$ is added if i and j are close, i.e., $\{i,j\} \in E_\tau \Leftrightarrow d(i,j) \le \tau$. Then, we ensure that points are not assigned to centers outside their range:

$$x_{ij} = 0 \qquad \text{for all } i \in L, j \in P, \{i,j\} \notin E_\tau \tag{2.11}$$

For the remaining clustering problems, we pick the adequate objective function from the following two (let $d_{ij} := d(i,j)$):

$$\min \sum_{i\in L, j\in P} x_{ij} d_{ij} \quad (2.12) \qquad \min \sum_{i\in L, j\in P} x_{ij} d_{ij} + \sum_{i\in L} y_i f_i \quad (2.13)$$

We now have all necessary constraints and objectives. For k-center and k-supplier, we use inequalities (2.7)-(2.11), no objective, and define the optimum to be the smallest τ for which the ILP has a solution. We get k-median and k-means by combining inequalities (2.7)-(2.10) with (2.12) and we get facility location by combining (2.7)-(2.9) with the objective (2.13). LP relaxations arise from all ILP formulations by replacing (2.9) by $y_i, x_{ij} \in [0,1]$ for all $i \in L, j \in P$. To create the fair variants of the ILP formulations, we add fairness constraints modeling the upper and lower bound on the balances.

$$\ell_h \sum_{j\in P} x_{ij} \le \sum_{j \in col_h(P)} x_{ij} \le u_h \sum_{j\in P} x_{ij} \qquad \text{for all } i \in L, h \in Col \tag{2.14}$$

Although very similar to the canonical clustering LPs, the resulting LPs become much harder to round even for k-center with two colors. We show the following in Section 2.2.6.

Lemma 35. *There is a choice of non-trivial fairness intervals such that the integrality gap of the LP-relaxation of the canonical fair clustering ILP is $\Omega(n)$ for the fair k-center/k-supplier/k-median/facility location problem. The integrality gap is $\Omega(n^2)$ for the fair k-means problem.*

Essential fairness. For a point set P', $\text{mass}_h(P') = |col_h(P')|$ is the *mass* of color h in P'. For a possibly fractional LP solution (x,y), we extend this notion to $\text{mass}_h(x,i) := \sum_{j\in col_h(P)} x_{ij}$. We denote the total mass assigned to i in (x,y) by $\text{mass}(x,i) = \sum_{j\in P} x_{ij}$. With this notation, we can now formalize our notion of *essential fairness.*

Definition 36 (Essential fairness)**.** *Let I be an instance of a fair clustering problem and let (x,y) be an integral, but not necessarily fair solution to I. We say that (x,y) is* essentially fair *if there exists a fractional fair solution (x',y') for I such that $\forall i \in L$:*

$$\lfloor \text{mass}_h(x',i) \rfloor \le \text{mass}_h(x,i) \le \lceil \text{mass}_h(x',i) \rceil \qquad \forall h \in Col \tag{2.15}$$

$$\textit{and} \quad \lfloor \text{mass}(x',i) \rfloor \le \text{mass}(x,i) \le \lceil \text{mass}(x',i) \rceil. \tag{2.16}$$

2.2.3 Black-Box Algorithms for Essentially Fair Clustering

For essentially fair clustering, we give a powerful framework that employs approximation algorithms for (unfair) clustering problems as a black-box and transforms their output into an essentially fair solution. In this framework, we start by computing an approximation for the standard variant of the clustering problem at hand. Next, we solve the LP for the fair variant of the clustering problem. Now we have an integral unfair solution, and a fractional fair solution. Our final and most important step is to combine these two solutions into an integral and essentially fair solution. It consists of two conceptual substeps: Firstly, we show that it is possible to find a fractional fair assignment to the centers of the integral solution that is sufficiently cheap. Secondly, we round the assignment. This last substep introduces the potential fairness violation of one point per color per cluster. Algorithms 2 and 3 summarize this framework.

Algorithm 2 Black-box assignment with essentially fair solutions.

```
Input: An instance $I$ of a clustering problem, and an integral solution $(\bar{x}, \bar{y})$ to $I$.
Output: An essentially fair solution to $I$ with a cost as specified in Theorem 39.
1: Compute an optimal solution to the LP-relaxation $(x^{LP}, y^{LP})$ of $I$ (see Section 2.2.2).
2: Combine $(\bar{x}, \bar{y})$ and $(x^{LP}, y^{LP})$ using Lemmas 37 and 38 into an essentially fair
   solution to $I$.
```

Algorithm 3 Black-box approximation algorithm for essentially fair clustering.

```
Input: An instance $I$ of a clustering problem.
Output: An essentially fair solution to $I$ with a cost as specified in Theorem 39.
1: Compute an $\alpha$-approximate, integral solution $(\bar{x}, \bar{y})$ to $I$.
2: Use Algorithm 2 with $I$ and $(\bar{x}, \bar{y})$ as input to obtain an essentially fair solution to
   $I$ with a cost as specified in Theorem 39.
```

We show that this approach yields constant-factor approximations with fairness violation for all mentioned clustering objectives. The description will be neutral whenever the objective does not matter. Thus, descriptions like *the LP* mean the appropriate LP for the desired clustering problem. When the problem gets relevant, we will specifically discuss the distinctions.

Step 1: Combining a fair LP solution with an integral unfair solution In the first step, we assume that we are given two solutions. Let (x^{LP}, y^{LP}) be an optimal solution to the LP. This solution has the property that the assignments to all centers are fair, however, the centers may be fractionally open and the points may be fractionally assigned to several centers. Let c^{LP} be the objective value of this solution. For k-center and k-supplier, it is the smallest τ for which the LP is feasible, for the other objectives, it is the value of the LP. We denote the cost of the best *integral* solution to the LP by c^*. We know that $c^{LP} \leq c^*$.

Let $(\bar{x}, \bar{y})$ be any integral solution to the LP that may violate fairness, i.e., inequality (2.14), and let $\bar{c}$ be the objective value of this solution. We think of $(\bar{x}, \bar{y})$ as being a solution of an α-approximation algorithm for the standard (unfair) clustering problem for some constant α. Since the unconstrained version can only have a lower optimum cost, we then have $\bar{c} \leq \alpha \cdot c^*$.

Our goal is now to combine (x^{LP}, y^{LP}) and $(\bar{x}, \bar{y})$ into a third solution, $(\hat{x}, \hat{y})$, such that the cost of $(\hat{x}, \hat{y})$ is bounded by $O(c^{LP} + \bar{c}) \subseteq O(c^*)$. Furthermore, the entries of $\hat{y}$ shall be integral. The entries of $\hat{x}$ may still be fractional after step 1.

Let S be the set of centers that are open in $(\bar{x}, \bar{y})$. For all $j \in P$, we use $\bar{\phi}(j)$ to denote the center in S closest to j, i.e., $\bar{\phi}(j) = \arg\min_{i \in S} d(j, i)$ (ties broken arbitrarily). Notice that the objective value of using S with assignment $\bar{\phi}$ for all points in P is at most $\bar{c}$, since assigning to the closest center is always optimal for the standard clustering problems without the fairness constraint.

Depending on the objective, L is a subset of P or not, i.e., $\bar{\phi}$ is not necessarily defined for all locations in L. We then extend $\bar{\phi}$ in the following way. Let $i \in L \backslash P$ be any center, and let j^* be the closest point to it in P. Then we set $\bar{\phi}(i) := \bar{\phi}(j^*)$, i.e., i is assigned to the center in S which is closest to the point in P which is closest to i. Finally, let $\bar{C}(i) = \bar{\phi}^{-1}(i)$ be the set of all points and centers assigned to i by $\bar{\phi}$.

Lemma 37. *Let (x^{LP}, y^{LP}) and $(\bar{x}, \bar{y})$ be two solutions to the LP, where $(\bar{x}, \bar{y})$ may violate inequality* (2.14)*, but is integral. Then the solution defined by*

$$\begin{aligned}
\hat{y} &:= \bar{y}, \\
\hat{x}_{ij} &:= \sum_{i' \in \bar{C}(i)} x^{LP}_{i'j} \quad \textit{for all } i \in L \textit{ with } \bar{y}_i = 1, j \in P, \\
\hat{x}_{ij} &:= 0 \quad \textit{for all } i \in L \textit{ with } \bar{y}_i = 0, j \in P.
\end{aligned}$$

satisfies inequality (2.14)*, $\hat{y}$ is integral, and the cost $\hat{c}$ of $(\hat{x}, \hat{y})$ is bounded by $c^{LP} + \bar{c}$ for k-center, by $2 \cdot c^{LP} + \bar{c}$ for k-supplier, k-median, and facility location, and by $12 \cdot c^{LP} + 8 \cdot \bar{c}$ for k-means.*

Proof. Recall that for k-center and k-supplier, speaking of the cost of an LP solution is a bit sloppy; we mean that $(\hat{x}, \hat{y})$ is a feasible solution in the LP with threshold $\hat{c}$.

The definition of $(\hat{x}, \hat{y})$ means the following. For every (fractional) assignment from a point j to a center i', we look at the cluster with center $i = \bar{\phi}(i')$ to which i' is assigned to by $\bar{\phi}$. We then transfer this assignment to i. So from the perspective of i, we collect all fractional assignments to centers in $\bar{C}(i)$ and consolidate them at i. Notice that the (fractional) number of points assigned to i after this process may be less than one since $(\bar{x}, \bar{y})$ may include centers that are very close together.

Since $\hat{y}$ is simply $\bar{y}$, it is integral as well and has the same number of centers, thus $\hat{y}$ also satisfies (2.10) if the problem uses it. Next, we observe that $(\hat{x}, \hat{y})$ satisfies fairness, i.e., respects inequality (2.14). This is true because (x^{LP}, y^{LP}) satisfies them, and because we move *all* assignment from a center i' to the same center $\bar{\phi}(i')$. This transferring operation preserves the fairness. Inequality (2.8) is true because we only move assignments to centers that are fully open in $(\bar{x}, \bar{y})$, i.e., the inequality cannot be violated as long as (2.7) is true (which it is for (x^{LP}, y^{LP}) since it is a feasible LP solution). Equality (2.7) is true for $(\hat{x}, \hat{y})$ since all assignment of j is moved to some fully open center. Thus $(\hat{x}, \hat{y})$ is a feasible solution for the LP. It remains to show that $\hat{c}$ is small enough, which depends on the objective.

k-median and k-means. We start by showing this for k-median (where the distances are a metric, i.e., $\beta = 1$ in the β-triangle inequality (1.1)) and k-means (where the distances are a semi-metric with $\beta = 2$). We observe that here, the cost of $(\hat{x}, \hat{y})$ is

$$\hat{c} = \sum_{j \in P} \sum_{i \in L} \hat{x}_{ij} d(i,j) = \sum_{j \in P} \sum_{i \in L} \sum_{i' \in \bar{C}(i)} x^{LP}_{i'j} d(i,j).$$

Now fix $i \in L$, $i' \in \bar{C}(i)$ and $j \in P$ arbitrarily. By the β-relaxed triangle inequality, $d(i,j) \leq \beta \cdot d(i',j) + \beta \cdot d(i',i)$. Furthermore, we know that $i' \in \bar{C}(i)$, i.e., $\bar{\phi}(i') = i$ and $d(i',i) \leq d(i', \bar{\phi}(j))$. We can use this to relate $d(i',i)$ to the cost that j pays in $(\bar{x}, \bar{y})$:

$$d(i',i) \leq d(i', \bar{\phi}(j)) \leq \beta \cdot d(j,i') + \beta \cdot d(j, \bar{\phi}(j)).$$

Adding this up yields

$$\begin{aligned} &\sum_{j \in P} \sum_{i \in L} \sum_{i' \in \bar{C}(i)} x^{LP}_{i'j} d(i,j) \\ \leq &\sum_{j \in P} \sum_{i \in L} \sum_{i' \in \bar{C}(i)} (\beta + \beta^2) x^{LP}_{i'j} d(i',j) + \sum_{j \in P} \sum_{i \in L} \sum_{i' \in \bar{C}(i)} \beta^2 \cdot x^{LP}_{i'j} d(j, \bar{\phi}(j)) \\ = &(\beta + \beta^2) \cdot c^{LP} + \beta^2 \cdot \bar{c}. \end{aligned}$$

For $\beta = 1$ (k-median), this is $2c^{LP} + \bar{c}$, for $\beta = 2$ (k-means), we get $6c^{LP} + 4\bar{c}$.

Facility location. For facility location, we have to include the facility opening costs. We open the facilities that are open in $(\bar{x}, \bar{y})$, which incurs a cost of $\sum_{i \in L} \bar{y}_i f_i$. The distance costs are the same as for k-median, so we get a total cost of

$$\sum_{j \in P} \sum_{i \in L} \sum_{i' \in \bar{C}(i)} 2x^{LP}_{i'j} d(i',j) + \sum_{j \in P} \sum_{i \in L} \sum_{i' \in \bar{C}(i)} x^{LP}_{i'j} d(j, \bar{\phi}(j)) + \sum_{i \in L} \bar{y}_i f_i \leq \ 2c^{LP} + \bar{c}.$$

k-center and k-supplier. For the k-center and k-supplier proof, we again fix $i \in L$, $i' \in \bar{C}(i)$ and $j \in P$ arbitrarily and use that $d(i,j) \leq d(i,i') + d(i',j)$. Now for k-center, we know that $d(i,i') \leq \bar{c}$ since $i' \in \bar{C}(i)$, and we know that $d(i',j) \leq c^{LP}$ for all j where x^{LP}_{ij} is strictly positive. Thus, if $\hat{x}_{ij}$ is strictly positive, then $d(i,j) \leq \bar{c} + c^{LP}$. For k-supplier, we have no guarantee that $d(i,i') \leq \bar{c}$ since i' is not necessarily an input point. Instead, $i' \in \bar{C}(i)$ means that the point j' in P which is closest to i' is assigned to i by $\bar{x}$. Since j' is the closest to i' in P, we have $d(i',j') \leq d(i',j)$. Furthermore, since $j' \in \bar{C}(i)$, $d(i,j') \leq \bar{c}$. Thus, we get for k-supplier that

$$d(i,j) \leq d(i,i') + d(i',j) \leq d(i,j') + d(i',j') + d(i',j) \leq \bar{c} + 2 \cdot c^{LP}.$$

□

Step 2: Rounding the x-variables For rounding the x-variables, we need to distinguish between two cases of objectives. Let $j \in P$ be a point that is fractionally assigned to some centers $L_j \subseteq L$.

First, we have objectives where we can transfer mass from an assignment of j to $i' \in L_j$ to an assignment of j to $i'' \in L_j$ without modifying the objective. We say that such objectives are *reassignable* (in the sense that we can reassign j to centers in L_j without changing the cost). k-center and k-supplier have this property.

Second, we have objectives where the assignment cost is separable, i.e., where the distances influence the cost via a term of the form $\sum_{i \in L, j \in P} c_{ij} \cdot x_{ij}$ for some $c_{ij} \in \mathbb{R}_{\geq 0}$. We call such objectives *separable*. Facility location, k-median and k-means fall into this category.

Lemma 38. *Let (x, y) be an α-approximate fractional solution for a fair clustering problem and the property that all $y_i, i \in L$ are integral. Then we can obtain an essentially fair integral solution (x', y) with at most the same cost as (x, y).*

Proof. We create our rounded α-approximate integral solution (x', y') by min-cost flow computations. We construct a min-cost flow instance which depends on our starting solution (x, y) as well as on the objective of the problem we are studying. Let $S = \{i \in L \mid y_i = 1\}$. We define a min-cost flow instance $(G = (V, A), c, b)$ (also see Figure 2.2) with unit capacities, costs c on the edges and balances b on the nodes. We begin by defining a graph $G^h = (V^h, A^h)$ for every color $h \in Col$ with

$$\begin{aligned} V^h &:= V_S^h \cup V_P^h, \quad V_S^h := \left\{ v_i^h \mid i \in S \right\}, \quad V_P^h := \left\{ v_j^h \mid j \in col_h(P) \right\}, \\ A^h &:= \left\{ (v_j^h, v_i^h) \mid i \in S, j \in col_h(P) : x_{ij} > 0 \right\}, \end{aligned}$$

as well as costs c^h by $c_a^h := c_{ij}$ for $a = (v_j^h, v_i^h) \in A^h, i \in S, j \in col_h(P)$ and balances b^h by $b_v^h := 1$ if $v \in V_P^h$ and $b_v^h := -\lfloor \mathrm{mass}_h(x, i) \rfloor$ if $v = v_i^h \in V_S^h$. We use the graphs G_h to define $G = (V, A)$ by

$$\begin{aligned} V :=& \{t\} \cup V_S \cup \bigcup_{h \in Col} V^h, \quad V_S := \{v_i \mid i \in S\} \\ A :=& \bigcup_{h \in Col} A^h \cup \left\{ (v_i^h, v_i) \mid i \in S, h \in Col : \mathrm{mass}_h(x, i) - \lfloor \mathrm{mass}_h(x, i) \rfloor > 0 \right\} \\ & \cup \{ (v_i, t) \mid i \in S : \mathrm{mass}(x, i) - \lfloor \mathrm{mass}(x, i) \rfloor > 0 \}, \end{aligned}$$

together with costs c of $c_a := c_a^h$ for $a \in A^h$ and 0 otherwise, and balances b of $b_v := b_v^h$ if $v \in V^h$, $b_v := -B_i$ if $v = v_i \in V_S$ and $b_t := -B$ with $B_i = \lfloor \mathrm{mass}(x, i) \rfloor - \sum_{h \in Col} \lfloor \mathrm{mass}_h(x, i) \rfloor$ and $B := |P| - \sum_{i \in S} \lfloor \mathrm{mass}(x, i) \rfloor$.

Separable objectives – k-median and k-means. We make the following observations.

1. B and B_i are integers for all $i \in S$, and so are all capacities, costs and balances. Consequently, there are integral optimal solutions for the min-cost flow instance (G, c, b),
2. (x, y) induces a feasible solution for (G, c, b), by defining a flow x in G as follows:

$$x_a := \begin{cases} x_{ij} & \text{if } a = (v_j^h, v_i^h) \in A^h, j \in P, i \in S, \\ \mathrm{mass}_h(x, i) - \lfloor \mathrm{mass}_h(x, i) \rfloor & \text{if } a = (v_i^h, v_i) \in A, h \in Col, i \in S, \\ \mathrm{mass}(x, i) - \lfloor \mathrm{mass}(x, i) \rfloor & \text{if } a = (v_i, t) \in A, i \in S. \end{cases}$$

Since (x, y) is a fractional solution, x satisfies capacity and non-negativity constraints because $x_{ij} \in [0, 1]$ for all $i \in L, j \in P$ and $\text{mass}_h(x, i) - \lfloor \text{mass}_h(x, i) \rfloor, \text{mass}(x, i) - \lfloor \text{mass}(x, i) \rfloor \in [0, 1]$ for all $i \in S$ and $h \in Col$ as well. We have flow conservation since the fractional solution needs to assign all points, and the flow of the edges (v_i^h, v_i) and (v_i, t) as well as the demand of v_i and t are chosen in such a way that flow conservation is preserved for all the other nodes as well.

3. Integral solutions $\bar{x}$ to the min-cost flow instance (G, c, b) induce an integral solution $(\bar{x}, y)$ to the original clustering problem by setting $\bar{x}_{ij} := \bar{x}_a$ for $a = (v_j^h, v_i^h) \in A^h$. Since the flow x is integral, this gives us an integral assignment of all points to centers in S.

   The assignment is essentially fair, since every $i \in S$ is guaranteed by our balances to have at least $\lfloor \text{mass}_h(x, i) \rfloor$ points of color $h \in Col$ and at least $\lfloor \text{mass}(x, i) \rfloor$ points in total assigned to it. Since there is at most one outgoing edge of unit capacity (v_i^h, v_i) for a node v_i^h, $i \in S$, $h \in Col$ as well as only one outgoing edge (v_i, t) for a node v_i, $i \in S$ if $\text{mass}_h(x, i) - \lfloor \text{mass}_h(x, i) \rfloor > 0$, we have at most $\lceil \text{mass}_h(x, i) \rceil$ points of color h and $\lceil \text{mass}(x, i) \rceil$ total points assigned to i.

Together, this yields that computing an integral min-cost flow $\hat{x}$ for (G, c, b) followed by applying the third observation to $\hat{x}$ yields a solution $(\hat{x}, y)$ to the clustering with an additive fairness violation of at most one.

Since (x, y) was inducing the fractional solution x with $\mathsf{cost}(x) = \mathsf{cost}(x, y)$ to the min-cost flow instances, and $\mathsf{cost}(x) \geq \mathsf{cost}(\hat{x})$ by construction we have $\mathsf{cost}(\hat{x}, y) \leq \mathsf{cost}(x, y)$.

Reassignable objectives – k-center and k-supplier. In the case of reassignable objectives, we do not have to care about costs, as long as the reassignments happen to centers in L_j for all points $j \in P$. We essentially use the same strategy as before, but instead of a min cost flow problem we solve the transshipment problem $(G = (V, A), b)$ with unit capacities on the edges and balances b on the nodes. Notice that the three observations from the previous case apply here as well, and reassignability guarantees that the cost does not increase. □

Lemmas 37 and 38 then lead directly to Theorem 8, formulated in more detail in the following theorem.

Theorem 39. *Algorithm 3 for fair clustering problems returns essentially fair solutions with a cost of $c^{LP} + \bar{c}$ for k-center, $2c^{LP} + \bar{c}$ for k-supplier, k-median and facility location, and $6c^{LP} + 4\bar{c}$ for k-means where c^{LP} is the cost of an optimal solution to the fair LP relaxation and $\bar{c}$ is the cost of the unfair clustering solution.*

We know that for k-center, k-supplier, k-median and facility location c^{LP} is at most as expensive as an optimal solution to the fair clustering problem. For k-means we know that c^{LP} is at most twice as expensive as an optimal solution to the fair clustering problem (Lemma 116). If we use an α-approximation algorithm to obtain the unfair clustering solution, we have that $\bar{c}$ is at most α times the cost of an optimal solution

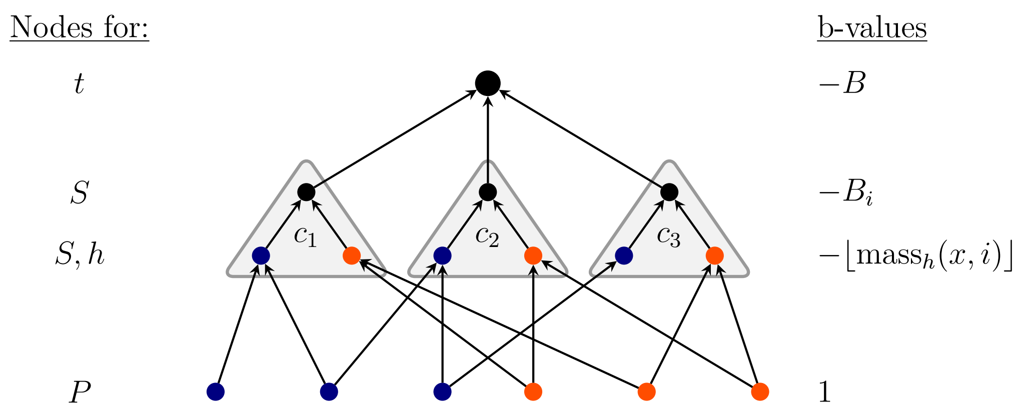

Figure 2.2: Example for the graph G used in the rounding of the x-variables. $B_i = \lfloor \text{mass}(x,i) \rfloor - \sum_{h \in Col} \lfloor \text{mass}_h(x,i) \rfloor$ and $B = |P| - \sum_{i \in S} \lfloor \text{mass}(x,i) \rfloor$.

to the fair clustering problem. Currently, the best known approximation ratios are 2 for k-center [47, 53], 3 for k-supplier [53], 1.488 for facility location [76], 2.675 for k-median [22, 79] and 6.357 for k-means [5], yielding the following theorem.

Theorem 40. *Algorithm 3 for fair clustering problems returns essentially fair solutions with an approximation ratio of* 3 *for k-center,* 5 *for k-supplier,* 4.675 *for k-median,* 3.488 *for facility location, and* 37.428 *for k-means.*

2.2.4 True Approximation Algorithms for Fair k-Center and k-Supplier

We now extend our weakly supervised rounding technique for k-center and k-supplier in the case of the exact fairness model. We replace the black-box algorithm with a specific approximation algorithm, and then achieve true approximation algorithms for the fair clustering problems by informed rounding of the LP solution.

5-approximation algorithm for k-center In this section, we consider the fair k-center problem with exact preservation of ratios and without any additive fairness violation.

We give a 5-approximation algorithm for this variant. The algorithm begins by choosing a set of centers. In contrast to Section 2.2.3 we do not use an arbitrary algorithm for the standard k-center problem but specifically look for nodes in the threshold graph $G_\tau = (P, E_\tau)$ where $E_\tau = \{(i,j) \mid i \neq j \in P, d(i,j) \leq \tau\}$ that form a maximal independent set S in G_τ^2. Here $G_\tau^t = (P, E_\tau^t)$ denotes the graph on P that connects two nodes $j \neq j' \in P$ if there exists a $j - j'$-path of length at most t in G_τ. As we use the following procedure independent for each connected component of G_τ, we will in the description and the following proofs of the procedure assume that G_τ is a connected graph. The procedure uses the approach by Khuller and Sussmann [66] (procedure AssignMonarchs) to find a set $S \subseteq P$ which ensures the following property: There exists a tree T spanning all the nodes in S such that for any two adjacent nodes i, i' in T the shortest $i - i'$-path in G_τ has a length of 3. The procedure begins by choosing an arbitrary node $r \in P$, called *root*, into S and marking every node within distance 2 of r (including itself). Until all the nodes in P are marked, it chooses an unmarked node u that is adjacent to a marked node v and marks all nodes in the distance two neighborhood of u. Observe that u is exactly at distance 3 from a node $u' \in S$ chosen earlier that caused v to get marked. Thus the run of the procedure implicitly defines the tree T over the nodes of S. In case G_τ is not a connected graph this procedure is run on each connected component and the set S has the following property: There exists a forest F such that F reduced to a connected component of G_τ is a tree T spanning all the nodes of S inside of that connected component and two adjacent nodes in T are exactly distance 3 apart in G_τ.

In the next phase we use Observation 34 and the fixed set of centers S to obtain the following adjusted LP for the fractional fair k-center problem.

$$\sum_{i \in S} x_{ij} = 1, \qquad \forall j \in P \tag{2.17}$$

$$\sum_{j \in col_h(P)} x_{ij} = r_h(P) \sum_{j \in P} x_{ij} \qquad \forall i \in S \tag{2.18}$$

$$\sum_{\substack{j \in col_h(P) \\ (i,j) \in E_\tau^2}} x_{ij} \geq b_h \qquad \forall i \in S, \forall h \in Col \tag{2.19}$$

$$x_{ij} = 0 \qquad \forall i \in S, j \in P \text{ with } (i,j) \notin E_\tau^3 \tag{2.20}$$

$$0 \leq x_{ij} \leq 1 \qquad \forall i \in S, j \in P \tag{2.21}$$

Here inequality (2.19) ensures that each cluster contains at least b_h points of color h. Let S_{opt} be the set of centers in an optimal solution and let $\phi_{\mathsf{opt}} : P \to S_{\mathsf{opt}}$ be an optimal fair assignment. For $\tau = \mathsf{opt}$, every center $i \in S$ has a distinct center in S_{opt} which is in the neighborhood of i in G_τ. Therefore, there exists b_h points of each color h within distance two of i. This ensures that inequality (2.19) is satisfiable for $\tau = \mathsf{opt}$. And since, every center in S_{opt} is within distance two of some $i \in S$, there exists a fair assignment of points in P to centers in S within distance three. Thus the above LP is feasible for the right τ.

Now for the final phase, the algorithm rounds a fractional solution for the above assignment LP to an integral solution of cost at most 5τ in a procedure motivated by the LP rounding approach used by Cygan et al. in [30] for the capacitated k-center problem. Let $\beta(i)$ denote the children of node $i \in S$ in the tree T. Define quantities $\Gamma(i)$ and $\delta(i)$, $\forall i \in S$ as follows:

$$\Gamma(i) = \left\lfloor \frac{\sum_{j \in col_{h_1}(P)} x_{ij} + \sum_{i' \in \beta(i)} \delta(i')}{b_{h_1}} \right\rfloor b_{h_1}$$

$$\delta(i) = \sum_{j \in col_{h_1}(P)} x_{ij} + \sum_{i' \in \beta(i)} \delta(i') - \Gamma(i)$$

For a leaf node i in the tree T we have $\beta(i) = \emptyset$, then $\Gamma(i)$ denotes the amount of color h_1 assigned to i rounded down to the nearest multiple of b_{h_1}, while $\delta(i)$ denotes the remaining amount. The idea is to reassign the remainder to the parent of i. Then for a node i' that is not a leaf $\Gamma(i')$ denotes the amount of color h_1 assigned to i' plus the remainder that all children of i' want to reassign to i'. Again this amount is rounded down to the nearest multiple of b_{h_1} and $\delta(i')$ again denotes the remainder. Since by definition of b_{h_1} the total number of points in $col_{h_1}(P)$ must be an integer multiple of b_{h_1}, $\Gamma(r)$ also denotes the amount of color h_1 points assigned to r plus the remainder that all children of r want to reassign to r and $\delta(r) = 0$.

Also note that $\Gamma(i)$ is always a positive integer multiple of b_{h_1} for any i, and $\delta(i)$ is always non-negative and less than b_{h_1}.

One can think of the x_{ij} variables as encoding flow from a node j to a node $i \in S$. We call it a color h flow if j has color h. We will reroute these flows (maintaining the ratio constraints) such that $\forall i \in S$, $j \in col_{h_1}(P)$ x_{ij} is equal to $\Gamma(i)$ which is an integral multiple of b_{h_1}.

Lemma 41. *There exists an integral assignment of all nodes with color h_1 to centers in S in G_τ^5 that assigns $\Gamma(i)$ nodes with color h_1 to each center $i \in S$.*

Proof. Construct the following flow network: Take sets $col_{h_1}(P)$ and S to form a bipartite graph with an edge of capacity one between a node $j \in col_{h_1}(P)$ and a center $i \in S$ if and only if $(i, j) \in E_\tau^5$. Connect a source s with unit capacity edges to all nodes in $col_{h_1}(P)$ and each center $i \in S$ with capacity $\Gamma(i)$ to a sink t. We now show a feasible fractional flow of value $|col_{h_1}(P)|$ in this network. For each leaf node i in T which is not the root, assign $\Gamma(i)$ amount of color h_1 flow from the total incoming color h_1 flow $\sum_{j \in col_{h_1}(P)} x_{ij}$ from nodes that are at most distance three away from i in G_τ and propagate the remaining $\delta(i)$ amount of color h_1 flow, coming from distance two nodes, upwards to be assigned to the parent of node i. This is always possible because by definition $\delta(i) < b_{h_1}$ and inequality (2.19) ensure that every center has at least b_{h_1} amount of color h_1 flow coming from distance two nodes. For every non-leaf node i, assign $\Gamma(i)$ amount of incoming color h_1 flow from distance five nodes (including the color h_1 flows propagated upwards by its children) and propagate $\delta(i)$ amount of color h_1 flow from distance two nodes (possible due to inequality (2.19)). Thus every center has $\Gamma(i)$ amount of color h_1 flow passing through it and it is easy to verify that the value of the total flow in the network is $|col_{h_1}(P)|$. Since the network only has integral capacities, there exists an integral max-flow of value $|col_{h_1}(P)|$. □

Lemma 42. *For any reassignment of a color h_1 flow, there exists a reassignment of color h-flow between the same centers for all $h \in Col \setminus \{h_1\}$, such that the resulting fractional assignment of the nodes satisfies the fairness constraints at each center.*

Proof. Say f_{h_1} amount of color h_1 flow is reassigned from center i_1 to another center i_2. Reassign $f_h = r_h \cdot f_{h_1}/r_{h_1}$ amount of color h flow from i_1 to i_2 for each color $h \in Col \setminus \{h_1\}$. This is possible as inequality (2.18) implies that the amount of color h points assigned to i_1 must be equal to $\frac{r_h}{r_1}$ times the amount of color h_1 points assigned to i_1 and f_{h_1} must be less than the amount of color h_1 points assigned to i_1. It is easy to verify that the ratios at i_1 and i_2 remain unchanged as by construction the ratio of the reassigned flows is equal to the original ratio. □

From Lemmas 41 and 42 we can say that there is a fair fractional assignment within distance 5τ such that all the color h_1 assignments are integral and every center i has $\Gamma(i)$ color h_1 nodes assigned to it. Since this assignment is fair the total incoming color h flow at each center must be $\Gamma(i)\frac{b_h}{b_{h_1}}$ which are integers for every center $i \in S$ and every color $h \in Col$.

Lemma 43. *There exists an integral fair assignment in G_τ^5.*

Proof. Construct a flow network for color h nodes similar to the one in lemma 41: Take sets $col_h(P)$ and S to form a bipartite graph with an edge of capacity one between a node $j \in col_h(P)$ and a center $i \in S$ if and only if $(i, j) \in E_\tau^5$. Connect a source s with unit capacity edges to all nodes in $col_h(P)$ and each center $i \in S$ with capacity $\Gamma(i)\frac{b_h}{b_{h_1}}$ to a sink t. The fractional assignment in G_τ^5 due to Lemma 42 is a feasible fractional flow for this network. Since the network only consists of integral demands and capacities, there exists an integral flow which induces the assignment for the color h nodes. $\square$

Lemmas 41, 42 and 43 imply the following theorem.

Theorem 44. *There exists a 5-approximation algorithm for the fair k-center problem with exact preservation of ratios.*

7-approximation algorithm for k-suppliers We adapt the algorithm for the k-suppliers model to obtain a 7-approximation algorithm. The procedure closely resembles the k-center algorithm: construct a bipartite threshold graph $G_\tau = (P \cup L, E_\tau)$ where $E_\tau = \{(i, j) \mid i \in L, j \in P, d(i, j) \leq \tau\}$. Choose an arbitrary *root* node $r \in P$, $S = \{r\}$ and mark r and all nodes in P that adjacent to r in G_τ^2. Until all nodes in P are marked, we choose an arbitrary unmarked node $u \in P$ that is in G_τ^2 adjacent to a marked node. We add u to S and mark u and all nodes adjacent to u in G_τ^2. Note that, since G_τ is bipartite, no two nodes in P are adjacent. The node u is exactly at distance four from a node $u' \in S$ chosen earlier. This process of selecting nodes in S defines a tree T over S with the property that adjacent nodes in T are exactly at distance four of each other in G_τ. Since we apply the procedure separately for each of the connected components of the threshold graph, we may safely assume that G_τ is connected.

Let us now temporarily open one center at each node in S and make the following observations for the k-suppliers case:

1. Observation 34 still holds.
2. The corresponding LP is the same as the k-center LP, except it has E_τ^4 in place of E_τ^3 in inequality (2.20). This ensures the feasibility of the LP since every location in L is at most distance three away from some node in S. (Note that in case G_τ is not connected, it can happen that some locations in L are not connected to any point and therefore more than distance three away from some node in S, but since they are not connected to any point we can safely ignore them, as they cannot be part of the optimal solution.)

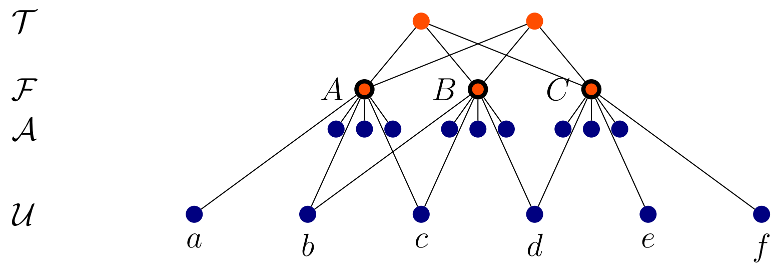

Figure 2.3: Example for the reduction from Exact Cover by 3-sets to the fair assignment problem for k-center, with $\mathcal{U} = \{a,b,c,d,e,f\}$ and $\mathcal{F} = \{A = \{a,b,c\}, B = \{b,c,d\}, C = \{d,e,f\}\}$.

3. Lemma 41 with G^6_τ instead of G^5_τ holds. The extra distance of one is introduced because the distance between a child node and its parent node in T is four instead of three.
4. Lemma 42 holds as it is and Lemma 43 holds when G^5_τ is replaced with G^6_τ.

Thus we have a distance six fair assignment to centers in S. However, this is not a valid solution for k-suppliers as $S \subseteq P$ and we are allowed to open centers only in L. We move each of these temporary centers to the closest location in L and obtain the following theorem.

Theorem 45. *There exists a 7-approximation algorithm for the fair k-supplier problem with exact preservation of ratios.*

Together Theorem 44 and Theorem 45 imply Theorem 7.

2.2.5 NP-Hardness of the Fair Assignment Problem for k-Center

We call the problem of computing a cost-minimal fair assignment of points to given centers the *fair assignment problem*. It exists for every objective listed above. Even for k-center, the fair assignment problem is NP-hard. This can be shown by a reduction from Exact Cover by 3-sets, a variant of the set cover problem. The input is a ground set $\mathcal{U}$ of elements and a family $\mathcal{F}$ of subsets such that each of the subsets in $\mathcal{F}$ has exactly three elements from $\mathcal{U}$. The objective is to find a set cover such that each element is included in exactly one set. For example, let $\mathcal{U} = \{a,b,c,d,e,f\}, \mathcal{F} = \{A = \{a,b,c\}, B = \{b,c,d\}, C = \{d,e,f\}\}$ be an instance. The set $\{A, C\}$ is an exact cover.

For an instance $\mathcal{U}, \mathcal{F}$ of the Exact Cover by 3-sets problem, we construct an unweighted graph, which then translates to an input for the fair assignment problem for k-center by assigning distance 1 to each edge and using the resulting shortest path metric. The nodes consist of $\mathcal{U}$, $\mathcal{F}$ and two sets defined below, $\mathcal{A}$ and $\mathcal{T}$. We start by adding an edge between all $e \in \mathcal{U}$ and any $A \in \mathcal{F}$ if $e \in A$. We assign color red to the nodes from $\mathcal{F}$ and blue to those from $\mathcal{U}$. Then we construct a set $\mathcal{A}$ which contains three auxiliary blue nodes for each node in $\mathcal{F}$. These are exclusively connected to their corresponding node in $\mathcal{F}$. Then we construct a set $\mathcal{T}$ of $|\mathcal{U}|/3$ red nodes.[1] and connect each node in $\mathcal{T}$ to every node in $\mathcal{F}$. Finally, we open a center at each node in $\mathcal{F}$. The construction is shown in Figure 2.3. Observe that the distance between an element $e \in \mathcal{U}$ and an open center at $A \in \mathcal{F}$ in this construction is 1 if $e \in A$, and otherwise, it is 3: If $e \notin A$, then there is no edge between e and A, and since there are no direct connections between the centers, the minimum distance between e and another open center is 3.

Lemma 46. *If there exists an exact cover, there exists a fair assignment of cost* 1 *where the red:blue ratio is 1:3 for each cluster.*

Proof. Assign each red node $A \in \mathcal{F}$ and the three auxiliary blue nodes connected to it to the center at A. If A is in the exact cover, assign the three blue nodes representing its elements and one red node from $\mathcal{T}$ to the center at A. It is straightforward to verify that this assignment is fair and assigns every node to some center to which it is connected via a direct edge. □

Lemma 47. *If there exists a fair assignment where red:blue ratio is* $1:3$ *for each clusters and the maximum radius of every cluster is less than* 3*, then there exists an exact cover.*

Proof. For $A \in \mathcal{F}$, the red node at A and the three auxiliary blue nodes attached to it must be assigned to the center at A as this is the only center within a distance of less than 3. Also, no center can have more than two red nodes assigned to it because there are only six blue nodes in distance less than 3 of any center. Therefore, each red node in $\mathcal{T}$ must be assigned to a distinct center and each such center A will have exactly three blue nodes from $\mathcal{U}$ assigned to it. These three blue nodes from $\mathcal{U}$ correspond to the elements in the set that A represents. Thus, the sets corresponding to the centers that have two red nodes assigned to them form an exact cover for $\mathcal{U}$. □

2.2.6 Integrality Gap of the Canonical Clustering LP

We show that any integral fair solution needs large clusters to implement awkward ratios of the input points. This allows us to derive a non-constant integrality gap for the canonical clustering LP.

Lemma 48. *Let P be a point set with r red and $r-1$ blue points and let $k \geq 1$. If the ratio of red points $r_{red}(C_i)$ is at most $\frac{r-k+1}{2r-2k+1}$ for each cluster C_i, then any fair solution can have at most k clusters.*

Proof. Consider a solution with $k' > k$ clusters. Since we have more red points there must be at least one cluster C_i that contains more red points than blue points. The ratio of red points $r_{red}(C_i)$ of this cluster is minimized if the other $k'-1$ clusters

[1]Note that if $|\mathcal{U}|$ is not a multiple of three, it cannot have an exact cover by 3-sets, so we can assume that $|\mathcal{U}|$ is a multiple of three.

contain only one blue and one red point, and C_i contains with the remaining $r-k'$ blue and $r-k'+1$ red points. However,

$$\frac{r-k'+1}{2r-2k'+1} > \frac{r-k+1}{2r-2k+1}$$

Since the highest ratio of red points in any other solution can only be higher, the claim follows. □

We remark that Lemma 48 is not true for essentially fair solutions.

The canonical fair clustering ILP consists of (2.7)–(2.11) and (2.14). In the k-median/facility location case and in the k-means case, let write opt_F for the optimum value of its LP relaxation and and let us call the value of an optimum integral solution opt_I. We then define the integrality gap of the ILP as $\mathsf{opt}_I \,/\, \mathsf{opt}_F$. In the k-center case, the ILP does not have an objective function, but we can define its integrality gap in the following sense: If τ_I (τ_F) is the smallest τ such that the LP-relaxation has a feasible *integral* (*fractional*) solution, respectively, then we define the integrality gap as τ_I/τ_F.

Lemma 49. *There is a choice of non-trivial fairness intervals such that the integrality gap of the LP-relaxation of the canonical fair clustering ILP is $\Omega(n)$ for the fair k-center/k-supplier/k-median/facility location problem. The integrality gap is $\Omega(n^2)$ for the fair k-means problem.*

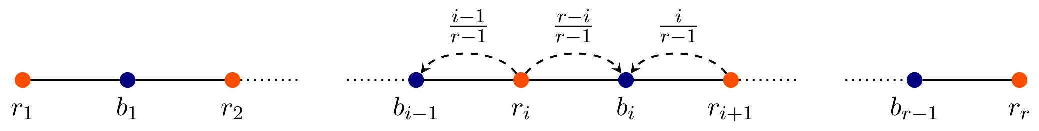

Figure 2.4: Integrality gap example for the fair clustering LP.

Proof. Consider the input points P lying on a line, as shown in Figure 2.4. Specifically, we have r red points $\{r_1, r_2, \ldots, r_r\}$ that alternate with $r-1$ blue points $\{b_1, b_2, \ldots, b_{r-1}\}$. Let the distance function $d : P \to \mathbb{N}$ be defined as follows.

$$\begin{aligned} d(r_i, r_j) &= 2(j-i) \text{ for all } 1 \le i \le j \le r \\ d(b_i, b_j) &= 2(j-i) \text{ for all } 1 \le i \le j \le r-1 \\ d(r_i, b_j) &= 2(j-i)+1 \text{ for all } 1 \le i \le j \le r \\ d(b_i, r_j) &= 2(j-i)-1 \text{ for all } 1 \le i < j \le r \end{aligned}$$

We require that the ratio of the red points of each cluster is between 0 and $(r-1)/(2r-3)$ and set $k = r-1$. The input ratio $r/(2r-1)$ of the red points lies in the interior of this interval as

$$\frac{r}{2r-1} < \frac{r-1}{2r-3} \iff 2r^2 - 3r < 2r^2 - 3r + 1,$$

and thus our input is well-defined and the fairness relaxation is non-trivial. We then ask for a clustering of P with at most k centers that respects the fairness constraints.

Consider the following feasible solution for the LP-relaxation. The solution opens a center at each of the $r-1=k$ blue points and assigns the blue point to itself and the red points on each side in the following way: for each $1 \le i \le r-1$, assign r_i to b_i by a fraction of $\frac{r-i}{r-1}$ and assign r_{i+1} to b_i by a fraction of $\frac{i}{r-1}$. Each red point is fully assigned in this way. We also get that in a cluster around some fixed b_i, the total assignment coming from red points is $\frac{r}{r-1}$ and the assignment coming from blue points is 1; thus, each cluster has a ratio of red points of

$$\frac{\frac{r}{r-1}}{1+\frac{r}{r-1}} = \frac{\frac{r}{r-1}}{\frac{2r-1}{r-1}} = \frac{r}{2r-1}.$$

We therefore respect the balance requirements.

However, as $(r-1)/(2r-3) = (r-k'+1)/(2r-2k'+1)$ for $k'=2$, by Lemma 48 any integral solution satisfying the ratio requirement can at most open two centers.

- In the k-center case, the fractional solution has a radius of 1 and the integral solution has a radius of at least $\lfloor (r-1)/2 \rfloor = \Omega(n)$. The k-center problem is a special case of the k-supplier problem; thus, the integrality gap for the k-supplier problem can only be larger.
- In the k-median case, the fractional solution has a cost of $O(n)$: The blue points incur no cost and each red point r_i contributes $(r-i)/(r-1)\cdot 1+(i-1)/(r-1)\cdot 1 = 1$ to the objective function. Since the optimum integral solution can have at most two centers, it has to contain one cluster spanning at least $\lfloor r/2 \rfloor$ consecutive points. This incurs a cost of at least $2 \cdot \sum_{j=1}^{\lfloor r/4 \rfloor -1} j = \Omega(n^2)$.
- In the facility location case, we observe that we can open at most two facilities in a fair integral solution. Hence, the analysis for the k-median case carries over (even if we set all opening costs to zero).
- In the k-means case, each red point r_i incurs a cost of $(r-i)/(r-1) \cdot 1^2 + (i-1)/(r-1) \cdot 1^2 = 1$ in the fractional solution; the blue points again incur no cost as they are chosen as centers. However, the integral solution now has a cost of at least $2 \cdot \sum_{j=1}^{\lfloor r/4 \rfloor -1} j^2 = \Omega(n^3)$.

□

This integrality gap yields a lower bound on the quality guarantee of any LP-rounding approach for this ILP. Thus, Lemma 35 implies that no fair constant-factor approximation can be achieved by rounding the canonical fair clustering ILP. The counterexample from breaks down in the essential fairness model.

2.3 Privacy Preserving Clustering

In this section we mostly discuss constrained k-center/k-supplier problems. In our method we use an approximation algorithm for several constrained k-center/k-supplier problems as a black box algorithm and create an approximation algorithm for the k-center/k-supplier problems which, in addition to the previous constraints, also satisfies the privacy constraint. We start by showing how to add privacy to the k-center problem with outliers. Then we show how to adjust the method to other constraints, which include capacities and fairness, as well as the combination of capacities and fairness. After that, we introduce the notion of strong privacy and show that our method can easily be adjusted to that case as well. We conclude the section with a discussion of the private facility location problem.

2.3.1 The Private k-Center Problem with Outliers

Theorem 50. *Assume that there exists an α-approximation algorithm A for the k-center problem with outliers. Then for instances P, L, k, ℓ, o of the private k-center problem with outliers, we can compute an $(\alpha + 2)$-approximation in polynomial time.*

Proof. Below, we describe an algorithm that uses a threshold graph with threshold τ. We show that for any given $\tau \in \mathbb{R}$, the algorithm has polynomial run time and, if τ is equal to opt, the value of the optimal solution, computes an $(\alpha + 2)$-approximation. Since we know that the value of every solution is equal to the distance between a point and a location, we can test all $O(|P||L|)$ possible distances as τ and return the best feasible clustering returned by any of them. The main proof is the proof of Lemma 51 below, which concludes this proof. □

Let us start by explaining the general idea on how the algorithm works for a fixed threshold τ. If $\tau < \mathsf{opt}$, then the algorithm could fail, in which case it returns $\tau < \mathsf{opt}$, so assume that $\tau \geq \mathsf{opt}$ for now. We first use the approximation algorithm to obtain a clustering which does not necessarily satisfy the privacy constraint. Then we try to reassign points to establish the lower bounds. A point j may be reassigned to any cluster which contains at least one point which is within distance 2τ of j. (Note that any point in j's optimum cluster is within distance 2τ of j). If it is possible to reassign points like that and obtain a feasible clustering, we can compute such a reassignment with a maximum flow algorithm. Otherwise we find a set of points P' for which we can show that any clustering with radius at most τ must use fewer clusters to cover all of them. We then use the approximation algorithm on P' to compute a new clustering with outliers. The new clustering will contain fewer clusters or the same number of clusters and fewer relevant outliers. We repeat the process until we find a feasible solution.

Lemma 51. *Assume that there exists an α-approximation algorithm A for the k-center problem with outliers. Let P, L, k, ℓ, o be an instance of the private k-center problem with outliers, let $\tau > 0$ and let opt denote the maximum radius in the optimal feasible clustering for P, L, k, ℓ, o. We can in polynomial time compute a feasible clustering with a maximum radius of at most $(\alpha + 2)\tau$ or determine $\tau < \mathsf{opt}$.*

Proof. We first use A to compute a solution without the lower bound. Let $\mathcal{C} = (S, \phi)$ be an α-approximation for the k-center problem with outliers on P, L, k, o. Note that $\mathcal{C}$ may contain clusters with less than ℓ points. Let $k' = |S|$ (note that $k' < k$ is possible). Finally, let $r = \max_{x \in P} d(x, \phi(x))$ be the largest distance of any point to its center. Observe that an optimal solution to the k-center problem with outliers can only have a lower objective value than the optimal solution to our problem because we only dropped a condition. Therefore, $\tau \geq \mathsf{opt}$ implies that $r \leq \alpha \cdot \mathsf{opt} \leq \alpha \cdot \tau$. If we have $r > \alpha \cdot \tau$, we return $\tau < \mathsf{opt}$.

We use $\mathcal{C}$ and τ to create a threshold graph which we use to either reassign points between the clusters to obtain a feasible solution or to find a set of points P' for which we can show that every feasible clustering with maximum radius τ uses less clusters than our current solution to cover it. In the latter case we compute another α-approximation which uses fewer clusters to cover P' and repeat the process. Note that for $\tau < \mathsf{opt}$ such a clustering does not necessarily exist, but for $\tau \geq \mathsf{opt}$ the optimal clustering provides a solution covering P' with fewer clusters. If we do not find such a clustering with maximum radius at most $\alpha \cdot \tau$, we return $\tau < \mathsf{opt}$.

We show that every iteration of the process reduces the number of clusters or the number of relevant outliers, therefore the process stops after at most $k \cdot o$ iterations. It may happen that our final solution contains much less clusters than the optimal solution (but it will be an approximation for the optimal solution with up to k centers).

We will use a network flow computation to move points from clusters with more than ℓ points to clusters with less than ℓ points. Moving a point to another cluster can increase the radius of the cluster. We only want to move points between clusters such that the radius does not increase by too much. More precisely, we only allow a point j to be reassigned to another center $i \in S$ if the distance $d(j, P(i))$ between j and the cluster assigned to i is at most 2τ. This is ensured by the structure of the network described in the next paragraph. Unless stated otherwise, when we refer to distances between a point j and a cluster $P(i)$ in the following, we mean the distance between j and $P(i)$ in its original state before any points have been reassigned.

Given $\mathcal{C}$ and τ, we create the threshold graph G_τ as the directed flow network (V_τ, E_τ) as follows. V_τ consists of a source s, a sink t, a node v_i for each $i \in S$, a node v_{out} for the set of outliers and a node w_j for each point $j \in P$. For all $i \in S$, we connect s to v_i if $P(i)$ contains more than ℓ points and set the capacity of (s, v_i) to $|P(i)| - \ell$. If $P(i)$ contains fewer than ℓ points, we connect v_i with t and set the capacity of (v_i, t) to $\ell - |P(i)|$. Furthermore, we connect v_i with w_j for all $j \in P(i)$ and set the capacity of (v_i, w_j) to 1. We also connect s to v_{out} with capacity o and v_{out} with w_j for all $j \in P(out)$ with capacity 1. Whenever a point j and a cluster $P(i)$ with $j \notin P(i)$ satisfy $d(j, P(i)) \leq 2\tau$ (i.e., there is a point $j' \in P(i)$ that satisfies $d(j, j') \leq 2\tau$), we connect w_j with v_i with capacity 1.

Formally the graph $G_\tau = (V_\tau, E_\tau)$ is defined by

$$V_\tau = \{v_{out}\} \cup \{v_i \mid i \in S\} \cup \{w_j \mid j \in P\} \cup \{s, t\} \text{ and} \tag{2.22}$$

$$E_\tau = \{(v_i, w_j) \mid j \in P(i)\} \cup \{(w_j, v_i) \mid j \notin P(i) \wedge d(j, P(i)) \leq 2\tau\} \tag{2.23}$$

$$\cup \{(v_{out}, w_j) \mid j \in P(out)\} \tag{2.24}$$

$$\cup \{(s, v_{out})\} \cup \{(s, v_i) \mid |P(i)| - \ell > 0\} \cup \{(v_i, t) \mid |P(i)| - \ell < 0\}. \tag{2.25}$$

We define the capacity function $cap : E_\tau \to \mathbb{R}$ by

$$cap(e) = \begin{cases} \ell - |P(i)|, & \text{if } e = (v_i, t) \\ |P(i)| - \ell, & \text{if } e = (s, v_i) \\ o, & \text{if } e = (s, v_{out}) \\ 1 & \text{otherwise.} \end{cases} \tag{2.26}$$

We use $G = (V, E)$ to refer to G_τ as τ is clear from context. We now compute an integral maximum s-t-flow f on G. According to f we can reassign points to different clusters.

Lemma 52. *Let f be an integral maximal s-t-flow on G. It is possible to reassign j to i for all edges (w_j, v_i) with $f((w_j, v_i)) = 1$.*

The resulting solution has a maximum radius of at most $r + 2\tau$. If f saturates all edges of the form (v_i, t), then the solution is feasible.

Proof. Let $j \in P(i)$. The choice of capacity 1 on (v_i, w_j) and flow conservation ensure $\sum_{(w_j,v)\in E} f((w_j, v)) \leq 1$. Therefore no point would have to be reassigned to multiple new clusters. Note that for every point $j \in P(i)$ that would be reassigned we must have $f((v_i, w_j)) = 1$ and for every edge (v_i, w_j) with $f((v_i, w_j)) = 1$ the point j would be reassigned.

For any $i' \in S$, let $j \in P(i)$ be any point which we want to reassign to i'. Then we must have $(w_j, v_{i'}) \in E$ and therefore there must be a point $j' \in P(i')$ with $d(j, j') \leq 2\tau$. Thus we have

$$d(j, i') \leq d(j, j') + d(j', i') \leq 2\tau + r = r + 2\tau.$$

Now assume that f saturates all edges of the form (v_i, t) for $i \in S$. If E contains the edge (v_i, t), then it cannot contain the edge (s, v_i) and therefore all incoming edges of v_i are of the form (w_j, v_i). Flow conservation then implies that the number of points reassigned to i minus the points from $P(i)$ reassigned away is equal to $f((v_i, t))$, which increases the number of points in $P(i)$ to ℓ.

If E contains the edge (s, v_i), then it cannot contain the edge (v_i, t) and therefore all outgoing edges of v_i are of the form (v_i, w_j). Flow conservation then implies that the number of points from $P(i)$ reassigned away minus the points reassigned to i is equal to $f((s, v_i))$, which cannot reduce the number of points assigned to i to fewer than ℓ points.

If E contains neither (s, v_i) nor (v_i, t), then the number of points in $P(i)$ is equal to ℓ and does not change (Which points are assigned to i might change, but the number of points assigned to i does not).

In all three cases $P(i)$ contains at least ℓ points after the reassignment. □

If f saturates all edges of the form (v_i, t) in G, then we reassign points according to Lemma 52 and return the new clustering.

Otherwise we look at the residual network G_f of f on G. Let V' be the set of nodes in G_f which cannot be reached from s. V' contains, among possibly others, all nodes representing clusters which would not satisfy the privacy constraint after the reassignment. We say cluster $P(i)$ *belongs* to V' if $v_i \in V'$, otherwise we say a cluster is uncritical. We say a point $j \in P(i)$ is *adjacent* to V' if $w_j \in V'$ and $v_i \notin V'$. Let $S(V')$ denote the set containing the centers of the clusters belonging to V'. Let $k'' = |S(V')|$. We say a point j belongs to V' if the cluster $P(i)$ with $j \in P(i)$ belongs to V'. Let $P(V')$ and $P_A(V')$ denote the set of points that belong to V' and the set of points adjacent to V'.

Lemma 53. *Any clustering on P with maximum radius at most τ that contains at least ℓ points in every cluster uses fewer than k'' clusters to cover all points in $P(V')$.*

Proof. We first observe that V' must have the following properties:

- $v_i \in V'$ and $(w_j, v_i) \in E$ implies $w_j \in V'$.
- $w_j \in V'$, $(w_j, v_i) \in E$ and $f((w_j, v_i)) > 0$ implies $v_i \in V'$.
- $w_j \in V'$ for some $j \in P(i)$ and $v_i \notin V'$ implies $f((v_i, w_j)) = 1$.

The first property follows from the fact that f can only saturate (w_j, v_i) if f also saturates $(v_{i'}, w_j)$ for $j \in P(i')$ or (v_{out}, w_j) for $j \in P(out)$. So, either (w_j, v_i) is not saturated, which means that v_i can be reached from any node that reaches w_j, or (w_j, v_i) *is* saturated, which means that the only incoming edge of w_j in G_f is (v_i, w_j). In both cases, if $v_i \in V'$, then $w_j \in V'$. The second property follows since $f((w_j, v_i)) > 0$ implies $(v_i, w_j) \in E(G_f)$. The third property is true since we defined $cap((v_i, w_j)) = 1$.

This implies that a reassignment due to Lemma 52 would reassign all points adjacent to V' to centers in $S(V')$ and moreover all reassignments from points in $P(V') \cup P_A(V')$ would be to centers in $S(V')$. Let n_i denote the number of points that would be assigned to i after the reassignment. Then $|P(V')| + |P_A(V')| = \sum_{i \in S(V')} n_i$.

Now we argue that this sum is smaller than $k'' \cdot \ell$ by observing that each $n_i \leq \ell$ and at least one n_i is strictly smaller than ℓ.

Let $P(i)$ be a cluster with more than ℓ points after the reassignment. Then (s, v_i) is not saturated by f and v_i can be reached from s in G_f. Therefore after the reassignment no cluster $P(i)$, $i \in S(V')$ contains more than ℓ points; in other words, $n_i > \ell$ implies $i \notin S(V')$.

Let $P(i)$ be a cluster which would still contain fewer than ℓ points after the reassignment. This implies that f does not saturate the edge (v_i, t). Therefore t can be reached from v_i and since f is a maximum s-t-flow, v_i cannot be reached from s. We must have $v_i \in V'$.

Because we assumed that the reassignment does not satisfy all lower bounds, at least one such cluster has to exist. This implies

$$|P(V')| + |P_A(V')| = \sum_{P(i) \in S(V')} n_i < k'' \cdot \ell.$$

This means that the clusters assigned to the centers ins $S(V')$ and the points in $P_A(V')$ do not contain enough points to satisfy the lower bound in k'' clusters.

By definition of G and V', for two points j_1, j_2 with $j_1 \in P(V')$ and $d(j_1, j_2) \leq 2\tau$ we must have $j_2 \in P(V') \cup P_A(V')$. Let $\mathcal{C}'$ be a clustering that abides the lower bounds and has a maximal radius of at most τ. Then every cluster C' in $\mathcal{C}'$ that contains at least one point from $P(V')$ can only contain points from $P(V') \cup P_A(V')$. Therefore $\mathcal{C}'$ must contain fewer than k'' clusters which contain at least one point from $P(V')$. □

If we have $\tau \geq \mathsf{opt}$, then Lemma 53 implies that the optimal solution covers all points in $P(V')$ with fewer than k'' clusters. An α-approximation on the point set $P(V')$ with at most $k'' - 1$ clusters, which contains at most o outliers, is then α-approximation for $P(V')$.

Unfortunately, we do not know how many outliers an optimal clustering has in $P(V')$. We therefore involve the outliers $P(out)$ in our new computation as well. Let $o' = |P(out)|$ denote the current number of outliers. We obtain the following Lemma through a counting argument.

Lemma 54. *We call a cluster special if it contains at least one point from $P(V')$ or only contains points from $P(out)$. Let $\mathcal{C}'$ be a clustering on P with a maximum radius of at most τ on all special clusters that respects the lower bounds on all special clusters, has at most o outliers and consists of at most k clusters out of which at most k'' are special. If $\mathcal{C}'$ has exactly k'' special clusters, then $\mathcal{C}'$ has at most $o' - 1$ outliers in $P(V') \cup P(out)$.*

Proof. Assume the clustering contains exactly k'' special clusters. Each of these clusters has to contain at least ℓ points from $P(V') \cup P_A(V') \cup P(out)$. We know

$$|P(V') \cup P_A(V') \cup P(out)| \leq |P(V') \cup P_A(V')| + o' < k''\ell + o'.$$

So there remain at most $o' - 1$ unclustered points in $P(V') \cup P_A(V') \cup P(out)$. □

Now we need to show that such a clustering exists if $\tau \geq \mathsf{opt}$ is the case.

Lemma 55. *If $\tau \geq \mathsf{opt}$, then there exists a clustering $\mathcal{C}'$ on P with a maximum radius at most τ on all special clusters that respects the lower bounds on all special clusters, has at most o outliers and consists of at most k clusters out of which at most k'' are special.*

Proof. We look at an optimal clustering $\mathcal{C}_{\mathsf{opt}}$. The optimal clustering $\mathcal{C}_{\mathsf{opt}}$ has to have at most o outlier, at most k clusters that all respect the lower bound, each with a radius of at most τ. So the only way $\mathcal{C}_{\mathsf{opt}}$ can violate a condition is if it contains $k''' > k''$ special clusters. Lemma 53 implies that in this case $\mathcal{C}_{\mathsf{opt}}$ must contain at least $k''' - k''$ clusters that contain only points in $P(out)$.

If all clusters in $\mathcal{C}_{\mathsf{opt}}$ are special this implies that all points in $P \backslash (P_A(V') \cup P(V') \cup P(out))$ must be outlier in $\mathcal{C}_{\mathsf{opt}}$. We will add a new cluster that contains all these points with an arbitrary center and arbitrarily select $k''' - k'' \geq 1$ clusters from $\mathcal{C}_{\mathsf{opt}}$ that contain only points in $P(out)$. We declare all points in these clusters as outliers and close the corresponding centers. This leaves us with $k'' + 1 \le k''' \le k$ clusters and k'' special clusters which contain at least $k'' \cdot \ell$ points. This leaves at most $o' - 1$ outliers. Otherwise, if $\mathcal{C}_{\mathsf{opt}}$ contains at least one cluster C which is not special, we add all outlier from $P \setminus (P_A(V') \cup P(V') \cup P(out))$ to C. Again we arbitrarily select $k''' - k''$ clusters from $\mathcal{C}_{\mathsf{opt}}$ that contain only points in $P(out)$. We declare all points in them as outliers and close the corresponding centers. By creation there are no unclustered points in $P \setminus (P_A(V') \cup P(V') \cup P(out))$ and exactly k'' special clusters with radius at most τ. Therefore this clustering contains at most $o' - 1$ outliers and has at most k clusters. □

Note that in the clustering, obtained as explained above, clusters which are not special only contain points in $P \setminus (P(V') \cup P(out))$ and points in $P(out)$ with a distance of at most 2τ to at least one point in $P \setminus (P(V') \cup P(out))$. Let out' denote the set of points in $P(out)$ with a distance higher than 2τ to every point in $P \setminus (P(V') \cup P(out))$. Then all points in $P(V') \cup out'$ must be part of a special cluster or be an outlier.

We now want to use A again to compute a new solution without the lower bound, which uses fewer clusters to cover $P(V')$. With outliers this creates the problem that the optimal solution could declare some points in $P(V')$ as outliers. As the number of outliers is restricted, the optimal solution must compensate for these outliers by assigning some points in $P(out)$ to a cluster. There are two cases how these clusters can look like.

- Case 1: The cluster contains only points in $P(out) \cup P(V')$.
- Case 2: The cluster contains at least one point in $P \setminus (P(V') \cup P(out))$.

As Case 1 clusters are special, Lemmas 54 and 55 show that we can deal with them by including them in the computation of a new clustering on $P(V') \cup P(out)$ (or on $P(V') \cup out'$) with at most k'' clusters and at most $o' - 1$ outliers or less than k'' clusters and at most o outliers. For any point $j \in P(out)$ that is part of a Case 2 cluster in the optimal solution, we know that there must exist an uncritical cluster $P(i) \in \mathcal{C}$ with $d(j, P(i)) \le 2\,\mathsf{opt}$. Therefore all such points could be assigned to uncritical clusters without increasing the radius of the uncritical clusters by too much. We can get rid of all these points by removing all points in $P(out)$ with distance at most 2τ to the next uncritical cluster. We define the set $NewP := P(V') \cup out'$.

We now use A again to compute new solutions without the lower bound: Let $\mathcal{C}'_1 = (S'_1, \phi'_1)$ be an α-approximation for the k-center problem with outliers on $NewP$, L, $k'' - 1$, o and let $\mathcal{C}'_2 = (S'_2, \phi'_2)$ be an α-approximation for the k-center problem with outliers on $NewP$, L, k'', $o' - 1$. Let $r'_i = \max_{j \in NewP} d(j, \phi'_i(j))$ for $i \in \{1, 2\}$.

Given that A is an α-approximation algorithm for the k-supplier problem with outliers or works on instances with $P \subseteq L$ we have no problem and can be sure that A actually computes α-approximations.

The problem with approximation algorithms for the k-center problem with outliers, that require $P = L$ to ensure their approximation guarantee, is that in the optimal solution on P some of the points in $NewP$ might be part of a cluster, whose center lies in $P \setminus NewP$.

In general we can just use a different approximation algorithm, which works for the k-supplier version. Unfortunately in several cases the best known approximation algorithms for k-supplier variants have worse approximation guarantees than the best known counterparts for the k-center variant. We will now explain how we can get around this problem by slightly altering the instance in case A requires $P = L$.

Adjustment for the case that A requires $P = L$ Note that we assume that in the optimal solution the distance between any point and its cluster center is at most τ. Therefore we know that all points in $NewP$ must have a center $P(V') \cup P_A(V') \cup P(out)$. The idea is to alter the set $NewP$ such that points in $P(out) \setminus out'$ as well as points in $P_A(V')$ can be used as a center for a special clusters but do not need to be part of a special cluster. We achieve this with a new structure, that allows us to declare these points as outliers while they essentially will not be counted towards the total number of outliers and can in that case be viewed as if they were assigned to an uncritical cluster. We alter $P_A(V') \cup P(V') \cup P(out)$ as follows. Let m denote the number of points in $P_A(V') \cup P(out) \setminus out'$. We then add m copies of every point in $P(V') \cup out'$ to $P_A(V') \cup P(V') \cup P(out)$. We call this new set $NewV$. It is easy to see that every clustering on $NewP$ with centers in $P_A(V') \cup P(V') \cup P(out)$, $\bar{k}$ clusters and $\bar{o}$ outliers can be translated into a clustering with the same maximal radius on $NewV$ with $\bar{k}$ clusters and at most $(m+1)\bar{o} + m$ outliers by assigning all copies of points in $NewP$ analogous to the original point and declaring all remaining points as outliers. We can also translate any clustering on $NewV$ with $\bar{k}$ clusters and $\bar{o}$ outliers into a clustering on $NewP$ with at most the same maximal radius, centers in $P_A(V') \cup P(V') \cup P(out)$, at most $\bar{k}$ clusters and at most $\left\lfloor \frac{\bar{o}}{m+1} \right\rfloor$ outliers. We do so as follows. First we merge clusters whose centers are copies of the same original point. Every point, for which at least one of its copies is assigned to a cluster will arbitrarily be assigned to a clusters which contains at least one of its copies. Only points for which all their copies are outliers in the clustering on $NewV$ will be declared as outliers in the clustering on $NewP$ as well.

This implies that instead of looking for a clustering with $\bar{k}$ clusters and at most $\bar{o}$ outliers in $NewP$ with centers in $P_A(V') \cup P(V') \cup P(out)$, we can look for a clustering on $NewV$ with $\bar{k}$ clusters and at most $(m+1)\bar{o} + m$ outliers.

We now use A again to compute new solutions without the lower bound: Let $\mathcal{C}_1' = (S_1', \phi_1')$ be the translation to $NewP$ of an α-approximation for the k-center problem with outliers on $NewV$, L, $k'' - 1$, $(m+1)o + m$ and let $\mathcal{C}_2' = (S_2', \phi_2')$ be the translation to $NewP$ of an α-approximation for the k-center problem with outliers on $NewV$, L, k'', $(m+1)(o'-1) + m$. Let $r_i' = \max_{j \in NewP} d(j, \phi_i'(j))$.

Note that in case $\tau < \mathsf{opt}$, it can happen that no such clustering exists or that we obtain $r_i' > \alpha \cdot \tau$ for both $i = 1$ and $i = 2$. We then return $\tau < \mathsf{opt}$. Otherwise $\mathcal{C}_i'$ must exist together with $r_i' \leq \alpha \cdot \tau$ for at least one $i \in \{1, 2\}$.

If $\mathcal{C}_2'$ exists and we have $r_2' \leq \alpha \cdot \tau$ we replace $S(V')$ by S_2' in $\mathcal{C}$ and adjust ϕ accordingly to obtain $\mathcal{C}_1 = (S_1, \phi_1)$ with $S_1 = (S \setminus S(V')) \cup S_2'$ and

$$\phi_1(p) = \begin{cases} \phi_2'(p) & \text{if } p \in NewP \\ \phi(p) & \text{otherwise.} \end{cases} \tag{2.27}$$

Otherwise, if $\mathcal{C}'_1$ exists, we have $r'_1 \leq \alpha{\cdot}\tau$ and either $\mathcal{C}'_2$ does not exist or we have $r'_2 > \alpha{\cdot}\tau$, we analogous replace $S(V')$ by S'_1 to obtain $\mathcal{C}_1 = (S_1, \phi_1)$ with $S_1 = (S \setminus S(V')) \cup S'_1$ and

$$\phi_1(p) = \begin{cases} \phi'_1(p) & \text{if } p \in NewP \\ \phi(p) & \text{otherwise.} \end{cases} \tag{2.28}$$

Notice that on points in $P_A(V')$ we use ϕ and not ϕ'_i and that it could happen that only points in $P_A(V')$ are assigned to some center in S'_1 or S'_2 . All such centers can be ignored and we will delete them from $\mathcal{C}_1$.

Notice that in case we replaced $S(V')$ by S'_2, $\mathcal{C}_1$ may contain more than $o' - 1$ outlier and in case we replaced $S(V')$ by S'_1, $\mathcal{C}_1$ may contain more than o outlier. In both cases, we can reduce the number of outliers below the required amount by assigning all points in $\phi_1^{-1}(out)$ with a distance of at most 2τ to an uncritical cluster in $\mathcal{C}$ to the closest uncritical cluster.

Instead of actually assigning these points we declare them as semi-outliers, which means that we view them as outliers, when it comes to the creation of the flow network and view them as assigned to their nearest cluster, when it comes to counting the number of outliers.

We denote the version of $\mathcal{C}_1$, where all semi-outliers are assigned to their closest cluster as $\bar{\mathcal{C}}_1$. We then obtain the following lemma.

Lemma 56. *In case we did not return $\tau < \mathsf{opt}$, then $\bar{\mathcal{C}}_1$ is a solution for the k-center problem with outliers on P, L, k, o and we have $r_1 = \max_{x \in P} d(x, \phi_1(x)) \leq \alpha \cdot \tau$. (Notice that $\phi_1(x)$ is the assignment in $\mathcal{C}_1$ and not the assignment in $\bar{\mathcal{C}}_1$.)*

We iterate the previous process with the new clustering $\mathcal{C}_1$ until we either determine $\tau < \mathsf{opt}$ or the reassignment of points according to Lemma 52 yields a feasible solution. In case we find a feasible solution, we will as assign all semi-outliers, which are not already assigned in this solution to the closest center. By definition of a semi-outlier this cannot increase the maximal radius to more then $(\alpha + 2)\tau$. We will show that each iteration reduces the number of clusters and has at most o outliers which are not semi-outliers or keeps the same number of clusters and reduces the number of outliers which are not semi-outliers. The process therefore terminates after at most $k \cdot o$ iterations.

After each computation of a network flow there are two cases which can apply to a semi-outlier j. In the first case there is at least one uncritical cluster with a distance at most 2τ to j. In that case, when j stays an outlier it will again be declared as a semi-outlier. In the second case all uncritical clusters have a distance more than 2τ to j. This can only happen if every uncritical cluster from the previous iteration, that was within distance at most 2τ of j became critical after the flow computation. In that case the computed flow must assign j to a center in $S(V')$. Since j has a distance of more than 2τ to the next uncritical cluster and the computed flow assigns it to a cluster in $S(V')$ it analogous to before follows that the optimal solution has to use less clusters to cover $P(V') \cup \{j\}$. In that case we assume that j is actually assigned as the flow suggests and view it as one of the points in $P(V')$. Since this case implies that we use A to compute a new clustering this does not affect the maximum radius.

As all semi-outlier either remain semi-outlier or enter the recomputation phase as a point in $P(V')$ we obtain the following lemma, which concludes the proof.

Lemma 57. *Each iteration reduces the number of clusters and has at most o outlier which are not semi-outlier or keeps the same number of clusters and reduces the number of outliers which are not semi-outlier.*

□

Corollary 58. *We can compute a* 4*-approximation for instances of the private k-center problem with outliers and a* 5*-approximation for instances of the private k-supplier problem in polynomial time.*

Proof. Follows from Theorem 50 together with the 2-approximation algorithm for k-center with outliers in [24] and the 3-approximation algorithm for k-supplier with outliers in [26]. □

2.3.2 Combining Privacy with other Constraints

We want to take the general idea from Section 2.3.1 and instead of outliers we want to combine privacy with other restrictions on the clusters. Given a specific restriction $\mathcal{R}$ and an α-approximation algorithm A for the k-center problem with restriction $\mathcal{R}$ we ask: Can we similar to Section 2.3.1 combine A with the use of a threshold graph to compute an $O(\alpha)$-approximation for the private k-center problem with restriction $\mathcal{R}$?

In Section 2.3.1 we made use of two properties of a clustering with outliers. In Lemma 52 we used that reassigning points to another cluster never increases the number of outliers and in Lemma 53 we used that outliers have the somewhat local property that computing a new clustering on the points V' from a subset of the clusters together with the set of outliers does not create more outliers on the remaining points.

In this section we now take a look at restriction properties which are similarly local, and show how to combine them with privacy.

As we have seen in Section 2.3.1, when we use an approximation algorithm A to recompute new clusters on a subset of the points, the optimal solution might cluster these points to any center in the original set L of locations. In case A can only deal with instances in which we have $P = L$, this might become a problem. We will therefore deal with the general k-supplier versions and only state separate results for the k-center versions, when the underlying approximation algorithms allow use to improve on the supplier result.

The private and capacitated k-center problem

Theorem 59. *Assume that there exists an α-approximation algorithm A for the soft capacitated k-supplier problem. Then we can compute an $\alpha + 2$-approximation for the soft private capacitated k-supplier problem in polynomial time.*

Assume that there exists an α-approximation algorithm A for the soft uniform capacitated k-supplier problem. Then we can compute a $2\alpha + 2$-approximation for the private uniform capacitated k-center problem and a $2\alpha+3$-approximation for the private uniform capacitated k-supplier problem in polynomial time.

Proof. Let P, L, k, u, ℓ be an instance of the private capacitated k-supplier problem.

Analogous to Section 2.3.1 we use a threshold graph with threshold τ and show that for any given $\tau \in \mathbb{R}$ the algorithm has polynomial run time and, if τ is equal to opt, the value of the optimal solution, computes a soft $(\alpha + 2)$-approximation. Since we know that the value of the optimal solution is equal to the distance between a point and a location, we test all $O(|P||L|)$ possible distances for τ and return the best feasible clustering returned by any of them.

The main proof is the proof of Lemma 60 below. The lemma then concludes the proof for the private soft capacitated k-supplier problem. For the private uniform capacitated k-center/k-supplier problem we might need to reassign some of the centers. Lemma 64 then concludes the proof. □

We now describe the procedure for a fixed value of $\tau > 0$.

Lemma 60. *Assume that there exists an α-approximation algorithm A for the soft capacitated k-supplier problem.*

Let P, L, k, u, ℓ be an instance of the private soft capacitated k-supplier problem, let $\tau > 0$ and let opt denote the maximum radius in the optimal feasible clustering for P, L, k, u, ℓ. We can in polynomial time compute a feasible clustering with a maximum radius of at most $(\alpha + 2)\tau$ or determine $\tau < \mathsf{opt}$.

Proof. The algorithm first uses A to compute a solution without the lower bound: Let $\mathcal{C} = (S, \phi)$ be an α-approximation for the capacitated k-supplier problem on P, L, k, u.

Again let $k' = |S|$ and let $r = \max_{j \in P} d(j, \phi(j))$ be the largest distance of any point to its assigned center. If we have $r > \alpha \cdot \tau$, we return $\tau < \mathsf{opt}$.

Given $\mathcal{C}$ and τ, we create, similar to Section 2.3.1, the threshold graph G_τ as the directed flow network (V_τ, E_τ) by

$$\begin{aligned} V_\tau &= \{v_i \mid 1 \le i \le k'\} \cup \{w_j \mid j \in P\} \cup \{s, t\} \text{ and} && (2.29)\\ E_\tau &= \{(v_i, w_j) \mid j \in P(i)\} \cup \{(w_j, v_i) \mid j \notin P(i) \wedge d(j, P(i)) \le 2\tau\} && (2.30)\\ &\cup \{(s, v_i) \mid |P(i)| - \ell > 0\} \cup \{(v_i, t) \mid |P(i)| - \ell < 0\}. && (2.31) \end{aligned}$$

We define the capacity function $cap : E_\tau \to \mathbb{R}$ by

$$cap(e) = \begin{cases} \ell - |P(i)|, & \text{if } e = (v_i, t) \\ |P(i)| - \ell, & \text{if } e = (s, v_i) \\ 1 & \text{otherwise.} \end{cases} \tag{2.32}$$

The only difference to Section 2.3.1 is that we do not have any outliers. We use $G = (V, E)$ to refer to G_τ as τ is clear from context. We now compute an integral maximum s-t-flow f on G. According to f we can reassign points to different clusters.

Analogous to Lemma 52 we obtain the following lemma.

Lemma 61. *Let f be an integral maximal s-t-flow on G. It is possible to reassign j to i for all edges (w_j, v_i) with $f((w_j, v_i)) = 1$. The resulting solution has a maximum radius of at most $r + 2\tau$. If f saturates all edges of the form (v_i, t), then the solution is feasible.*

In case f saturates all edges of the form (v_i, t) we reassign points according to Lemma 61 and return the new clustering. Otherwise, we look at the residual network G_f of f on G. We define V', $S(V')$, k'', $P(V')$ and $P_A(V')$ as before, i.e., V' is the set of nodes in G_f which cannot be reached from s, $S(V')$ is the set containing the centers of the clusters belonging to V', $k'' = |S(V')|$ and $P(V')$ and $P_A(V')$ denote the set of points that belong to V' and the set of points adjacent to V'. As before, we obtain the following lemma.

Lemma 62. *Any clustering on P with maximum radius at most τ that respects the lower bounds uses fewer than k'' clusters to cover all points in $P(V')$.*

In case we have $\tau \geq \mathsf{opt}$ this implies that the optimal solution covers all points in $P(V')$ with fewer than k'' clusters. An α-approximation on the point set $P(V')$ with at most $k'' - 1$ clusters which abides only the upper bounds is then an α-approximation for $P(V')$. We now use A again to compute a new solution without the lower bound: Let $\mathcal{C}' = (S', \phi')$ be an α-approximation for the capacitated k-supplier problem on $P(V')$, L, $k'' - 1$, u. Let $r' = \max_{j \in P(V')} d(j, \phi'(j))$. Note that in case $\tau < \mathsf{opt}$, it can happen that no such clustering exists or that we obtain $r' > \alpha \cdot \tau$. We then return $\tau < \mathsf{opt}$.

Otherwise we replace replace $S(V')$ by S' in $\mathcal{C}$ and adjust ϕ accordingly to obtain $\mathcal{C}_1 = (S_1, \phi_1)$ with $S_1 = (S \setminus S(V')) \cup S'$ and

$$\phi_1(p) = \begin{cases} \phi'(p) & \text{if } p \in P(V') \\ \phi(p) & \text{otherwise.} \end{cases} \tag{2.33}$$

Note that it is possible, that S' contains a center that already is part of $S \setminus S(V')$. Without capacities we could, in that case, simply combine the corresponding clusters. Unfortunately it can happen that combining these clusters would create a cluster that violates the upper bound. In that case we will assume that both centers are separate points at the same location. As we assumed the capacities to be soft this does not create a problem.

Lemma 63. *In case we did not return $\tau < \mathsf{opt}$, $\mathcal{C}_1$ is a solution for the soft capacitated k-supplier problem on P, L, k, u and we have $r_1 = \max_{j \in P} d(j, \phi_1(j)) \leq \alpha \cdot \mathsf{opt}$.*

We iterate the previous process with new clustering $\mathcal{C}_1$ until we either determine $\tau < \mathsf{opt}$ or the reassignment of points according to Lemma 61 yields a feasible solution. Since the number of clusters is reduced in each iteration, the process terminates after at most k iterations. □

Lemma 64. *Given the value of the optimal solution* opt *and an* $(\alpha+2)$*-approximation for the private soft uniform capacitated* k*-supplier problem, obtained through an* α*-approximation for the soft uniform capacitated* k*-supplier problem and a flow-network as described above, we can compute a* $(2\alpha+2)$*-approximation for the private uniform capacitated* k*-center problem and a* $(2\alpha+3)$*-approximation for the private uniform capacitated* k*-supplier problem in polynomial time.*

Proof. Let $P(i), i \in S$ denote the computed clusters before we reassigned points based on the computed network flow. We call them initial clusters. It can happen that some $i_1 \neq i_2 \in S$ represent the same points $i \in L$. Without capacities we could simply merge all initial clusters with the same center, but with capacities we might need to change some of the centers in order to obtain a feasible solution. In the center variant we choose a point $j_i \in P(i)$ for each $i \in S$ and declare j_i as the new center of $P(i)$. As for two points $j_1, j_2 \in P(i)$ we have $d(j_1, j_2) \leq 2\alpha\,\mathsf{opt}$, the maximal cluster radius after reassigning points based on the network flow is at most $2\alpha + 2$. In the supplier version we want each initial cluster to have a center that is at most opt away from one of its points, i.e., one of the centers to which one of its points can potentially be assigned to in the optimal solution, such that no two initial clusters have the same center. In case such an assignment exists we can compute it with a maximum matching algorithm that tries to match the initial clusters to the allowed centers. It can happen that no such assignment exists. In this case we compute a maximal matching which does not match all initial clusters. For every matched cluster $P(i)$ let l_i denote the location it is matched with. Before we show that it is possible to distribute all points from the unmatched clusters to close matched clusters without violating the capacity constraint, we want to satisfy the lower bound for all matched clusters.

As we used the initial clusters it can still happen that some clusters contain less than ℓ points and we will now show that it is possible to reassign points in order to satisfy the lower bounds.

Let G denote the last flow network used as shown in the proof of Lemma 60 and let F denote its flow. As F is a feasible flow for G there also exists a feasible flow for the reduced network, which ignores demands and incoming edges of all unmatched cluster. Such a flow must satisfy the lower bounds for all matched clusters.

Let us first define the matching instance in detail. Let $V_c = \{v_i \mid i \in S\}$ contain a node for every initial cluster and let $V_L = \{w_l \mid l \in L\}$ contain a node for every feasible location. Let E denote the set of edges, then we have $(v_i, w_l) \in E$ if and only if there exists a point $j \in P(i)$ such that $d(j,l) \leq \mathsf{opt}$. Then for each point $j \in P(i)$ the edge between v_i and $w_{\phi_{\mathsf{opt}}(j)}$ must be part of E. In case the graph $\bar{G} = (V_c \cup V_L, E)$ does not contain a matching that matches all nodes in V_c there must exist $V' \subseteq V_c$ such that the neighborhood $N(V') \subseteq V_L$ contains less nodes than V', because $N(V')$ must contain w_l for every $l \in L$ for which there exists a point $j \in \bigcup_{v_i \in V'} P(i)$ with $\phi_{\mathsf{opt}}(j) = l$. This implies that the optimal solution uses fewer clusters to covers all points in $\bigcup_{v_i \in V'} P(i)$. We compute a maximal matching M in $\bar{G}$. For each $i \in S$, for which v_i is matched in M we denote the corresponding location by l_i. For every unmatched $v_i \in V_c$ we know that for all $j \in P(i)$ the corresponding center in the optimal solution $\phi_{\mathsf{opt}}(j)$ must be matched to some other cluster, i.e., $\phi_{\mathsf{opt}}(j) = l_{i'}$ for some $i' \neq i \in S$. As the matching we computed is maximal we know that G cannot contain a path which alternates between matching and non-matching edges and contains a node belonging to an unmatched cluster as well as an unmatched location. We allow each point $j \in P$ to be reassigned to any center $i \in S$ if we have $d(j, l_i) \leq \mathsf{opt}$. We can model this with a flow network $G' = (\{s,t\} \cup V_c, E')$ which contains a directed edge $(v_i, v_{i'})$ in case $P(i')$ is a matched cluster and there exists at least one point $j \in P(i)$ with $d(j, l_{i'}) \leq \mathsf{opt}$. Let the capacity of an edge $(v_i, v_{i'})$ be equal to $|\{j \in P(i) \mid d(j, l_{i'}) \leq \mathsf{opt}\}|$. Additionally we connect s to every unmatched cluster $P(i)$ with capacity $|P(i)|$ and connect every matched cluster $P(i')$ to t with capacity $u - |P(i')|$. The capacities are chosen such that we want to assign every point from every unmatched cluster and never increase the number of points assigned to a matched cluster to more than u. We will now show that a feasible reassignment exists. We start by assigning every point in an unmatched cluster to the center $i \in S$ with $l_i = \phi_{\mathsf{opt}}(j)$. In case such a reassignment would violate the capacity of a cluster, this cluster must contain at least one point which is assigned to a different center in the optimal solution. The location of this center must be part of our matching as we would otherwise have found an alternating path which contains a node belonging to an unmatched cluster as well as an unmatched location. Therefore we could iteratively reassign such a point to the cluster which is matched to the location of its center in the optimal solution. As each such iterative step increases the number of points assigned to the same center as in the optimal solution, this process must stop after at most n iterations. This shows that there exists a feasible reassignment and we can therefore find one by computing a maximum flow in G'. The radius in each cluster is then at most $(2\alpha + 1)\,\mathsf{opt}$ as the distance between two points in the same initial cluster is at most $2\alpha\,\mathsf{opt}$, the assigned center is at most opt away from one of the points and all points reassigned at the end have a distance of at most opt to its center. As this reassignment can only have increases the number of points in a cluster, a feasible reassignment that enforces the lower bound for each cluster will still be possible. This leads to a radius of at most $(2\alpha + 3)\,\mathsf{opt}$.

□

If we adjust the 5-approximation for the soft uniform capacitated k-center problem by Khuller and Sussman[66] to the supplier variant, we obtain a 6-approximation. With that we get the following result.

Corollary 65. *We can compute a* 14*-approximation for instances of the private k-center problem with uniform capacities in polynomial time and a* 15*-approximation for the supplier variant.*

If the capacities are soft, then we can compute an 8*-approximation for both the center and the supplier variant.*

Fair and private k-center We consider exact fairness with an arbitrary number of colors.

As in Observation 34 we let $b_h = \frac{|col_h(P)|}{gcd(|col_{h'}(P)||h' \in Col)}$ for every $h \in Col$ and use the fact that in every feasible clustering every cluster contains a multiple of $b := \sum_{h \in Col} b_h$ points and the same multiple of b_h points with color h.

We can therefore assume without loss of generality that in instances of the private and fair k-center problem the lower bound is an integer multiple of b, i.e., $\ell = \ell' \cdot b$ for some $\ell' \in \mathbb{N}$.

Recall that with exact fairness a subset $F \subseteq P$ is a fairlet of P, exactly if for every $h \in Col$ F contains exactly b_h points with color h, i.e., for all $h \in Col$ we have $|col_h(F)| = b_h$.

We look at private and fair clustering in two layers, the first layer consists of computing a fairlet decomposition $\mathcal{F} = \{F_i \mid i \in I\}$ of P and the second layer consists of clustering these fairlets. As we know that $\mathcal{F}$ must contain $\frac{n}{b}$ fairlets, we assume without loss of generality $I = \{1, \ldots, \frac{n}{b}\}$.

We now show a couple useful properties.

Lemma 66. *Let $\mathcal{F} = \{F_i \mid i \in I\}$ and $\mathcal{G} = \{G_i \mid i \in I\}$ be two fairlet decompositions of P, then there exists a bijective mapping $\pi : I \to I$ such that for each $i \in I$ we have $F_i \cap G_{\pi(i)} \neq \emptyset$.*

Proof. Let $G = (V \cup W, E)$ be a bipartite graph defined by $V = \{v_i, i \in I\}$, $W = \{w_i, i \in I\}$ and $\{v_i, w_j\} \in E \Leftrightarrow F_i \cap G_j \neq \emptyset$. Then the existence of a perfect matching in G is equivalent to the existence of a mapping π as described. If we set costs c to the edges by $c(\{v_i, w_j\}) = |F_i \cap G_j|$ we can see that for every subset of $V' \subseteq V$ and every subset $W' \subseteq W$ the total cost of all edges adjacent to V' is equal to $|V'| \cdot b$ and the total cost of all edges adjacent to W' is equal to $|W'| \cdot b$. Therefore the neighborhood of $V' \subseteq V$ contains at least $|V'|$ nodes. Hall's "'Marriage Theorem"' [51] then concludes the proof. $\square$

Lemma 67. *Let $\mathcal{G} = \{G_i \mid i \in I\}$ be an arbitrary fairlet decomposition of P with a maximum diameter $d = \max_{i \in I} \max_{j,j' \in G_i} d(j, j')$ and assume that there exists a fair and private clustering $\mathcal{C}'$ of P with radius c. Then there exists a fair and private clustering $\mathcal{C}$ of P with a radius of at most $c + d$ which satisfies that for each $i \in I$ all points in G_i are part of the same cluster.*

Proof. Let $\mathcal{F} = \{F_i \mid i \in I\}$ a fairlet decomposition such that each fairlet $F_i \in \mathcal{F}$ is a subset of a cluster in $\mathcal{C}'$. Lemma 66 shows that there exists a bijective mapping $\pi : I \to I\}$ such that for each $i \in I$ we have $F_i \cap G_{\pi(i)} \neq \emptyset$. For all $i \in I$ we replace F_i by $G_{\pi(i)}$ in its cluster in $\mathcal{C}'$ to create the new clustering $\mathcal{C} = \{S, \phi'\}$. Formally ϕ' is defined as follows. For all $i \in I$ and $j \in G_{\pi(i)}$ we have $\phi'(j) = \phi(j')$ for some $j' \in F_i$. Note that ϕ' is well defined since for all $i \in I$ we have $\phi(j) = \phi(j')$ for all $j, j' \in F_i$.

Since replacing a fairlet with a different fairlet does not change the number of points in a cluster $\mathcal{C}$ is a fair and private clustering and by construction satisfies that for each $i \in I$ all points in G_i are part of the same cluster.

We know that for all $i \in I$ we have $F_i \cap G_{\pi(i)} \neq \emptyset$. Let $j' \in F_i \cap G_{\pi(i)}$ then we have $d(j', \phi(j')) \leq c$ and $d(j, j') \leq d$ for all $j \in G_{\pi(i)}$. By the triangle inequality we immediately obtain $d(j, \phi'(j) = \phi(j')) \leq c + d$. $\square$

We use an approximation algorithm for the fair k-center problem to obtain a fairlet decomposition. We use this fairlet decomposition as explained above in order to obtain an approximation algorithm for the private and fair k-center problem and obtain the following lemma.

Lemma 68. *Assume that there exists an α-approximation algorithm A for the fair k-center problem and a β-approximation algorithm B for the private k-center problem. Then we can compute a $(2\beta\alpha + 2\beta + \alpha)$-approximation for the private and fair k-center problem, in polynomial time. In case algorithm B also works for the case $P \subseteq L$, the approximation ratio improves to $(\beta\alpha + \beta + \alpha)$*

Proof. We assume that the instance P, L, k, ℓ Col, col is solvable, i.e., $n \geq \ell$. We first use A to compute a solution without the lower bound: Let $\mathcal{C} = (S, \phi)$ be an α-approximation for the fair k-center problem on P, L, k, Col, col.

Again let $k' = |S|$ and let $r = \max_{j \in P} d(j, \phi(j))$ be the largest distance of any point to its assigned center.

We use $\mathcal{C}$ to compute an arbitrary fairlet decomposition $\mathcal{F} = \{F_i \mid i \in I\}$ of P such that each fairlet F_i is a subset of a cluster in $\mathcal{C}$. By construction the maximum diameter of any fairlet in $\mathcal{F}$ is at most $2\alpha\,\mathsf{opt}$. Lemma 67 then tells us that there exists a clustering with maximal radius at most $(2\alpha + 1)\,\mathsf{opt}$, which satisfies that for each $i \in I$ all points in F_i are part of the same cluster.

The idea is that we now want to compute a solution that satisfies the lower bound and assigns all points in the same fairlet to the same cluster.

We assign a center to each of the fairlets. When we use the same center that has been used in $\mathcal{C}$ we obtain that in each fairlet the distance of a point to the center is at most $\alpha\,\mathsf{opt}$.

We declare the center of each fairlet as its representative and use algorithm B to compute a solution that satisfies the lower bound $\ell' = \ell/b$ on the set of representatives. We then replace each of the representatives by the points of the fairlet it represents and return the result as our clustering. It is easy to see that satisfying a lower bound of ℓ' on the set of representatives and satisfying the lower bound of ℓ on the final cluster are essentially the same for clusters composed as a union of these fairlets. Therefore this approach returns a feasible clustering.

In order to find an upper bound on the maximal radius of the obtained clustering, we at first find an upper bound on the maximal radius of the best private solution in which every cluster is a union of our computed fairlets.

When reducing the points in each cluster to the fairlet representatives, the radius of each cluster can be at most $(\alpha + 1)\,\mathsf{opt}$. In case B works with $P \subseteq L$, we can use B to compute a β-approximation on the set of representatives as points and the set of original locations L as possible centers to obtain a clustering on the representatives with maximal radius at most $\beta(\alpha + 1)\,\mathsf{opt}$. Replacing each of the representatives by the corresponding fairlet yields a clustering with a maximal radius of at most $\beta(\alpha + 1)\,\mathsf{opt} + \alpha\,\mathsf{opt}$. In case B only works for instances with $P = L$, the maximal radius of the best solution on the set of representatives is at most $2(\alpha + 1)\,\mathsf{opt}$ and obtain a final clustering with radius at most $2\beta(\alpha + 1)\,\mathsf{opt} + \alpha\,\mathsf{opt}$.

□

Instead of using an approximation algorithm for the fair k-center problem in order to obtain a fairlet decomposition we could also use an approximation algorithm for the fairlet decomposition problem.

Lemma 69. *Assume that there exists an α-approximation algorithm A for the fairlet decomposition problem and a β-approximation algorithm B for the private k-center problem. In case B works in the case $P \subseteq L$, then we can compute a $(\beta\alpha + \beta + \alpha)$-approximation for the private fair k-center problem in polynomial time. For the supplier version the approximation ratio increases by* 1.

Proof. It is only left to show the supplier case, as everything else follows directly from the previous discussion. As B works in case we have $P \subseteq L$, we use the explained approach on the slightly changed instance, in which we use $P \cup L$ as the set of allowed locations. As allowing more solutions cannot increase the maximum radius in the optimal solution, the solution we compute on the altered instance can have a maximal radius of at most $(\beta\alpha + \beta + \alpha)\,\mathsf{opt}$. This solution might not be feasible as a computed centers could potentially be in $P \setminus L$. We exchange all such centers by the closest feasible location in L. This can increase the maximum radius by at most opt as the distance between each point and its closest location is at most opt. It can happen that now multiple clusters have the same center. If we merge all clusters with the same center, we obtain a feasible solution. □

Corollary 70. *We can compute a* 17*-approximation for instances of the private and fair k-center problem and a* 23*-approximation for instances of the private and fair k-supplier problem in polynomial time.*

If $b_h = 1$ for at least one color $h \in Col$, the approximation ratios improve to 8 *and* 9.

Proof. All results follow from Lemma 68, Lemma 69 together with the 2-approximation algorithm for the private k-center problem [3], which works in case we have $P \subseteq L$, and an approximation algorithm for the fairlet decomposition problem. In case $b_h = 1$ for some $h \in Col$ we use the 2-approximation algorithm from Section 2.2.1 and for the general case use the 5- and 7-approximation algorithms from Theorem 44 and Theorem 45. □

Privacy, fairness and capacities In this section we consider instances of the private capacitated and fair k-center problem.

We let P, L, k, u, Col, col, ℓ be an instance of the private capacitated and fair k-center problem.

As every fair cluster can be partitioned into fairlets, we again assume without loss of generality that the lower bound ℓ as well as all upper bounds $\{u(j) \mid j \in P\}$ are integer multiples of b.

We again look at fair clustering in two layers, the first layer consists of computing a fairlet decomposition $\mathcal{F} = \{F_i \mid i \in I\}$ of P and the second layer consists of clustering these fairlets.

Notice that Lemma 67 also works for fair, private and capacitated clustering with the exact same proof.

The idea is again to compute a fairlet decomposition of $\mathcal{F} = \{F_i \mid i \in I\}$ with small diameter and then use an approximation algorithm to compute a private and capacitated clustering on a set of points $\{f_i \mid i \in I\}$ which represents the fairlets.

Through the construction we obtain the following lemma analogous to Lemma 68.

Lemma 71. *Assume that there exists an α-approximation algorithm A for the private capacitated k-center problem. Assume that there exists a β-approximation algorithm B for the fairlet decomposition problem. Then we can compute an $(2\beta\alpha + 2\beta + \alpha)$-approximation of the private capacitated and fair k-center problem in polynomial time. In case algorithm B also works for the case $P \subseteq L$, the approximation ratio improves to $(\beta\alpha + \beta + \alpha)$.*

Corollary 72. *We can compute an $O(1)$-approximation for instances of the private capacitated and fair k-center/k-supplier in polynomial time.*

Proof. We use the 13-approximation algorithm by [32] for the k-supplier problem with uniform lower bound and non-uniform upper bounds and the 9-approximation algorithm by [32] for the k-supplier problem with uniform lower bound and uniform upper bounds for both the k-center and k-supplier variant, as they work for the case $P \subseteq L$.

Together with Lemma 71 and the 5-approximation algorithm from Theorem 44 this yields an approximation ratio of $13 \cdot 5 + 13 + 5 = 83$. In case of a uniform upper bound this reduces the approximation ratio to 59.

In case $b_h = 1$ for some $h \in Col$ the 2-approximation algorithm for the fairlet decomposition problem from Section 2.2.1 improves the approximation ratio to 41 for non-uniform upper bounds and 29 for uniform upper bounds.

Together with the 7-approximation algorithm from Theorem 45 we obtain the following approximation ratios for the fair k-supplier problem with non-uniform upper bounds and uniform lower bounds.

In the general case we obtain approximation ratios of 111 and 79 and in case $b_h = 1$ for some $h \in Col$ we obtain approximation ratios of 41 and 29. $\square$

2.3.3 Strongly Private k-Center

Similar to fairness we assume that instances of the strongly private k-center problem contain, in addition to P, L and k, a set of colors Col and a function $col : P \to Col$ which assigns a color to each of the points. In order to preserve the privacy although additional information about each point is know we demand that each cluster contains enough representatives of each color.

Formally, the strongly private k-center problem consists of an instance of the k-center problem together with a set of colors Col, a function $col : P \to Col$ and a lower bound ℓ_h for each color $h \in Col$, where the problem is to compute a set of centers $S \subseteq L$ with $|S| \leq k$ and an assignment $\phi : P \to S$ of the points to the selected centers that satisfies $\ell_h \leq |col_h(P(i))|$ for all $h \in Col$ and all $i \in S$ and minimizes

$$\max_{j \in P} d(j, \phi(j)).$$

We again adjust our method from Section 2.3.1 in order to apply it to the strongly private k-center problem and obtain the following lemma.

Theorem 73. *Assume that there exists an α-approximation algorithm A for the k-supplier problem. Then we can compute an $(\alpha + 2)$-approximation for the strongly private k-center problem in polynomial time.*

Proof. Let P, L, k, Col, col, $\{\ell_h \mid h \in Col\}$ be an instance of the strongly private k-supplier problem.

Analogous to Section 2.3.1 we use threshold graphs with threshold τ and show that for any given $\tau \in \mathbb{R}$, the algorithm has polynomial run time, and, if τ is equal to opt, the value of the optimal solution, computes an $(\alpha + 2)$-approximation. Since we know that the value of the optimal solution is equal to the distance between a point and a location, we test all $O(|P||L|)$ possible distances for τ and return the best feasible clustering returned by any of them. The main proof is the proof of Lemma 74 below. The lemma then concludes the proof. $\square$

We now describe the procedure for a fixed value of $\tau > 0$.

Lemma 74. *Assume that there exists an α-approximation algorithm A for the k-supplier problem. Let P, L, k, Col, col, $\{\ell_h \mid h \in Col\}$ be an instance of the strongly private k-supplier problem, let $\tau > 0$ and let opt denote the maximum radius in the optimal feasible clustering for P, L, k, Col, col, $\{\ell_h \mid h \in Col\}$. We can in polynomial time compute a feasible clustering with a maximum radius of at most $(\alpha + 2)\tau$ or determine $\tau < \mathsf{opt}$.*

Proof. The algorithm first uses A to compute a solution without the lower bounds: Let $\mathcal{C} = (S, \phi)$ be an α-approximation for the k-supplier problem on P, L, k. Again let $k' = |S|$ and let $r = \max_{j \in P} d(j, \phi(j))$ be the largest distance of any point to its assigned center. If we have $r > \alpha \cdot \tau$, we return $\tau < \mathsf{opt}$. Given $\mathcal{C}$ and τ, we create, similar to Section 2.3.1, a threshold graph $G_{\tau,h} = (V_{\tau,h}, E_{\tau,h})$ for every $h \in Col$ by

$$V_{\tau,h} = \{v_i \mid i \in S\} \cup \{w_j \mid j \in col_h(P)\} \cup \{s, t\} \text{ and} \tag{2.34}$$

$$E_{\tau,h} = \{(v_i, w_j) \mid j \in col_h(P(i))\} \cup \{(w_j, v_i) \mid j \in col_h(P) \setminus P(i) \wedge d(j, P(i)) \leq 2\tau\} \tag{2.35}$$

$$\cup\{(s,v_i) \mid |col_h(P(i))| - \ell_h > 0\} \cup \{(v_i,t) \mid |col_h(P(i))| - \ell_h < 0\}. \tag{2.36}$$

We define the capacity functions $cap_i : E_{\tau,i} \to \mathbb{R}$ by

$$cap(e) = \begin{cases} \ell_h - |col_h(P(i))|, & \text{if } e = (v_i,t) \\ |col_h(P(i))| - \ell_h, & \text{if } e = (s,v_i) \\ 1 & \text{otherwise.} \end{cases} \tag{2.37}$$

The only difference to Section 2.3.1 is that we do not have outliers and create a separate threshold graph for every color.

We use $G_h = (V_h, E_h)$ to refer to $G_{\tau,h}$ as τ is clear from context. We now compute integral maximum s-t-flows f_h on G_h. According to f_h we can reassign points of color h to different clusters.

Analogous to Lemma 52 we obtain the following lemma.

Lemma 75. *Let f_h be an integral maximal s-t-flow on G_h. It is possible to reassign j to $P(i)$ for all edges (w_j, v_i) with $f((w_j,v_i)) = 1$. The resulting solution has a maximum radius of at most $r + 2\tau$. If f_h saturates all edges of the form (v_i,t), then the solution contains at least ℓ_h points of color h in every cluster.*

If for all $h \in Col$, f_h saturates all edges of the form (v_i,t) in G_h, then we reassign points according to Lemma 75 and return the new clustering. Note that for each $h \in Col$ f_h would only suggest to reassigns points of color h. Therefore the reassignments according to the flows computed for different colors do not interfere with each other. Otherwise chose an arbitrary color $h \in Col$ such that f_h does not saturate all edges of the form (v_i,t) in G_h. We again look at the residual network G_{f_h} of f_h on G_h. We define V', $S(V')$, k'', $P(V')$ and $P_A(V')$ as before, i.e., V' is the set of nodes in G_{f_h} which cannot be reached from s, $S(V')$ is the set containing the centers of the clusters belonging to V', $k'' = |S(V')|$ and $P(V')$ denotes the set of points that belong to V'. As before, we obtain the following lemma.

Lemma 76. *Any clustering on P with maximum radius at most τ that contains at least ℓ_h points of color h in every cluster uses fewer than k'' clusters to cover all points in $P(V')$.*

In case we have $\tau \geq \mathsf{opt}$ this implies that the optimal solution covers all points in $P(V')$ with fewer than k'' clusters. An α-approximation on the point set $P(V')$ with at most $k''-1$ clusters is therefore an α-approximation for $P(V')$. We now use A again to compute a new solution without the lower bounds: Let $\mathcal{C}' = (C', \phi')$ be an α-approximation for the k-supplier problem on $P(V')$, L, $k''-1$. Let $r' = \max_{j \in P(V')} d(j, \phi'(j))$. Note that in case $\tau < \mathsf{opt}$, it can happen that no such clustering exists or that we obtain $r' > \alpha \cdot \tau$. We then return $\tau < \mathsf{opt}$. Otherwise we replace $S(V')$ by S' in $\mathcal{C}$ and adjust ϕ accordingly to obtain $\mathcal{C}_1 = (S_1, \phi_1)$ with $S_1 = (S \setminus S(V')) \cup S'$ and

$$\phi_1(j) = \begin{cases} \phi'(j) & \text{if } j \in P(V') \\ \phi(j) & \text{otherwise.} \end{cases} \tag{2.38}$$

Note that it can happen that S' and $S \setminus S(V')$ share a center. In that case we simply merge the corresponding clusters.

Lemma 77. *In case we did not return $\tau < \mathsf{opt}$, $\mathcal{C}_1$ is a solution for the k-supplier problem P, L, k and we have $r_1 = \max_{j \in P} d(j, \phi_1(j)) \leq \alpha \cdot \tau$.*

We iterate the previous process with the new clustering $\mathcal{C}_1$ until we either determine $\tau < \mathsf{opt}$ or the reassignment of points according to Lemma 75 yields a feasible solution. Since each iteration reduces the number of clusters, the process terminates after at most k iterations. Note that the color $h \in Col$, according to which we define the set V' as the set of nodes in G_{f_h} which cannot be reached from s, does not have to be the same for every iteration, instead in each iteration the color can be chosen arbitrarily among all colors $h \in Col$ for which f_h does not saturate all edges of the form (v_i, t) in G_h. □

Corollary 78. *We can compute a 5-approximation for instances of the strongly private k-supplier problem in polynomial time.*

Proof. Follows from Theorem 73 together with the 3-approximation algorithm for k-supplier in [47]. □

Corollary 79. *We can compute a 4-approximation for instances of the strong private k-center problem in polynomial time.*

Proof. Note that in the k-center variant we have $P(V') \subseteq L$ every time we need to recompute a clustering on a set $P(V') \subseteq P$. We will show that the 2-approximation algorithm for the k-center problem in [47] also works on instances with $P \subseteq L$. Theorem 73 then concludes the proof. □

2.3.4 (Metric) Facility Location

Let $P = L$, $f \in \mathbb{N}$, $\ell \in \mathbb{N}$, $u \in \mathbb{N}$ with $2\ell \leq u$ be an instance of the private capacitated facility location problem with uniform upper and uniform lower bounds and uniform facility opening costs. Let $\mathcal{C} = (S, \phi)$ be a γ-approximation for the private facility location problem on P, L, f, ℓ. We set $k' = |S|$. We define an instance where all points are translated to their centers. So we let P' contain $|P(i)|$ copies of i for each $i \in S$. More precisely, place a point p_j at location $\phi(j)$ for all $j \in P$ and call the resulting set P'. Note that we use P' in order to simplify the analysis and although it is not fully supported by the definition, where P is a *subset* of X we will use the same terminology, when we talk about clusterings on P'.

Lemma 80. *Let $\mathcal{C}' = (S', \phi')$ be any clustering for P'. The clustering $\mathcal{C}'' = (S'', \phi'')$ on P with $S'' = S'$ and $\phi''(j) = \phi'(p_j)$ for all $j \in P$. Then*

$$\sum_{j \in P} d(j, \phi''(j)) \leq \sum_{j \in P} d(j, \phi(j)) + \sum_{j' \in P'} d(j', \phi'(j')).$$

Proof. Let $j \in P$ be any point. Then

$$d(j, \phi''(j)) = d(j, \phi'(p_j)) \leq d(j, p_j) + d(p_j, \phi'(p_j))$$

by the triangle inequality. Summing this over all $j \in P$, we get that

$$\sum_{j \in P} d(j, \phi''(j)) \le \sum_{j \in P} d(j, \phi(j)) + \sum_{j' \in P'} d(j', \phi'(j')).$$

□

Let $P'(i)$ denote the set $\{p_j \mid j \in P(i)\}$. Assume that we have *soft capacities*, i.e., that we can open a center multiple times. Then we compute a solution to the upper bounded facility location problem on P', L, f, u in the following way. Firstly, we open a center at every location $i \in S$. This costs $f \cdot k'$ opening cost. To the center $i \in S$, we assign $|P(i)| \mod u$ points from $P'(i)$. After this step, the number of points in $P'(i)$ that are not yet assigned is a multiple of u (this is true for all $i \in S$). We can thus satisfy all their demand by opening at most n/u additional centers. Since any feasible solution opens at least n/u centers, $n/u \le k_{\mathsf{opt}}$. Thus we additionally pay at most $f \cdot k_{\mathsf{opt}}$ for opening the missing centers, where k_{opt} denotes the number of centers the optimal solution opens. Again, assigning the points costs nothing in P'.

Corollary 81. *There is a solution to the upper bounded facility location problem on P, L, f, u with soft capacities which costs at most*

$$\sum_{j \in P} d(j, \phi(j)) + f \cdot (k' + k_{\mathsf{opt}}).$$

Proof. Follows from the above discussion and Lemma 80. □

We want to reconcile this solution with the lower bound solution. The facilities from the second step are valid because they contain $u \ge \ell$ points. The facilities from the first step might be invalid. We will reassign some of the points to different centers at the same location. This costs nothing.

For each $i \in S$ there are two cases. In the first case we only opened one center at the location i. In that case we have assigned $|P(i)| \ge \ell$ points to that location. Otherwise there exists at most one center at location i to which we assigned fewer than u points. In case such a center exists we call it i_1 and let b_1 denote the number of points assigned to i_1. To every other centers at location i we must have assigned exactly u points. Because we have opened at least two centers at location i, we can chose another center $i_2 \ne i_1$ at location i. In case we have $b_1 \ge \ell$ we do nothing. Otherwise we reassign $u/2$ points from i_2 to i_1, ensuring that we get two facilities which have at least $u/2 \ge \ell$ and at most u points; the rest of the facilities at this location will remain unchanged. Thus, at no additional cost, we get a solution that respects both upper and lower bounds.

Corollary 82. *There is a solution to the private soft capacitated facility location problem on P, L, f, u, ℓ with $2\ell \le u$ which costs at most*

$$\sum_{j \in P} d(j, \phi(j)) + f \cdot (k' + k_{\mathsf{opt}}).$$

Now assume we do not have soft capacities. Then we consider the solution computed by the above soft capacity algorithm; it partitions the points into $k \leq k' + k_{\mathsf{opt}}$ clusters $\{C_i \mid 1 \leq i \leq k\}$. For each cluster C_i, we let $\arg\min_{j \in C_i} \sum_{j' \in C_i} d(j', j)$ be its center. Say that the points in C_i were previously assigned to center $c \notin C_i$ and are now assigned to c'. Furthermore, let c'' be the point in C_i that is closest to c. Then we have

$$\sum_{j \in C_i} d(j, c') \leq \sum_{j \in C_i} d(j, c''),$$

and for each point $j \in C_i$,

$$d(j, c'') \leq d(j, c) + d(c'', c) \leq 2d(j, c)$$

because c'' is the closest point to c in C_i. This implies

$$\sum_{j \in C_i} d(j, c') \leq 2 \sum_{j \in C_i} d(j, c),$$

so the assignment cost goes up by a factor of at most two.

Corollary 83. *There is a solution to the private capacitated facility location problem on P, L, f, u, ℓ with $2\ell \leq u$ which costs at most*

$$2 \sum_{j \in P} d(j, \phi(j)) + f \cdot (k' + k_{\mathsf{opt}}) \leq 2\gamma \, \mathsf{opt}_L + \mathsf{opt}_U \leq (2\gamma + 1) \, \mathsf{opt},$$

where opt_L is the cost of an optimal solution for the private facility location problem on P, L, f, ℓ and opt_U is the cost of an optimal solution for the capacitated facility location problem on P, L, f, u.

Chapter 3

Parity Games

This chapter is devoted to parity games. We prove the results discussed in Section 1.3 and describe the corresponding algorithms in full detail. We start by stating some useful definitions and known fundamental properties of parity games. After that we describe several kernelization methods for general and bipartite parity games. Then we explain a very simple exponential time algorithm and explain how we use its overall concept with some specific adjustments to obtain our new algorithms. We conclude the chapter with the run time analysis of our new algorithms.

3.1 Fundamental Properties and Notation of Parity Games

A parity game $G = (V_0 \uplus V_1, E, p)$ consists of a directed graph $(V_0 \uplus V_1, E)$, where V_0 is the set of *even* nodes and V_1 is the set of *odd* nodes, and a priority function $p : V_0 \cup V_1 \to \mathbb{N}_0$. We often abuse notation and also refer to $(V_0 \uplus V_1, E)$ as the graph G. For each node $v \in V(G)$, we denote by $N_G^+(v) = \{w \in V_0 \uplus V_1 \mid (v, w) \in E\}$ and $N_G^-(v) = \{u \in V_0 \uplus V_1 \mid (u, v) \in E\}$ the set of out-neighbors and in-neighbors of v in G, respectively.

Two standard assumptions about parity games are (1) that G is bipartite with $E \subseteq (V_0 \times V_1) \cup (V_1 \times V_0)$, and (2) that each node $u \in V$ has at least one outgoing edge $(u, v) \in E$. The first assumption is often made because it is easy to transform a non-bipartite instance into a bipartite instance. However, the usual transformation increases the number of nodes in V_i by an amount of $|\{v \in V_{1-i} \mid N_G^-(v) \cap V_{1-i} \neq \emptyset\}|$, and can therefore increase the parameter $k = \min\{|V_0|, |V_1|\}$ significantly. We therefore consider bipartite and non-bipartite instances separately in Theorem 11.

We write $n = |V(G)|$, $m = |E|$ and $p = |\{p(v) \mid v \in V(G)\}|$. The game is played by two players, the *even* player (or player 0) and the *odd* player (or player 1). The game starts at some node $v_0 \in V(G)$. The players construct an infinite path (a *play*) as follows. Let u be the last node added so far to the path. If $u \in V_0$, then player 0 chooses an edge $(u, v) \in E$. Otherwise, if $u \in V_1$, then player 1 chooses an edge $(u, v) \in E$. In either case, node v is added to the path and a new edge is then chosen by either player 0 or player 1. As each node has at least one outgoing edge, the path constructed can always be continued. Let $v_0, v_1, v_2, \ldots$ be the infinite path constructed by the two players and let $p(v_0), p(v_1), p(v_2), \ldots$ be the sequence of the priorities of the nodes on the path. Player 0 *wins* the game if the largest priority seen infinitely often is even, and player 1 wins if the largest priority seen infinitely often is odd.

We will define $p_1(v)$ as $p(v)$ if $p(v)$ is odd and as $-p(v)$ if $p(v)$ is even. This allows us to say that, in case $p_1(v) > p_1(u)$ for some $v, u \in V$, player 1 *prefers* $p(v)$ over $p(u)$. Observe that removing an arbitrary finite prefix of a play in a parity game does not change the winner; we refer to this property of parity games as *prefix independence*.

Definition 84. *A* strategy *for player $i \in \{0, 1\}$ in a game G, is a function s_i that maps every finite walk $v_0, v_1, \ldots, v_k$ in G that ends in a node $v_k \in V_i$, an edge $(v_k, v_{k+1}) \in E$. A strategy s_i is* positional *if the edge $(v_k, v_{k+1}) \in E$ chosen depends only on the last node v_k visited and is independent of the prefix path $v_0, v_1, \ldots, v_{k-1}$.*
A strategy s_i is winning *(for player i) from a start node v_0 if following this strategy ensures that player i wins the game, regardless of which strategy is used by the other player.*

The fundamental determinacy theorem for parity games [37, 48] says that for every parity game G and every start node v_0, either player 0 has a winning strategy or player 1 has a winning strategy. Furthermore, if a player has a winning strategy from some node in a parity game, then she also has a winning positional strategy from this node. From now on we will therefore, unless stated differently, assume every strategy to be positional. Given positional strategies s_0 on V_0 and s_1 on V_1 and a start node $v_0 \in V$ the infinite path starting in v_0 corresponding to these strategies consists of a finite prefix and an infinite recurrence of a cycle $C = C(s_0, s_1, v_0)$. We call C the cycle *corresponding to* s_0, s_1, v_0 and say that s_0 and s_1 *create* C. The parity of the highest priority $p(u)$ of all nodes $u \in V(C)$ in cycle C then determines the winner of the game.

Definition 85. *The* winning set *of player $i \in \{0, 1\}$ is the set $\mathsf{win}_i(G) \subseteq V$ of nodes of the game G from which player i has a winning strategy.*

Definition 86. *For $i \in \{0, 1\}$, an* i-dominion *is a set of nodes $D \subseteq V$ so that player i can win from every node of D, without leaving D and without allowing the other player to leave D.*

An example of an i-dominion is the set $\mathsf{win}_i(G)$, but there may be smaller subsets of $\mathsf{win}_i(G)$ that are i-dominions as well. Although finding i-dominions can be just as hard as finding $\mathsf{win}_i(G)$, searching only for dominions with certain properties (e.g. small dominions) can be easier. In our algorithm we will use the fact that once an i-dominion is found, it can easily be removed from the graph, leaving a smaller game to be solved.

Next, we recall some well-known results about parity games that form the basis of the algorithms for solving parity games by McNaughton [81] and Zielonka [103]. We include them here as our algorithm relies on them as well; for a detailed exposition we refer to Grädel et al. [48]. Fix a parity game $G = (V_0 \uplus V_1, E, p)$.

Definition 87. *For $i \in \{0,1\}$, a set $B \subseteq V(G)$ is i-*closed *if for every $u \in B$ the following holds (we use the notation $\neg i$ for the element $1 - i \in \{0,1\}$):*

- *If $u \in V_i$, then there exists some $(u,v) \in E$ such that $v \in B$; and*
- *if $u \in V_{\neg i}$, then for every $(u,v) \in E$, we have $v \in B$.*

In other words, a set B is i-closed if player i can always choose to stay in B while simultaneously player $\neg i$ cannot escape from it, i.e., B is a "trap" for player $\neg i$.

Lemma 88. *For each $i \in \{0,1\}$, the set $\mathsf{win}_i(G)$ is i-closed.*

Let $A \subseteq V(G)$ be a set of nodes and let $i \in \{0,1\}$. The *i-reachability set* of A is the set $\mathsf{reach}_i(A)$ of nodes in A together with all nodes in $V(G) \setminus A$ from which player i has a strategy σ to enter A at least once (regardless of the strategy of the other player); we call such a strategy σ an *i-reachability strategy* to A.

Lemma 89. *For $A \subseteq V(G)$ and $i \in \{0,1\}$, the set $V(G) \setminus \mathsf{reach}_i(A)$ is $(\neg i)$-closed.*

We will from now on assume that the graph of the parity game we operate on is encoded as an adjacency list.

Lemma 90. *For every set $A \subseteq V(G)$ and $i \in \{0,1\}$, the set $\mathsf{reach}_i(A)$ can be computed in $O(m)$ time, where $m = |E|$ is the number of edges in the game.*

If $B \subseteq V(G)$ is such that for each node $u \in V(G) \setminus B$ there is an edge (u,v) with $v \in V(G) \setminus B$, then the subgame $G - B$ is the game obtained from G by removing the nodes of B. We will only be using B's for which $V(G) \setminus B$ is an i-closed set for some i. In this case every node in $v \in V(G) \setminus B$ has at least one out-going edge (v,w) with $w \in V(G) \setminus B$ and $G - B$ will therefore be well-defined. The next lemmas show some useful properties of subgames.

Lemma 91. *Let G' be a subgame of G and let $i \in \{0,1\}$. If the node set of G' is i-closed in G, then $\mathsf{win}_i(G') \subseteq \mathsf{win}_i(G)$.*

The next lemma shows that if we know some non-empty subset U of the winning set of some player i in a game G, then computing the winning sets of both players in G can be reduced to computing their winning sets in the smaller game $G - \mathsf{reach}_i(U)$.

Lemma 92. *For any parity game G and $i \in \{0,1\}$, if $U \subseteq \mathsf{win}_i(G)$ and $U^* = \mathsf{reach}_i(U)$, then $\mathsf{win}_i(G) = U^* \cup \mathsf{win}_i(G - U^*)$ and $\mathsf{win}_{\neg i}(G) = \mathsf{win}_{\neg i}(G - U^*)$.*

The next lemma complements Lemma 92 by providing a way to find a non-empty subset of the winning set of player i in a parity game G or to conclude that player $\neg i$ can win from every node in G.

Lemma 93. *Let G be a parity game with largest priority $p_{\max}$ and let $V_{p_{\max}} \subseteq V(G)$ be the set of nodes with priority $p_{\max}$. Let $i = p_{\max} \pmod 2$ and let $G' = G - \mathsf{reach}_i(V_{p_{\max}})$. Then $\mathsf{win}_{\neg i}(G') \subseteq \mathsf{win}_{\neg i}(G)$. Also, if $\mathsf{win}_{\neg i}(G') = \emptyset$, then $\mathsf{win}_i(G) = V$, i.e., player i wins from every node of G.*

3.2 Kernelization of Parity Games

We start by describing some reduction rules for parity games. Theses rules allow us to efficiently compute the winning sets of the original parity game once we know the winning sets of the reduced game.

3.2.1 General Parity Games

Lemma 94. *Any parity game $G = (V_0 \uplus V_1, E, p)$ can be transformed in time $O(pmn)$ into a parity game $G' = (V_0' \uplus V_1', E', p')$ with $V_1' \subseteq V_1$ such that*

- *E' does not contain edges that start and end in V_1', and*
- *for each node $v \in V_0'$ either $N_G^+(v) \subseteq V_1'$ or $N_G^-(v) \subseteq V_1'$, and*
- *$|V_0'| \leq \min\{n + pk, (p+1)^k + pk\}$, where $k = |V_1|$.*

Moreover, G and G' have the same winning sets on V_1' and the winner of the remaining nodes of G can be computed either during the transformation or from the winning sets of G' in linear time.

Proof. We will modify G in multiple steps. We will at first explain, what kind of modifications we want to make and later explain how they can be computed.

We will slightly abuse notation and refer in every step to the parity game that we obtained in the step before as $G = (V_0 \uplus V_1, E, p)$. First we eliminate all edges inside V_1. This can easily be achieved by adding a new node v_e with $p'(v_e) = p(w)$ to V_0 for each edge $e = (v, w) \in E$ with $v, w \in V_1$ and by replacing the edge e by the two edges (v, v_e) and (v_e, w). Since the new node v_e only has a single outgoing edge, this transformation does neither change the winning sets nor the winning strategies.

Next, we remove certain cycles inside V_0 from the game. Let $W_0 \subseteq V_0$ denote all nodes in V_0 that are part of at least one cycle that lies completely inside V_0 and whose highest priority is even. Clearly player 0 can win from all nodes in $\mathsf{reach}_0(W_0)$ by enforcing that such a cycle is entered and never left again. Hence, we can remove $\mathsf{reach}_0(W_0)$ from the game according to Lemma 92. Let $W_1 \subseteq V_0$ denote all nodes that are left in V_0 and from which player 0 cannot reach V_1. Then all paths that start in some node $u \in W_1$ must end in some cycle that is completely contained in V_0. Since we have removed all cycles whose highest priority is even, the maximum priority of this cycle must be odd. Thus, player 1 wins from all nodes in $\mathsf{reach}_1(W_1)$. Hence, we can also remove $\mathsf{reach}_1(W_1)$ from the game according to Lemma 92.

We again use the notation $G = (V_0 \uplus V_1, E, p)$ to refer to the parity game obtained after the previously discussed steps. Since we have removed all cycles from V_0 whose highest priority is even, player 0 loses for sure if she does not leave V_0. Hence, we can assume without loss of generality that the play leaves V_0 from every starting node if player 0 plays an optimal strategy. Then for every node $v \in V_0$ player 0 uses a (possibly empty) path inside V_0 followed by an edge that leads to some node $w \in V_1$. To determine the winning sets of a strategy of player 0 it is not important to know the exact paths player 0 chooses. Rather, it suffices to know for each $v \in V_0$ which node $w \in V_1$ will be reached and what the highest priority on the chosen v-w-path is. To get rid of long paths, we add $p \cdot |V_1|$ new nodes to V_0, one node $v(p', w)$ for each pair of a priority p' and a node $w \in V_1$. Node $v(p', w)$ has a priority p' and its only out-neighbor is w. The winner does not change if player 0 goes from $v \in V_0$ directly to $v(p', w)$ and from there directly to $w \in V_1$ instead of taking some other path from v inside V_0 with maximum priority p', followed by an edge that leads to w. For all such paths we add the corresponding edge $(v, v(p', w))$. We and can then, without changing the winning sets of the game, delete all edges inside V_0, that do not end in one of the new nodes $v(p', w)$. Observe that this ensures that all out-neighbors of the new nodes $v(p', w)$ belong to V_1 while all in-neighbors of the old nodes $v \in V_0$ belong to V_1.

It can be the case that for some pair $(v, w) \in V_0 \times V_1$ there are multiple nodes $v(p', w)$ that can be reached from v. We can assume without loss of generality that if player 0 decides to go from v to w via one of these nodes then she chooses the one that is best for her, i.e., the one with lowest p_1-value. All edges from v to other nodes $v(p', w)$ can be removed.

$|V_0'| \leq n + \min\{m, k^2\} + pk$ follows directly from the previously discussed construction: initially V_0 consists of $n - k \leq n$ nodes, there are at most $\min\{m, k^2\}$ edges inside V_1 for which we create a new node v_e, and there are only pk new nodes $v(p', w)$. To get rid of the term $\min\{m, k^2\}$ we can identify each node v_e, which derived from an edge $e = (v, w)$ inside V_1, with the node $v(p(w), w)$. This ensures that there are only pk new nodes. To show that $|V_0'| \leq (p + 1)^k + pk$ we can reduce the number of old nodes in V_1 to ensure that at most $(p + 1)^k$ remain. At first we remove all nodes $v \in V_0$ with $N_G^-(v) = \emptyset$, because they obviously cannot be part of a cycle and we can compute in linear time to which winning set they belong, once we know to which winning set their out-neighbors belong. Now let v and v' be two such nodes in V_0 with $N_G^+(v) = N_G^+(v')$. We then identify v and v' without changing the winning sets in V_1, since all nodes in $N_G^+(v)$ must have a priority at least as high as $\max\{p(v), p(v')\}$. This is because the priority of any node in $N_G^+(v)$ ($N_G^+(v')$) corresponds to the highest priority on a path that starts in v (v') and therefore must be at least $p(v)$ ($p(v')$). Afterward there remains at most one node $v \in V_0$ for each possible set $N_G^+(v)$.

Since $N_G^+(v)$ can contain at most one new node corresponding to w for each $w \in V_1$ and there are p different ones to choose from there are at most $(p+1)^k$ different possibilities for $N_G^+(v)$.

It remains to analyze the run time of the transformation. We consider the different steps of the reduction separately. The first step of removing all edges inside V_1 can be performed in $O(m)$ because we only need to check for every edge $e = (v, w) \in E$ if $v, w \in V_1$ and then remove one edge and add two edges and a node. The second step of removing dominions completely inside V_0 can be executed in time $O(\log(p) \cdot m)$ as follows. First, we solve the solitary game on $V_0 \setminus \mathsf{reach}_1(V_1)$ and remove the 0-reachability-set of its 0-winning set; this can be done in time $O(\log p \cdot m)$ [18]. Thereafter, we compute the 1-reachability set of $V_0 \setminus \mathsf{reach}_0(V_1)$ and remove it; this can be done in time $O(m)$ [64]. The third step of removing long paths inside V_0 can be performed as follows. The algorithm computes the best priority for player 0 that a path in V_0 from a node $v \in V_0$ to a node $w \in V_1$ can have. To determine which nodes in $v \in V_0$ can reach which nodes $w \in V_1$ via a path in V_0 whose highest priority has fixed value of p', consider the subgraph $G^{\leq p'}$ of G that is induced by the set $V^{\leq p'}$ of nodes of priority at most p' and remove from it those edges that start in V_1. We then consider the set of nodes with priority exactly p' and compute with a DFS in time $O(m)$ all nodes in V_1 reachable from them. Then we compute with a DFS for each node in V_0 which of the nodes with priority p' they can reach. This takes a total of at most $2n$ applications of DFS for each priority and therefore in total $O(pmn)$ time. In the last step, where we remove and contract some of the nodes in V_0, we can find all nodes without incoming edges in time $O(m)$ and we can order all remaining nodes by their outgoing edges in time $O(|V_1| \cdot (n+p+1))$ using a version of radix-sort, where we view the set of out-neighbors as an $(p+1)$-adic number with $|V_1|$ digits. Thereafter, in linear time we identify sets of nodes with the same outgoing neighbors and identify them in total time $O(n+m)$. $\square$

3.2.2 Bipartite Parity Games

In this section we give some reduction rules that efficiently reduce any bipartite game $G = (V_0 \uplus V_1, E, p)$ to a structurally simpler bipartite game $G' = (V_0' \uplus V_1', E', p')$, such that the winning sets of G can be efficiently recovered from the winning sets of G'. After exhaustive application of the reduction rules, the reduced game G' will have size bounded by some function of k and p only, independent of the size of G.

The digraphs of our underlying parity game may have self-loops and bidirected edges, but (without loss of generality) no parallel edges between the same two nodes. Thus, whenever parallel edges arise during the application of one of the reduction rules, we remove one of them without explicit mention.

Lemma 95. *Let $G = (V_0 \uplus V_1, E, p)$ be a bipartite parity game, and let $u, v \in V_0$ be such that $N_G^+(v) \subseteq N_G^+(u)$ and $p_1(v) \geq p_1(u)$. Let G' be the parity game obtained from G by deleting the edges $\{(w,u) \in E \mid (w,v) \in E\}$. Then the winning sets of G and G' are equal.*

Proof. We show that an edge $(w', u) \in \{(w,u) \in E \mid (w,v) \in E\}$ can only be part of a winning strategy for player 1 on node w' if the edge (w', v) is part of a winning strategy for player 1 on w' as well. Therefore, after deleting (w', u), player 1 wins from w' in G' if and only if he wins from w' in G. Deleting the edges in $\{(w,u) \in E \mid (w,v) \in E\}$ does therefore not change the winning sets.

Assume that player 1 has a winning strategy $s_1 : V_1 \to V_0$ for w' with $s_1(w') = u$. Let $s_1' : V_1 \to V_0$ be defined by $s_1'(w') = v$ and $s_1'(w) = s_1(w)$ for all $w \in V_1 \setminus \{w'\}$. We claim that s_1' is a winning strategy for player 1 on w' as well. Assume that there exists a counter strategy s_0' for s_1' such that player 0 wins the game with starting node w'. We will define a strategy s_0 for player 0 and show that s_0 is a counter strategy for s_1. Note that s_0 will not necessarily be a positional strategy. For all $w \in V_0 \setminus \{u\}$, s_0 chooses the same successor as s_0', but on u it might change its behavior. Each time the play encounters u directly after encountering w', strategy s_0 chooses $s_0'(v)$ as the successor of u. Every other time the play encounters u, strategy s_0 chooses $s_0'(u)$ as the successor of u.

The play defined by s_0' and s_1' can then be transformed into the play defined by s_0 and s_1 by replacing every appearance of the sequence $w', v, s_0'(v)$ with the sequence $w', u, s_0'(v)$. Let C' be the cycle created by s_0' and s_1'; then C' is a winning cycle for player 0. Then s_0 and s_1 will also create the cycle C' if C' does not contain the sequence $w', v, s_0'(v)$. Let us therefore assume that C' contains the sequence $w', v, s_0'(v)$. Let C be the closed walk obtained, when replacing v with u in C'. Notice that C does not have to be a cycle, as it could contain u twice. After a finite prefix the play defined by s_0 and s_1 will consist of an infinite recurrence of C. Since we have $p_1(v) \geq p_1(u)$ player 0 is wining the play defined by s_0 and s_1. This contradicts that s_1 is a winning strategy for player 1. □

Lemma 96. *Let $G = (V_0 \uplus V_1, E, p)$ be a bipartite parity game, and let $u, v \in V_0$ be nodes with $N_G^+(u) = N_G^+(v)$ and $p(v) = p(u)$. Let G' be the parity game obtained from G by contracting u and v into a new node v' with priority $p(v)$. Then u and v belong to the same winning set $\mathsf{win}_i(G)$ in G and v' belongs to the winning set $\mathsf{win}_i(G')$ of the same player in G'. For all other nodes the winning sets of G and G' coincide.*

Proof. Note that u and v belong to the winning set of the same player i in G. We can assume that player 0 chooses the same successor for u and v in her optimal strategy. Then no cycle created by optimal strategies contains both u and v and, after the contraction, each simple cycle that does not contain both u and v is again a simple cycle with the same priorities. Also, each cycle in the contracted game either exists in the original game (i.e., it does not contain v') or an equivalent cycle, which can be created by replacing v' with v or u, exists in the original game. We can also map winning strategies in the original game, where u and v have the same successor into winning strategies in the resulting game and vice versa by simply identifying the successors of v' with the successor of u and v and vice versa, while keeping the rest of the strategy. We again assume that in the winning strategies in the original game, v and u have the same successor w. We then set the successor of v' to w and set the successor of any node w' with successor v or u to v'. The other way around v and u get the same successor as v' and any node w' with successor v' gets either v or u as its successor, depending on which of the edges (w', v) and (w', u) exists in the original game. A pair of strategies and the pair of strategies, to which they are mapped to, then create corresponding cycles and must therefore either both be winning for player 1 or both be winning for player 0. □

Lemma 97. *Let $G = (V_0 \uplus V_1, E, p)$ be a bipartite parity game, and let $v \in V(G)$ be such that $N_G^-(v) = \emptyset$. Then for the parity game $G' = G - v$ and for $i \in \{0, 1\}$, any node $v' \neq v$ is winning for player i in G if and only if it is winning for player i in G'.*

Proof. The condition $N_G^-(v) = \emptyset$ implies that v cannot be part of any cycle. Let $v \in V_i$; then $v \in \mathsf{win}_i(G)$ is equivalent to the existence of some node $w \in \mathsf{win}_i(G) \cap N_G^+(v)$. Since all possible strategies for all nodes except v are also possible strategies in $G - \{v\}$, all nodes in $V \setminus \{v\}$ belong to the same winning set in G and in $G - v$. (Notice that $G - \{v\}$ is again a parity game.) Once we computed the winning sets for $G - \{v\}$, we can check in time $O(n)$ whether $v \in \mathsf{win}_i(G)$ or $v \in \mathsf{win}_{\neg i}(G)$. □

Lemma 98. *Let $G = (V_0 \uplus V_1, E, p)$ be a parity game with largest priority $p_{\max} = \max\{p(v) \mid v \in V(G)\}$. If $p^{-1}(z) = \emptyset$ for some $z \in \{1, \dots, p_{\max}\}$ then let $G' = (V_0 \uplus V_1, E, p')$ be the parity game obtained from G by setting $p'(v) = p(v) - 2$ for all $v \in V$ with $p(v) > z$ and $p'(v) = p(v)$ for all $v \in V$ with $p(v) < z$. Then the winning sets of the games G and G' coincide.*

Proof. Let s_0 and s_1 be strategies for player 0 and player 1, respectively, and let $C = (v_0, v_1, \dots, v_\ell)$ be the cycle created by these strategies when the game starts at some node v. The parity i of the largest element in the set $Q = \{p(v_0), \dots, p(v_\ell)\}$ determines which player wins in G and the parity i' of the largest element in the set $Q' = \{p'(v_0), \dots, p'(v_\ell)\}$ determines which player wins in G'. It is easy to see that our reduction ensures that $i = i'$. Since this is true for any cycle, the lemma follows. □

Corollary 99. *In any parity game with maximum priority $p_{\max}$ to which the reduction rule described in Lemma 98 cannot be applied anymore, the set of priorities is either $\{0, 1, \dots, p_{\max}\}$ or $\{1, \dots, p_{\max}\}$.*

Lemma 100. *Let $G = (V_0 \uplus V_1, E, p)$ be a bipartite parity game with $|V_1| = k$ that is reduced according to Lemmas 95–97. Then $|V_0| \leq 2^k \cdot \min\{k, p\}$.*

Proof. For each node $v \in V_0$ there are 2^k possible choices for $N_G^+(v)$. Lemma 95 yields that for two nodes $v \neq u \in V_0$ with $N_G^+(v) = N_G^+(u)$ we must have $N_G^-(v) \cap N_G^-(u) = \emptyset$. Lemma 97 then yields that there can be at most k nodes in V_0 for every possible choice of $N_G^+(v)$. Also Lemma 96 yields that for each possible choice of $(N_G^+(v), p(v))$ there exists at most one node in V_0. □

Lemma 101. *There exists a sequence of applications of the reduction rules described in Lemmas 95–98 with a total run time of $O(n^3)$ that leads to a game which cannot be reduced by any of the descried rules anymore.*

Proof. We show for each reduction rule separately how to apply it exhaustively in time $O(n^3)$. It can happen, that some reductions corresponding to one of the rules lead to allowing some other reductions, which were not allowed before. Therefore we cannot only apply all reductions corresponding to one rule after all reductions corresponding to another rule.

Most of the run time will be necessary to test if Lemma 95 or Lemma 96 applies to an ordered pair of nodes. We will argue how to apply the reductions such that we do not have to test the same ordered pair of nodes more than once, yielding a total run time of $O(n^3)$.

To apply all reductions of Lemma 98, we first sort the nodes in increasing order of their priorities and create an order of subsets each containing all nodes with the same priority; this can be done in $O(n \log(n))$ time. We then save for each of the subsets if its corresponding priority is odd or even and unite consecutive sets with the same parity. If the parity of the priority in the first subset is even, all nodes in the i-th subset get priority $i - 1$; otherwise all nodes in the i-th subset get priority i. Uniting sets can be done in linear time, and we cannot unite more than n times. The time for applying Lemma 98 is thus $O(n^2)$.

To apply all reductions for Lemma 95, we need to check for each pair of nodes $\{u, v\} \subseteq V_0$ with $p_1(v) \geq p_1(u)$ whether $N_G^+(v) \subseteq N_G^+(u)$, and find all nodes w with $(w, u) \in E$ and $(w, v) \in E$. There are $O(n^2)$ node pairs $\{u, v\} \in V_0$ with $p_1(v) \geq p_1(u)$, which can easily be found using the order of subsets created for Lemma 98. Checking if $N_G^+(v) \subseteq N_G^+(u)$ and finding all nodes w with $(w, u) \in E$ and $(w, v) \in E$ can be done in time $O(n)$. The total run time for Lemma 95 therefore is $O(n^3)$.

To apply all reductions for Lemma 96 we need to check for each pair of nodes $\{u, v\} \subseteq V_0$ with $p(v) = p(u)$ whether $N_G^+(v) = N_G^+(u)$. There are $O(n^2)$ such pairs $\{u, v\}$ with $p(v) = p(u)$, which can easily be found using the order of subsets created for Lemma 98. Testing whether $N_G^+(v) = N_G^+(u)$ and identifying u and v can be done in time $O(n)$. The total run time for Lemma 96 therefore is $O(n^3)$.

To apply all reductions for Lemma 97, we only need to check for each node if it has incoming edges and possibly delete it. Testing a node can be done in constant time, and deleting a node takes at most linear time. The time for applying Lemma 97 is thus $O(n^2)$.

We will first apply all feasible reductions for Lemma 98, then all feasible reductions for Lemmas 95, 96 and 97. Any reduction that is now possible was not feasible in the beginning.

Observe that some reductions can result in other reductions becoming feasible. Since we do not change the out-neighborhood of any node in V_0, reductions corresponding to Lemmas 95 and 96 for a pair of nodes $\{u, v\} \subseteq V_0$ can only become feasible when we combine the two subsets containing v and u in a reduction corresponding to Lemma 98. For each node pair $\{u, v\} \subseteq V_0$ this can happen at most once. The total run time for all reductions corresponding to Lemma 95 and 96 therefore is in $O(n^3)$. Reductions corresponding to Lemma 97 and a node $v \in V_0$ can only become feasible when we remove incoming edges of v. This can happen at most n times for each node $v \in V_0$, before we remove it. The total run time for all reductions corresponding to Lemma 97 therefore is in $O(n^2)$. Reductions corresponding to Lemma 98 can only become feasible when all nodes of one subset have been removed. This can happen at most n times; hence any node will be moved to another subset at most n times. The total run time for all reductions corresponding to Lemma 98 therefore is in $O(n^2)$. $\square$

We can now prove our main kernelization result.

Proof(Theorem 13). The part of the theorem for general instances follows directly from Lemma 94. The part for bipartite instances follows from Lemma 100 and Lemma 101 because the reduced bipartite parity game $G' = (V_0' \uplus V_1', E', p')$ satisfied $|V_0'| \leq 2^k \cdot \min\{k, p\}$ and $|V_1'| \leq k$. Since G' is bipartite, this implies that it contains at most $k2^k \cdot \min\{k, p\}$ edges. $\square$

3.3 A Simple Exponential-Time Algorithm

A simple algorithm with run time $O(2^n)$ for the solution of parity games originates from the work of McNaughton [81] and was first presented for parity games by Zielonka [103]; see also Grädel et al. [48]. Algorithm **win**(G) receives a parity game G and returns the pair of winning sets $(\mathsf{win}_0(G) = W_0, \mathsf{win}_1(G) = W_1)$.

Algorithm **win**(G) is based on Lemmas 92 and 93. Let $p_{\max}$ be the largest priority in G and let $V_{p_{\max}}$ be the set of nodes with priority $p_{\max}$. Let $i = p_{\max} \pmod 2$ be the player who owns the highest priority. The algorithm first finds the winning sets (W_0', W_1') of the smaller game $G' = G - \mathsf{reach}_i(V_{p_{\max}})$ in a first recursive call. If $W_{\neg i}' = \emptyset$, then by Lemma 93 player i wins from all nodes of G and we are done. Otherwise, again by Lemma 93 we know that $W_{\neg i}' \subseteq \mathsf{win}_{\neg i}(G)$. The algorithm then finds the winning sets (W_0'', W_1'') of the smaller game $G'' = G - \mathsf{reach}_{\neg i}(W_{\neg i}')$ by a second recursive call. By Lemma 92, $\mathsf{win}_i(G) = W_i''$ and $\mathsf{win}_{\neg i}(G) = \mathsf{reach}_{\neg i}(W_{\neg i}') \cup W_{\neg i}'' = V(G) \setminus W_i''$.

Algorithm 4 win(G)

Input: A parity game $G = (V_0 \uplus V_1, E, p)$ with maximum priority $p_{\max}$.
Output: (W_0, W_1), where W_i is the winning set of player $i \in \{0, 1\}$.

```
1: if $V = \emptyset$ then
2:     return $(\emptyset, \emptyset)$
3: $i \leftarrow p_{\max} \pmod 2; j \leftarrow \neg i$
4: $(W_0', W_1') \leftarrow \mathbf{win}(G - \mathsf{reach}_i(V_{p_{\max}}))$
5: if $W_j' = \emptyset$ then
6:     $(W_i, W_j) \leftarrow (V, \emptyset)$
7: else
8:     $(W_0'', W_1'') \leftarrow \mathbf{win}(G - \mathsf{reach}_j(W_j'))$
9:     $(W_i, W_j) \leftarrow (W_i'', V \setminus W_i'')$
10: return $(W_0, W_1)$
```

Theorem 102. *Algorithm* **win**(G) *finds the winning sets of any parity game on n nodes in time $O(2^n)$.*

Proof. The correctness of the algorithm follows from Lemmas 92 and 93, as argued above. Let $T'(n)$ be the number of steps needed by algorithm **win**(G) to solve a game G on n nodes. Algorithm **win**(G) makes two recursive calls **win**(G') and **win**(G'') on games with at most $n-1$ nodes. Other than that, it performs only $O(n^2)$ operations. (The most time consuming operations are the computations of the sets $\mathsf{reach}_i(V_{p_{\max}})$ and $\mathsf{reach}_j(W_j')$.) Therefore, $T'(n) \le 2T'(n-1) + O(n^2)$, which implies $T'(n) = O(2^n)$. □

3.4 Overview of the New Algorithms

Before we describe our new algorithms that lead to Theorems 11 and 14 in detail in Sect. 3.6 and Sect. 3.7, we present an overview of the main ideas. The algorithm **new-win**(G) by Jurdziński, Paterson, and Zwick [64] with run time $n^{O(\sqrt{n})}$ is a slight modification of the just described algorithm $\mathbf{win}(G)$. At the beginning of each recursive call it tests in time $O(n^\ell)$ if the parity game contains a dominion D of size at most $\ell = \lceil\sqrt{2n}\rceil$. If this is the case then D is removed and the remaining game is solved recursively. Else, the parity game is solved by the algorithm $\mathbf{win}(G)$, except that the recursive calls in lines 4 and 8 are made to $\mathbf{new\text{-}win}(G)$. Since this happens only when G does not contain a dominion of size at most ℓ, the dominion $\mathsf{reach}_j(W'_j)$ that is removed in line 8 has size greater than ℓ and hence, the second recursive call is to a substantially smaller game. Overall, this leads to the improved run time of $n^{O(\sqrt{n})}$.

Our new algorithms are based on a similar idea. Instead of simply searching for a dominion of size at most ℓ, our algorithm $\mathbf{new\text{-}win}_1(G)$ that leads to Theorem 11 searches for a dominion that contains at most $\ell = \lfloor\sqrt{2k}\rfloor$ nodes of the odd player, assuming without loss of generality that the odd player controls fewer nodes, i.e., $k = |V_1|$. If such a dominion is found then we remove it from the game and solve the remaining game recursively. Otherwise, we use the algorithm $\mathbf{win}(G)$ to solve the parity game, except that the recursive calls in lines 4 and 8 are made to $\mathbf{new\text{-}win}_1(G)$. It can happen that in the game to which the first recursive call in line 4 is made, the odd player controls again k nodes. We will show that in bipartite instances this cannot happen in two consecutive calls. For general instances we use that the observation that at least the number of different priorities decreases by one in the recursive call. Searching efficiently for a dominion that contains at most $\ell = \lfloor\sqrt{2k}\rfloor$ nodes of the odd player is more involved than simply searching for dominions whose total size is at most ℓ. We use multiple recursive calls of $\mathbf{new\text{-}win}_1$ to test if such a dominion exists, which makes the recursion of our algorithm and its analysis more complicated.

Our second algorithm leading to Theorem 14 is based on the same approach and inspired by the algorithm of Jurdziński, Paterson, and Zwick [64]. In this case we let s_j, for some $j \in \mathbb{N}$, equal the number of nodes with out-degree at most j. We separate the nodes into s_j nodes with out-degree at most j and $n - s_j$ nodes with out-degree larger than j and, at the beginning of each iteration, search for and remove dominions that contain at most $\ell = \lceil\sqrt{2(n-s_j)}\rceil$ nodes with out-degree larger than j and at most $s = \lceil\sqrt{s_j \cdot \log_j s_j}\rceil$ nodes with out-degree at most j. This algorithm runs in time $n^{O\left(\sqrt{n-s_j}+\sqrt{\frac{s_j}{\log_j s_j}}\right)}$, which implies Theorem 14. □

3.5 Finding Small Dominions

We now describe how dominions with the previously discussed properties can be found. Let $G = (V_0 \uplus V_1, E, p)$ be a parity game. Recall that for $i \in \{0,1\}$, a set $D \subseteq V$ is an i-dominion if player i can win from every node of D without ever leaving D, regardless of the strategy of player $\neg i$. Note that any i-dominion must be i-closed. A set $D \subseteq V$ is a *dominion* if it is either a 0-dominion or a 1-dominion. By prefix independence of parity games, the winning set $\mathsf{win}_i(G)$ of player i is an i-dominion.

For $k, p \in \mathbb{N}$, let $T(k)$ denote the maximum number of steps needed to solve a bipartite parity game $G = (V_0 \uplus V_1, E, p)$ and let $T(k,p)$ denote the maximum number of steps needed to solve a general parity game $G = (V_0 \uplus V_1, E, p)$ with $|V_1| = k$ and $p = |\{p(v) \mid v \in V\}|$ using some fixed algorithm. For $k, p, \ell \in \mathbb{N}$, let $\mathsf{dom}_k(\ell)$ denote the maximum number of steps required to find a dominion D with $|V_1 \cap D| \leq \ell$ in a bipartite parity game $G = (V_0 \uplus V_1, E, p)$ with $|V_1| = k$ and let $\mathsf{dom}_{k,p}(\ell)$ denote the maximum number of steps required to find a dominion D with $|V_1 \cap D| \leq \ell$ in a general parity game $G = (V_0 \uplus V_1, E, p)$ with $|V_1| = k$ and $p = |\{p(v) \mid v \in V\}|$, or to determine that no such dominion exists.

We will in the analysis of run times make the assumption that computation and removal of reachability sets as well as kernelization are elementary operation and can therefore be performed in time $O(1)$. To obtain the actual run times of our algorithms we will in the end multiply the computed run times by a factor corresponding to the time needed for these operations.

Lemma 103. *For $k \geq 4$, $\mathsf{dom}_k(\ell) = O(k^\ell \cdot T(\ell))$ and $\mathsf{dom}_{k,p}(\ell) = O(k^\ell \cdot T(\ell, p))$.*

Proof. There are $O(k^\ell)$ sets $V_D \subseteq V_1$ with $|V_D| = \ell$. We argue below that for each such set V_D, one can determine whether or not there exists a dominion D with $D \cap V_1 \subseteq V_D$ by solving two parity games that are subgames of G, i.e., these games arise from G by removing some of the nodes. This implies the lemma because each of these subgames can be solved in time $T(\ell)$ or $T(\ell, p)$ for bipartite or general parity games, respectively.

Let $V_D \subseteq V_1$ be a set with $|V_D| = \ell$. We will now show how to check if there exists an i-dominion D with $D \cap V_1 \subseteq V_D$. If such an i-dominion D exists, then it is i-closed. Therefore, it does not contain any node $v \in V$ from which player $\neg i$ can reach a node in $V_1 \setminus V_D$. Let $V' = V(G) \setminus \mathsf{reach}_{\neg i}(V_1 \setminus V_D)$ be the set of nodes from which player $\neg i$ cannot force to reach a node in $V_1 \setminus V_D$; the set V' can therefore be computed by computing and removing a reachability set, which as we assumed is an elementary operations. We then have $D \subseteq V'$, and since no node in $V_1 \setminus V_D$ can be part of V', it holds that $V' \cap V_1 \subseteq V_D$. Since V' is an i-closed set, the game $G - \mathsf{reach}_{\neg i}(V_1 \setminus V_D)$ is well defined. Let $\mathsf{win}_i(V')$ be the winning set of player 1 in the game $G - \mathsf{reach}_{\neg i}(V_1 \setminus V_D)$. Then $\mathsf{win}_i(V')$ is an i-dominion that contains D.

This shows that for each set $V_D \subseteq V_1$ with $|V_D| = \ell$ we only need to compute for each $i \in \{0, 1\}$ the sets $V'_i = V \setminus \mathsf{reach}_{\neg i}(V_1 \setminus V_D)$ of nodes from which player $\neg i$ cannot force to enter $V_1 \setminus V_D$ and compute the winning sets of the game $G - \mathsf{reach}_{\neg i}(V_1 \setminus V_D)$ to determine whether or not there exists a dominion D with $D \cap V_1 \subseteq V_D$. $\square$

With the algorithm described in Lemma 103 we can find a dominion D such that $|D \cap V_1| \leq \ell$ if such a dominion exists. We denote this algorithm by $\mathbf{dominion}_1(G, \ell)$ and assume that it returns either the pair (D, i) if an i-dominion D is found, or $(\emptyset, -1)$ if not.

We will give the pseudocode for algorithm $\mathbf{dominion}(G, \ell, s)$. In the pseudocode, let U_m denote the set of marked nodes from U and let $\mathbf{king}(U, \text{strategy}_i)$ denote an execution of the algorithm by King et al. [67] that determines the winners of the subgame G restricted to U with a given strategy for player i.

Algorithm 5 dominion(G, ℓ, s)

Input: A parity game $G = (V_0 \uplus V_1, E, p)$ and $\ell, s \in \{0, \dots, |V(G)|\}$.

Output: An i-dominion (D, i) for $i \in \{0, 1\}$ or $(\emptyset, -1)$ if no dominion is found.

```
 1: Fix a total order $\prec$ on the nodes of $G$.
 2: For each $u \in V(G)$ sort the edges emanating from $u$ by $\prec$ on their respective
    endpoint.
 3: for $i \in \{0,1\}$ do
 4:     for $v \in V_i$ do
 5:         for $\langle a_1, \dots, a_\ell, b_1, \dots, b_s \rangle \in \{1, \dots, |V(G)|\}^\ell \times \{1, \dots, j\}^s$ do
 6:             $r_1 = 1, r_2 = 1, U = \{v\}, U_m = \emptyset$
 7:             while $U \neq \emptyset$, $r_1 \leq \ell$ and $r_2 \leq s$ do
 8:                 Choose $u = \min(U, \prec)$.
 9:                 $U = U \setminus \{u\}$, $U_m = U_m \cup \{u\}$.
10:                 if $u \in V_i$ then
11:                     if $|\delta^+(u)| > j$ then
12:                         if $|\delta^+(u)| \leq r_1$ then
13:                             Let $e = (u, w)$ be the $a_{r_1}$-th outgoing edge of $u$.
14:                             $U = U \cup (\{w\} \setminus U_m)$, $r_1 = r_1 + 1$, $\text{strategy}_i(u) = w$.
15:                         else
16:                             $U = \emptyset$, $U_m = \emptyset$.
17:                     else
18:                         if $|\delta^+(u)| \leq r_2$ then
19:                             Let $e = (u, w)$ be the $b_{r_2}$-th outgoing edge of $u$.
20:                             $U = U \cup (\{w\} \setminus U_m)$, $r_2 = r_2 + 1$, $\text{strategy}_i(u) = w$.
21:                         else
22:                             $U = \emptyset$, $U_m = \emptyset$.
23:                 else
24:                     $U = U \cup (N^+(u) \setminus U_m)$
25:             if $U_m \neq \emptyset$ contains at most $\ell$ high out-degree nodes and at most $s$ low
    out-degree nodes then
26:                 $(W_0, W_1) = \textbf{king}(U, \text{strategy}_i)$.
27:                 if $W_i = U$ then
28:                     return $(U, i)$
29: return $(\emptyset, -1)$.
```

3.6 New Algorithms for Solving Parity Games

We present the algorithm **new-win**$_1(G)$ discussed in Sect. 3.4 in detail. Let $G = (V_0 \uplus V_1, E, p)$ with $|V_1| = k$ be a parity game with p distinct priorities.

The algorithm **new-win**$_1$ starts by trying to find a "small" dominion D, where small means $|D \cap V_1| \leq \ell$, where $\ell = \lfloor\sqrt{2k}\rfloor$ is a parameter chosen to minimize the run time of the algorithm. If such an i-dominion is found, then we remove it together with its i-reachability set from the game and solve the remaining game recursively. If no small dominion is found, then **new-win**$_1$ simply calls algorithm **old-win**$_1$, which is almost identical to algorithm **win**. The only difference between **old-win**$_1$ and **win** is that its recursive calls are made to **new-win**$_1$ and not to itself.

The recursion stops once the number of odd nodes is at most 4, in which case we will test each of the at most $((p+1)^4)^4$ (due to the size of our kernel) different strategies for player 1 in constant time. We will call this brute force method **solve**(G). We will also kernelize using the reduction rules described in Sect. 3.2. We will call the kernelization subroutine **kernel**(G). The pseudocode of **new-win**$_1(G)$ can be found in Sect. 3.8.

The correctness of the algorithm follows analogously to the correctness of **win**(G). We analyze the run time of **new-win**$_1(G)$ and prove Theorem 11 in Section 3.9.

3.7 Out-Degree based Algorithm

We now describe our second algorithm **new-win**$_2(G, j)$. In order to describe it, let $j \in \mathbb{N}$ and let s_j denote the number of nodes of out-degree at most j. **new-win**$_2(G, j)$ is then almost identical to **new-win**$_1(G)$, but instead of dominions that contains at most $\ell' = \lfloor\sqrt{2k}\rfloor$ nodes of the odd player, we search for and delete dominions that contain at most $\ell = \lceil\sqrt{2(n - s_j)}\rceil$ nodes with out-degree larger than j and at most $s = \lceil\sqrt{s_j \cdot \log_j s_j}\rceil$ nodes with out-degree at most j. This algorithm has a run time of $n^{O\left(\sqrt{n-s_j}+\sqrt{\frac{s_j}{\log_j s_j}}\right)}$, which implies Theorem 14.

In the following let us assume $j = \arg\min_{1\leq j'\leq n}\left\{\sqrt{n - s_{j'}} + \sqrt{\frac{s_{j'}}{\log_{j'} s_{j'}}}\right\}$. We say that a node v has *high out-degree* if $|N_G^+(v)| > j$ and *low out-degree* otherwise. For $n, z, \ell, s \in \mathbb{N}$, let $\mathsf{dom}_{n,z}(\ell, s)$ denote the maximum number of steps required to find a dominion D with at most ℓ high out-degree and at most s low out-degree nodes in a parity game $G = (V_0 \uplus V_1, E, p)$ with n nodes out of which z are high out-degree nodes, or to determine that no such dominion exists.

Lemma 104. *For all values of $\ell, s \in \{0, \dots, n\}$, it holds*

$$\mathsf{dom}_{n,z}(\ell, s) = O\left(n^{\ell+1} j^s (\ell + s)^2 \cdot \max\{1, \log(\ell + s)\}\right) = O\left(n^{\ell+4} j^s\right).$$

Proof. Fix an arbitrary total order $\prec$ on $V(G)$. Let $u \in V(G)$ be a node of G and let $(u, v_1), \dots, (u, v_{|\delta^+(u)|})$ be the edges emanating from u, where $v_i \prec v_{i+1}$ for all

$i \in \{1, \ldots |\delta^+(v)| - 1\}$; we call (u, v_i) the *i-th outgoing edge of u.* The algorithm generates at most $O(n \cdot n^\ell j^s)$ 0-closed sets of nodes that contain at most ℓ nodes with an out-degree greater than j and at most s nodes with an out-degree at most j, which are candidates for being 0-dominions. For every node $v \in V$ and every sequence $\langle a_1, \ldots, a_\ell, b_1, \ldots, b_s \rangle \in \{1, \ldots, n\}^\ell \times \{1, \ldots, j\}^s$ construct a set $U \subseteq V$ as follows. Start with $U = \{v\}$ and $r_1 = 1, r_2 = 1$. Nodes added to U are initially unmarked. As long as there is still an unmarked node in U, pick the smallest such node $u \in U$ with respect to $\prec$ and mark it.

- If $u \in V_0$ and u has high out-degree then add the endpoint of the a_{r_1}-th outgoing edge of u to U (if it is not already present in U) and increment r_1.
- If $u \in V_0$ and u has low out-degree then add the endpoint of the b_{r_2}-th outgoing edge of u to U (if it is not already present in U) and increment r_2.
- If $u \in V_1$ then add the endpoints of all outgoing edges of u that are not yet part of U to U.

If at some stage U contains either more than ℓ nodes with high out-degree or more than s nodes with low out-degree, or the endpoint of the i-th outgoing edge of some node v with out-degree $|\delta^+(v)| < i$ should be added to U, then discard the set U and restart the construction with the next sequence. If the process above ends without discarding U, then a 0-closed set containing at most ℓ high out-degree and at most s low out-degree nodes has been found. Furthermore, for every node $u \in U \cap V_0$, one of the outgoing edges of u was selected. This corresponds to a suggested strategy for player 0 in the game G restricted to the set U.

Our algorithm therefore considers by exhaustive search all 0-closed sets containing at most ℓ high out-degree and at most s low out-degree nodes, and for each set considers all possible positional strategies for player 0. Using an algorithm of King et al. [67] we can check in time $O((\ell + s)^2 \log(\ell + s))$ time whether a given pair of set U and proposed strategy is indeed a winning strategy for player 0 from all nodes of U. Thus, if there is a 0-dominion containing at most ℓ high out-degree and at most s low out-degree nodes, then the algorithm will find one. Finding 1-dominions can be done in an analogous manner. $\square$

With the just described algorithm, we can find a dominion D with at most ℓ nodes with high out-degree and at most s nodes with low out-degree if such a dominion exists. We denote this algorithm by **dominion**$_2(G, \ell, s)$, and suppose that it returns either the pair (D, i) if such an i-dominion D is found, or $(\emptyset, -1)$ if not.

The algorithm **new-win**$_2$ starts by trying to find a "small" dominion D, where small means that D contains at most ℓ nodes with out-degree greater than j and at most s nodes with out-degree at most j, where $\ell = \lceil \sqrt{2(n - s_j)} \rceil$ and $s = \lceil \sqrt{s_j \cdot \log_j s_j} \rceil$ are parameters chosen to minimize the run time of the whole algorithm. If such an i-dominion is found, then we remove it together with its i-reachability set from the game and solve the remaining game recursively. If no small dominion is found, then **new-win**$_2$ simply calls algorithm **old-win**$_2$, which is almost identical to algorithm **win**. The only difference between **old-win**$_2$ and **win** is that its recursive calls are made to **new-win**$_2$ and not to itself.

The recursion stops once the number of nodes with out-degree at most j and the number of nodes with out-degree greater than j are both at most 3, in which case we will test all of the at most constant different strategies for the two players in constant time. We will call this brute force method **solve(G)**. The pseudocode of **new-win**$_2$ can be found in Sect. 3.8.

The correctness of **new-win**$_2$ follows analogously to the correctness of the simple algorithm **win**. We analyze the run time of **new-win**$_2$ and prove Theorem 14 in Sect. 3.9.

We can now prove Corollary 15.

Proof(Corollary 15). First consider a parity game on n nodes played on a graph with maximum out-degree Δ. Then $s_\Delta \le n$ and

$$\sqrt{n - s_{\Delta_{\max}}} + \sqrt{\frac{s_{\Delta_{\max}}}{\log_{\Delta_{\max}} s_{\Delta_{\max}}}} \le \sqrt{\frac{n}{\log_{\Delta_{\max}} n}} = \sqrt{\frac{\log(\Delta_{\max})n}{\log n}} \enspace .$$

Now the first part of the corollary follows immediately from Theorem 14.

Let us now consider the case that the average out-degree is Δ_{avg} and let $z = \log(n)\Delta_{\text{avg}}$. Then Markov's inequality implies $s_z \ge (1 - 1/\log(n))n$. Hence,

$$\sqrt{n - s_z} + \sqrt{\frac{s_z}{\log_z s_z}} \le \sqrt{\frac{n}{\log(n)}} + \sqrt{\frac{n}{\log_z n}} = \sqrt{\frac{n}{\log(n)}} + \sqrt{\frac{\log(\log(n)\Delta_{\text{avg}})n}{\log n}} \enspace .$$

Now the second part of the corollary follows immediately from Theorem 14. □

3.8 Pseudocode for Algorithms new-win

We will now give the pseudocode for the algorithms **new-win**$_1(G)$ and **new-win**$_2(G, j)$ together with their subroutines **old-win**$_1(G)$ and **old-win**$_2(G, j)$. In the pseudocodes, we call a function **solve**(G). This function denotes a bruteforce method to solve parity games and is only used on very small games.

Algorithm 6 new-win$_1(G)$

```
Input: A parity game $G = (V_0 \uplus V_1, E, p)$.
Output: A partition $(W_0, W_1)$ of $V$, where $W_i$ is the winning set of player $i \in \{0,1\}$.
1: $k \leftarrow |V_1|; \ell \leftarrow \left\lfloor \sqrt{2k} \right\rfloor$ ; $G = \textbf{kernel}(G)$
2: if $k \le 4$ then return solve$(G)$
3: $(D, i) \leftarrow \textbf{dominion}_1(G, \ell)$
4: if $D = \emptyset$ then
5:     $(W_0, W_1) \leftarrow \textbf{old-win}_1(G)$
6: else
7:     $(W'_0, W'_1) \leftarrow \textbf{new-win}_1(G - \mathsf{reach}_i(D))$
8:     $(W_{\neg i}, W_i) \leftarrow (W'_{\neg i}, V \setminus W'_{\neg i})$
9: return $(W_0, W_1)$
```

Algorithm 7 old-win$_1(G)$

```
Input: A parity game $G = (V_0 \uplus V_1, E, p)$.
Output: A partition $(W_0, W_1)$ of $V$, where $W_i$ is the winning set of player $i \in \{0,1\}$.
1: $G = \textbf{kernel}(G)$
2: $i \leftarrow p_{\max} \pmod 2$
3: $(W'_0, W'_1) \leftarrow \textbf{new-win}_1(G - \mathsf{reach}_i(V_{p_{\max}}))$
4: if $W'_{\neg i} = \emptyset$ then
5:     $(W_i, W_{\neg i}) \leftarrow (V, \emptyset)$
6: else
7:     $(W''_0, W''_1) \leftarrow \textbf{new-win}_1(G - \mathsf{reach}_{\neg i}(W'_{\neg i}))$
8:     $(W_i, W_{\neg i}) \leftarrow (W''_i, V \setminus W''_i)$
9: return $(W_0, W_1)$
```

Algorithm 8 new-win$_2(G, j)$

```
Input: A parity game $G = (V_0 \uplus V_1, E, p)$ and $j \in \{1, \ldots |V|\}$.
Output: A partition $(W_0, W_1)$ of $V$, where $W_i$ is the winning set of player $i \in \{0,1\}$.
1: $s_j \leftarrow |\{v \in V || \delta^+(v)| \leq j\}|; \ell \leftarrow \left\lceil \sqrt{2(n - s_j)} \right\rceil; s \leftarrow \left\lceil \sqrt{s_j \cdot \log_j s_j} \right\rceil$
2: if $s_j \leq 3$ and $n - s_j \leq 3$ then return $\textbf{solve}(G)$
3: $(D, i) \leftarrow \textbf{dominion}_2(G, \ell, s)$
4: if $D = \emptyset$ then
5:     $(W_0, W_1) \leftarrow \textbf{old-win}_2(G, j)$
6: else
7:     $(W'_0, W'_1) \leftarrow \textbf{new-win}_2(G - \mathsf{reach}_i(D), j)$
8:     $(W_{\neg i}, W_i) \leftarrow (W'_{\neg i}, V \setminus W'_{\neg i})$
9: return $(W_0, W_1)$
```

Algorithm 9 old-win$_2(G, j)$

```
Input: A parity game $G = (V_0 \uplus V_1, E, p)$.
Output: A partition $(W_0, W_1)$ of $V$, where $W_i$ is the winning set of player $i \in \{0,1\}$.
1: $i \leftarrow p_{\max} \pmod 2$
2: $(W'_0, W'_1) \leftarrow \textbf{new-win}_2(G - \mathsf{reach}_i(V_{p_{\max}}), j)$
3: if $W'_{\neg i} = \emptyset$ then
4:     $(W_i, W_{\neg i}) \leftarrow (V, \emptyset)$
5: else
6:     $(W''_0, W''_1) \leftarrow \textbf{new-win}_2(G - \mathsf{reach}_{\neg i}(W'_{\neg i}), j)$
7:     $(W_i, W_{\neg i}) \leftarrow (W''_i, V \setminus W''_i)$
8: return $(W_0, W_1)$
```

3.9 Analysis of the Run Time

We will show that algorithm **new-win**(G) has a run time of $O(p \cdot m \cdot n) \cdot (p+k)^{O(\sqrt{k})}$ on general instances and in time $O(n^3) \cdot k^{O(\sqrt{k})}$ on bipartite instances. We will also show that algorithm **new-win**(G, j) has a run time of $n^{O\left(\sqrt{n-s_j}+\sqrt{\frac{s_j}{\log_j s_j}}\right)}$, where s_j the number of nodes in G with out-degree at most j.

Note that the part $O(pmn)$ of the run time comes from the reduction of the instance and the computation and removal of reachability sets of found dominions. Since we do both of these often, we assume that they are be elementary computations with computation time $O(1)$ and show that the total remaining run time remaining is $(p+k)^{O(\sqrt{k})}$. In bipartite instances we need $O(n^3)$ time to reduce the instance and to compute and remove reachability sets. We will show that the remaining run time on bipartite instances is $k^{O(\sqrt{k})}$, when computation and removal of reachability sets and the reductions are viewed as an elementary operation. Let $T(k,p)$ denote the time required by algorithm **new-win** on a game $G = (V_0 \uplus V_1, E, p)$ with $|V_1| = k$ and $p = |\{p(v) \mid v \in V\}|$, when reduction of an instance and computation and removal of reachability sets of found dominions are viewed as elementary computations and have run time $O(1)$.

Lemma 105. *The following recurrence relation holds:*

$$(a) \quad T(k,p) \le \max\{T(k-1,p), T(k,p-1) + T(k-\ell,p)\} + \mathsf{dom}_{k,p}(\ell) + O(1) \ .$$

Proof. Algorithm **new-win**(G) tries to find dominions D with $|D \cap V_1| \le \ell = \lfloor \sqrt{2k} \rfloor$. By definition this takes at most $\mathsf{dom}_{k,p}(\ell)$ time on general instances. If a (non-empty) dominion is found, then the algorithm simply proceeds on the remaining game, which has at most $k-1$ odd nodes, and thus it solves this game in time bounded by $T(k-1,p)$. Otherwise, a call to **old-win**(G) is made. This results in a call to **new-win**$(G - \mathsf{reach}_i(V_{p_{\max}}))$. In this case the call takes at most $T(k, p-1)$ time because we removed all nodes with the highest priority. If the set W'_j returned by the call is empty, then we are done. Otherwise, $W'_j = \mathsf{win}_j(G - \mathsf{reach}_i(V_{p_{\max}}))$ and $W'_j \subseteq \mathsf{win}_j(G)$ by Lemma 91. Therefore, W'_j is a j-dominion of G. We are in the case that there is no dominion D with $|D \cap V_1| \le \ell$ in G. Thus, $|W'_j \cap V_1| > \ell$, and hence the second recursive call **new-win**$(G - \mathsf{reach}_j(W'_j))$ takes time at most $T(k - \lceil \ell \rceil, p)$. Consequently,

$$T(k,p) \le \max\{T(k-1,p), T(k,p-1) + T(k - \lceil \ell \rceil, p)\} + \mathsf{dom}_{k,p}(\ell) + O(1). \qquad \square$$

Let $T(k, \mathsf{even})$ and $T(k, \mathsf{odd})$ denote the time required by algorithm **new-win** on a bipartite game $G = (V_0 \uplus V_1, E, p)$ with $|V_1| = k$ when the largest priority is even respectively odd, when computation and removal of reachability sets of found dominions and the reductions are viewed as elementary computations. We denote by $T(k)$ the time required by algorithm **new-win** on any bipartite game with $|V_1| = k$; thus $T(k) \le \max\{T(k, \mathsf{even}), T(k, \mathsf{odd})\}$.

Lemma 106. *The following recurrence relations hold:*

$$\begin{aligned}
(b_1) \quad T(k,\mathsf{odd}) &\le T(k-1) + T(k-\ell) + \mathsf{dom}_k(\ell) + O(1),\\
(b_2) \quad T(k,\mathsf{even}) &\le \max\{T(k,\mathsf{odd}), T(k-1)\} + T(k-\ell) + \mathsf{dom}_k(\ell) + O(1)\\
&\le T(k-1) + 2T(k-\ell) + 2\mathsf{dom}_k(\ell) + O(1),\\
(b_3) \quad T(k) &\le T(k-1) + 2T(k-\ell) + 2\mathsf{dom}_k(\ell) + O(1) \ .
\end{aligned}$$

Proof. From the definition it follows directly that $T(k) \le \max\{T(k,\mathsf{even}), T(k,\mathsf{odd})\}$. Showing (b_1) and (b_2) therefore yields (b_3). Algorithm **new-win**(G) tries to find dominions D with $|D \cap V_1| \le \ell = \lceil\sqrt{2k}\rceil$. By definition this takes at most $\mathsf{dom}_k(\ell)$ time on bipartite instances. If a (non-empty) dominion is found, then the algorithm simply proceeds on the remaining game, which has at most $k-1$ odd nodes, and the remaining run time is therefore at most $T(k-1)$. Otherwise, a call to **old-win**(G) is made. This results in a call to **new-win**$(G - \mathsf{reach}_i(V_{p_{\max}}))$. Here we have to distinguish whether the highest priority is odd or even.

If the highest priority is odd then, by Lemma 97, the set $\mathsf{reach}_1(V_{p_{\max}}) \cap V_1$ is non-empty and the call takes at most $T(k-1)$ time.

In case the highest priority is even, we either have $\mathsf{reach}_0(V_{p_{\max}}) \cap V_1 \ne \emptyset$ or $\mathsf{reach}_0(V_{p_{\max}}) = V_{p_{\max}}$ in which case Lemma 98 yields that in $G - \mathsf{reach}_i(V_{p_{\max}})$ the highest priority has to be odd. Therefore, this call needs time at most $\max\{T(k,\mathsf{odd}), T(k-1)\}$.

If the set W_j' returned by the call is empty, then we are done. Otherwise, $W_j' = \mathsf{win}_j(G - \mathsf{reach}_i(V_{p_{\max}}))$ and this is part of $\mathsf{win}_j(G)$ by Lemma 91. Therefore, W_j' is a j-dominion of G. We are in the case that there is no dominion D with $|D \cap V_1|$ at most ℓ in G, so we know that $|W_j' \cap V_1| > \ell$, and therefore the second recursive call **new-win**$(G - \mathsf{reach}_j(W_j'))$ takes at most $T(k-\ell)$ time. Thus, we obtain

$$T(k,\mathsf{odd}) \le T(k-1) + T(k-\ell) + \mathsf{dom}_k(\ell) + O(1)$$

and

$$\begin{aligned}
T(k,\mathsf{even}) &\le \max\{T(k,\mathsf{odd}), T(k-1)\} + T(k-\ell) + \mathsf{dom}_k(\ell) + O(1)\\
&\le T(k-1) + 2T(k-\ell) + 2\mathsf{dom}_k(\ell) + O(1),
\end{aligned}$$

which yields $T(k) \le T(k-1) + 2T(k-\ell) + 2\mathsf{dom}_k(\ell) + O(1)$. □

For $j \in \mathbb{N}_0$, let $T'(s_j, n-s_j)$ denote the time required by algorithm **new-win** on a game G on n nodes of which s_j nodes have out-degree at most j.

Lemma 107. *The following recurrence relation holds:*

$$\begin{aligned}
(c) \quad T'(s_j, n-s_j) \le &\max\{T'(s_j-1, n-s_j), T'(s_j, n-s_j-1)\}\\
&+ \max\{T'(s_j-s, n-s_j), T'(s_j, n-s_j-\ell)\}\\
&+ \mathsf{dom}_{n,n-s_j}(\ell, s) + O(1) \ .
\end{aligned}$$

Proof. Algorithm **new-win**(G, j) tries to find dominions D containing at most s nodes with out-degree at most j and at most ℓ nodes with out-degree greater than j. By definition this takes at most $\mathsf{dom}_{n,n-s_j}(\ell, s)$ time. If a (non-empty) dominion is found, then the algorithm simply proceeds on the remaining game, which has at most $n-1$ nodes, and the remaining time is therefore at most $\max\{T'(s_j-1, n-s_j), T'(s_j, n-s_j-1)\}$. Otherwise, a call to **old-win**(G, j) is made. This results in a call to **new-win**$(G - \mathsf{reach}_i(V_{p_{\max}}), j)$, this call is to a game with fewer nodes and can be solved in time bounded by

$$\max\{T'(s_j-1, n-s_j), T'(s_j, n-s_j-1)\}\ .$$

If the set W'_k returned by the call is empty, then we are done. Otherwise, $W'_k = \mathsf{win}_k(G - \mathsf{reach}_i(V_{p_{\max}}))$, and $W'_k \subseteq \mathsf{win}_k(G)$ by Lemma 91. Therefore, W'_k is a k-dominion of G. We are in the case that there is no dominion D containing at most s nodes with out-degree at most j and at most ℓ nodes with out-degree greater than j, so W'_k either contains more than s nodes with out-degree at most j or more than ℓ nodes with out-degree greater than j, and therefore the second recursive call **new-win**$(G - \mathsf{reach}_k(W'_k))$ takes time bounded by $\max\{T'(s_j-s, n-s_j), T'(s_j, n-s_j-\ell)\}$.

All other computations can be done in constant time. Thus, we obtain

$$\begin{aligned} T'(s_j, n-s_j) &\le \max\{T'(s_j-1, n-s_j), T'(s_j, n-s_j-1)\} \\ &\quad + \max\{T'(s_j-s, n-s_j), T'(s_j, n-s_j-\ell)\} \\ &\quad + \mathsf{dom}_{n,n-s_j}(\ell, s) + O(1)\ . \end{aligned}$$ □

We analyze recurrences (a) and (b) with $\ell = \lfloor\sqrt{2k}\rfloor$ in Theorem 108 in Sect. 3.9.1, which eventually shows that $T(k,p) \le (p+k)^{O(\sqrt{k})}$ and $T(k) \le k^{O(\sqrt{k})}$, and recurrence (c) with $\ell = \left\lceil\sqrt{2(n-s_j)}\right\rceil$ and $s = \left\lceil\sqrt{\frac{s_j}{\log_j s_j}}\right\rceil$ in Theorem 112 in Sect. 3.9.1, which eventually shows that $T'(s_j, n-s_j) \le n^{O\left(\sqrt{n-s_j}+\sqrt{\frac{s_j}{\log_j s_j}}\right)}$. This completes the analysis of the run time of **new-win**(G) and **new-win**(G, j), and proves Theorem 11 and Theorem 14. □

3.9.1 Recurrence Relation Computations

In this section we analyze the recurrence relations used to bound the run time of **new-win**.

Theorem 108. *For $k \in \mathbb{N}$ and $\ell = \lfloor\sqrt{2k}\rfloor$, we obtain*

$$\begin{aligned} T(k,p) &= (p+k)^{O(\sqrt{k})}, \\ T(k) &= k^{O(\sqrt{k})}. \end{aligned}$$

To prove Theorem 108, we first establish some lemmas.

Lemma 109. *For $k, p \in \mathbb{N}$ and $\ell = \lfloor\sqrt{2k}\rfloor$, it holds*

$$\begin{aligned} T(k,p) &\leq 2(k+p)^{\lfloor\sqrt{2k}\rfloor} \cdot \mathsf{dom}_{k,p}(\left\lfloor\sqrt{2k}\right\rfloor) \quad \textit{and} \\ T(k) &\leq 2(2k)^{\lfloor\sqrt{2k}\rfloor} \cdot \mathsf{dom}_{k}(\left\lceil\sqrt{2k}\right\rceil) \ . \end{aligned}$$

Proof. For every pair of integers k and p we construct binary trees $T_{k,p}$ and T_k in the following way. The root of $T_{k,p}$ is labeled by k and $k+p$ and the root of T_k is labeled by k. A node labeled by a number $k > 4$ has two children: in $T_{k,p}$ a left child labeled by k and $k+p-1$ and a right child labeled by $k - \left\lceil\sqrt{2k}\right\rceil$ and $p + k - \left\lceil\sqrt{2k}\right\rceil$. In T_k a left child labeled by k' and a right child labeled by $k - \left\lceil\sqrt{2k}\right\rceil$. A node labeled by k' in T_k has two children: a left child labeled by $k-1$ and a right child labeled by $k - \left\lceil\sqrt{2k}\right\rceil$. Nodes labeled by a number $k \leq 4$ are leaves. A node labeled by k and $k+p$ has a cost of $\mathsf{dom}_{k,p}(\left\lfloor\sqrt{2k}\right\rfloor)$ associated with it and a node labeled by k or k' has a cost of $\mathsf{dom}_k(\left\lceil\sqrt{2k}\right\rceil))$ associated with it. It follows from Lemma 105 and Lemma 106 that the sum of the costs of the nodes of $T_{k,p}$ and T_k, is an upper bound on $T(k,p)$ and $T(k)$, respectively. Clearly, the length of every path from the root to a leaf is at most $p+k+1$ in $T_{k,p}$ and $2k$ in T_k. We say that such a path makes a right turn when it descends from a node to its right child. We next claim that each such path makes at most $\left\lfloor\sqrt{2k}\right\rfloor$ right turns. This follows immediately from the observation that the function $f(n) = n - \left\lceil\sqrt{2n}\right\rceil$ can be iterated on $2k$ at most $\left\lfloor\sqrt{2k}\right\rfloor$ times before reaching the value of 4 or less. This observation can be proved by induction, based on the fact that if $\frac{1}{2}j^2 < n \leq \frac{1}{2}(j+1)^2$, then $n - \left\lceil\sqrt{2n}\right\rceil \leq \frac{1}{2}j^2$. (Initially we have $j = \left\lfloor\sqrt{2k}\right\rfloor$ and finally, with $1 \leq n \leq 4$, we have $j \geq 1$.) As each leaf of $T_{k,p}$ and T_k is determined by the positions of the right turns on the path leading to it from the root, we get that the number of leaves is at most $\binom{k+p}{\lfloor\sqrt{2k}\rfloor}$ in $T_{k,p}$ and at most $\binom{2k}{\lfloor\sqrt{2k}\rfloor}$ in T_k. The total number of nodes is therefore at most at most $2\binom{k+p}{\lfloor\sqrt{2k}\rfloor} \leq 2(k+p)^{\lfloor\sqrt{2k}\rfloor}$ in $T_{k,p}$ and at most $2\binom{2k}{\lfloor\sqrt{2k}\rfloor} \leq 2(2k)^{\lfloor\sqrt{2k}\rfloor}$ in T_k. As the cost of each node is at most $\mathsf{dom}_{k,p}(\left\lfloor\sqrt{2k}\right\rfloor)$ in $T_{k,p}$ and at most $\mathsf{dom}_k(\left\lceil\sqrt{2k}\right\rceil)$ in T_k, we immediately get

$$\begin{aligned} T(k,p) &\leq 2(k+p)^{\lfloor\sqrt{2k}\rfloor} \cdot \mathsf{dom}_{k,p}(\left\lfloor\sqrt{2k}\right\rfloor) \quad \text{and,} \\ T(k) &\leq 2(2k)^{\lfloor\sqrt{2k}\rfloor} \cdot \mathsf{dom}_{k}(\left\lceil\sqrt{2k}\right\rceil) \ . \end{aligned}$$

□

We obtain together with Lemma 103 that

$$T(k,p) \leq 2(k+p)^{\lfloor\sqrt{2k}\rfloor} \cdot O\Big(k^{\lfloor\sqrt{2k}\rfloor} \cdot T(\left\lfloor\sqrt{2k}\right\rfloor, p)\Big),$$

as well as

$$T(k) \leq 2(2k)^{\left\lfloor\sqrt{2k}\right\rfloor} \cdot O\left(k^{\lceil\sqrt{2k}\rceil} T(\left\lceil\sqrt{2k}\right\rceil)\right).$$

Lemma 110. *Suppose that*

$$T(k,p) \leq 2(k+p)^{\lfloor\sqrt{2k}\rfloor} \cdot O\left(k^{\lfloor\sqrt{2k}\rfloor} \cdot T(\left\lfloor\sqrt{2k}\right\rfloor, p)\right)$$

and that $T(\ell,q) \leq c' \cdot (q+\ell)^{8\lfloor\sqrt{2\ell}\rfloor}$ *for some constant* $c' \in \mathbb{R}$ *and for all pairs* $(\ell,q) \in \{1,\ldots,4\} \times \mathbb{N}$. *Then there exist constants* $c_1 \geq c', c_2 \geq 8$ *such that for all* $k \in \mathbb{N}$,

$$T(k,p) \leq c_1 \cdot (p+k)^{c_2\lfloor\sqrt{2k}\rfloor} \ .$$

Proof. Since we have $T(k,p) \leq 2(k+p)^{\lfloor\sqrt{2k}\rfloor} \cdot O\left(k^{\lfloor\sqrt{2k}\rfloor} \cdot T(\left\lfloor\sqrt{2k}\right\rfloor, p)\right)$, there exists a constant $c_1' > 0$ such that $T(k,p) \leq 2(k+p)^{\lfloor\sqrt{2k}\rfloor} \cdot c_1' k^{\lfloor\sqrt{2k}\rfloor} \cdot T(\left\lfloor\sqrt{2k}\right\rfloor, p)$. Let $\alpha_k = \frac{\lfloor\sqrt{2k}\rfloor}{\left\lfloor\sqrt{2\lfloor\sqrt{2k}\rfloor}\right\rfloor}$. Then for $k \geq 5$, it holds $\alpha_k \geq 1.5 > 1$. Let $c_1 = \max\{c_1', c'\}$, and let $c_2 = 6 + 3\log(2c_1)$. Suppose, for sake of contradiction, that the statement of the lemma does not hold for this choice of (c_1, c_2). Then there exists a pair $(k',p') \in \mathbb{N} \times \mathbb{N}$ for which $T(k',p') > c_1 \cdot (p'+k')^{c_2\lfloor\sqrt{2k'}\rfloor}$. Let k' be the smallest integer for which such a pair exists, and let $p' = p'(k')$ be the smallest integer for which $T(k',p') > c_1 \cdot (p' +' k)^{c_2\lfloor\sqrt{2k'}\rfloor}$. Note that $k' \geq 4$ and $T(\ell,q) \leq c_1 \cdot q^3 \cdot (q+\ell)^{c_2\sqrt{\ell}}$ for all pairs (ℓ,q) with $\ell \leq k'$, $q \leq p'$ and $\ell + q < k' + p'$. Further, it holds $c_2 \geq \frac{c_2}{\alpha_k} + 2 + \log(2c_1')$ for all $k \geq 4$. For $k \geq 4$ we also have $\left\lfloor\sqrt{2k}\right\rfloor < k$. This implies that

$$\begin{aligned}
T(k',p') &\leq 2(k'+p')^{\lfloor\sqrt{2k'}\rfloor} \cdot c_1' \cdot k'^{\lfloor\sqrt{2k'}\rfloor} \cdot T(\left\lfloor\sqrt{2k'}\right\rfloor, p')) \\
&\leq 2(k'+p')^{\lfloor\sqrt{2k'}\rfloor} \cdot c_1' \cdot k'^{\lfloor\sqrt{2k'}\rfloor} \cdot c_1 \cdot (p' + \left\lfloor\sqrt{2k'}\right\rfloor)^{c_2\left\lfloor\sqrt{2\lfloor\sqrt{2k'}\rfloor}\right\rfloor} \\
&\leq 2(k'+p')^{\lfloor\sqrt{2k'}\rfloor} \cdot c_1' \cdot k'^{\lfloor\sqrt{2k'}\rfloor} \cdot c_1 \cdot (p' + \left\lfloor\sqrt{2k'}\right\rfloor)^{c_2\left\lfloor\sqrt{2\lfloor\sqrt{2k'}\rfloor}\right\rfloor} \\
&\leq (2c_1'c_1)(k'+p')^{2\lfloor\sqrt{2k'}\rfloor + c_2\left\lfloor\sqrt{2\lfloor\sqrt{2k'}\rfloor}\right\rfloor} \\
&\leq c_1(k'+p')^{2\lfloor\sqrt{2k'}\rfloor + \frac{c_2}{\alpha_{k'}}\lfloor\sqrt{2k'}\rfloor + \log(2c_1')} \\
&\leq c_1(k'+p')^{(2+\frac{c_2}{\alpha_{k'}}+\log(2c_1'))\lfloor\sqrt{2k'}\rfloor} \\
&\leq c_1(k'+p')^{c_2\lfloor\sqrt{2k'}\rfloor}
\end{aligned}$$

This contradicts the existence of k', and therefore concludes the proof. □

Lemma 111. *Suppose that*

$$T(k) \leq 2(2k)^{\lfloor\sqrt{2k}\rfloor} \cdot O\left(k^{\lfloor\sqrt{2k}\rfloor} T(\left\lfloor\sqrt{2k}\right\rfloor)\right)$$

and that $T(\ell) \leq c'\ell^{\lfloor\sqrt{2\ell}\rfloor}$ *for some constant* $c' \in \mathbb{R}$ *and for all* $\ell \leq 4$. *Then there exist constants* $c_1 \geq c', c_2 \geq 1$ *such that for all* $k \in \mathbb{N}$,

$$T(k) \leq c_1 k^{c_2\lfloor\sqrt{2k}\rfloor} \ .$$

Proof. Since $T(k) \leq 2(2k)^{\lfloor\sqrt{2k}\rfloor} \cdot O\left(k^{\lfloor\sqrt{2k}\rfloor} T(\left\lfloor\sqrt{2k}\right\rfloor)\right)$, there exists a constant $c_1' > 0$ such that $T(k) \leq 2(2k)^{\lfloor\sqrt{2k}\rfloor} \cdot c_1' \left(k^{\lfloor\sqrt{2k}\rfloor} T(\left\lfloor\sqrt{2k}\right\rfloor)\right)$. Let $\alpha_k = \frac{\lfloor\sqrt{2k}\rfloor}{\left\lfloor\sqrt{2\lfloor\sqrt{2k}\rfloor}\right\rfloor}$. Then for $k \geq 5$ it holds that $\alpha_k \geq 1.5 > 1$. Let $c_1 = \max\{c_1', c'\}$ and let $c_2 = 12 + 3\log c_1$. Suppose, for sake of contradiction, that the statement of the lemma does not holds for this choice of (c_1, c_2). Then exists a $k' \in \mathbb{N}$ such that $T(k') > c_1 k'^{c_2\lfloor\sqrt{2k'}\rfloor}$. Let k' be the smallest such integer. Note that we must have $k' \geq 4$ and $T(\ell) \leq c_1 \ell^{c_2\lfloor\sqrt{2\ell}\rfloor}$ for all $\ell < k'$. Further, it holds that $c_2 \geq \frac{c_2}{\alpha_k'} + 4 + \log c_1$ for all $k \geq 5$. For $k \geq 4$ we also have $\left\lfloor\sqrt{2k}\right\rfloor < k$. This implies that

$$\begin{aligned}
T(k') &\leq 2(2k')^{\lfloor\sqrt{2k'}\rfloor} \cdot c_1' \left(k'^{\lfloor\sqrt{2k'}\rfloor} T(\left\lfloor\sqrt{2k'}\right\rfloor)\right) \\
&\leq 2(2k')^{\lfloor\sqrt{2k'}\rfloor} \cdot c_1 \left(2c_1 k'^{\lfloor\sqrt{2k'}\rfloor} k'^{c_2\left\lfloor\sqrt{2\lfloor\sqrt{2k'}\rfloor}\right\rfloor}\right) \\
&\leq 2c_1(2k')^{\lfloor\sqrt{2k'}\rfloor} k'^{(\frac{c_2}{\alpha_k'}+1)\lfloor\sqrt{2k'}\rfloor + \log c_1} \\
&\leq 2c_1 k'^{2\lfloor\sqrt{2k'}\rfloor} k'^{(\frac{c_2}{\alpha_k'}+1)\lfloor\sqrt{2k'}\rfloor + \log c_1} \\
&\leq c_1 k'^{\lfloor\sqrt{2k'}\rfloor(\frac{c_2}{\alpha_k'}+3)+\log c_1+\log 2} \\
&\leq c_1 k'^{\lfloor\sqrt{2k'}\rfloor(\frac{c_2}{\alpha_k'}+4+\log c_1)} \\
&\leq c_1 k'^{c_2\lfloor\sqrt{2k'}\rfloor} .
\end{aligned}$$

This contradicts the existence of k', and therefore concludes the proof. □

Since $T(k,p) \in O(p^{(k^2)})$, it holds that $O(p^{8\lfloor\sqrt{2k}\rfloor})$. Moreover, as $T(k) = O(1)$ for $k \leq 4$, we conclude that $T(k,p) = (p+k)^{O(\sqrt{k})}$ and $T(k) = k^{O(\sqrt{k})}$. This completes the proof of Theorem 108. □

Next, we will prove the following.

Lemma 112. *For $s_j, j, n \in \mathbb{N}$, $s = \left\lceil\sqrt{\frac{s_j}{\log_j s_j}}\right\rceil$ and $\ell = \left\lceil\sqrt{2(n-s_j)}\right\rceil$ we obtain*

$$T(s_j, n-s_j) = n^{O\left(\sqrt{n-s_j}+\sqrt{\frac{s_j}{\log_j s_j}}\right)} .$$

To prove Lemma 112, we first establish another lemma.

Lemma 113. *For $s_j, j, n \in \mathbb{N}$, $s = \left\lceil\sqrt{\frac{s_j}{\log_j s_j}}\right\rceil$ and $\ell = \left\lceil\sqrt{2(n-s_j)}\right\rceil$, it holds*

$$T(s_j, n-s_j) \leq n^{O\left(\sqrt{n-s_j}+\sqrt{\frac{s_j}{\log_j s_j}}\right)} \cdot \left(\mathsf{dom}_{n,n-s_j}(\ell, s) + O(1)\right) .$$

Proof. For each parity game G on n nodes and $s_j = s_j(G)$ we construct a binary tree T_G in the following way. The root of T_G is labeled by $(s_j, n-s_j)$. A node labeled by

(a, b) with $a > 3$ and $b > 3$ has up to two children: a left child labeled by $(a-1, b)$ or $(a, b-1)$, and possibly a right child labeled by $(a - \sqrt{a \cdot \log_j a}, b)$ or $(a, b - \sqrt{b})$. Each child of a node corresponds to one of the recursive calls. The choice on how we label the children depends on the behavior of the algorithm. We label the children of a node by (a', b') and (a'', b'') such that the recursive calls of the algorithm are to games containing at most a', respectively a'' nodes of out-degree at most j and at most b', respectively b'', nodes of out-degree greater than j. Nodes labeled by (a, b) with $a, b \in \{0, 1, 2, 3\}$ are leaves. A node labeled by (a, b) has a cost of $\left(\mathsf{dom}_{a+b,b}(\sqrt{b}, \sqrt{a \cdot \log_j a}) + O(1)\right)$ associated with it.

It follows from Lemma 107 that the sum of the costs of the nodes of T_G is an upper bound on the run time of **new-win**(G, j). The worst possible sum of the costs of the nodes of T_G we can obtain for some instance G with $s_j = s_j(G)$ and $n = |V|$ therefore is an upper bound of $T(s_j, n - s_j)$. Clearly, the length of every path in T_G from the root to a leaf is at most n. We say that such a path makes a *right turn* when it descends from a node to its right child. We next claim that each such path makes at most $O(\sqrt{n - s_j} + \sqrt{\frac{s_j}{\log_j s_j}})$ right turns. This follows immediately from the observation that the function $f(x) = x - \left\lceil \sqrt{2x} \right\rceil$ can be iterated on $n - s_j$ at most $O(\sqrt{n - s_j})$ times before reaching the value of 3 or less and the function $g(x) = x - \left\lceil \sqrt{x \cdot \log_j x} \right\rceil$ can be iterated on s_j at most $O(\sqrt{\frac{s_j}{\log_j s_j}})$ times before reaching the value of 3 or less. As each leaf of T_G is determined by the positions of the right turns on the path leading to it from the root, we get that the number of leaves in T_G is at most $n^{O\left(\sqrt{n-s_j}+\sqrt{\frac{s_j}{\log_j s_j}}\right)}$. The total number of nodes in T_G is therefore at most $n^{O\left(\sqrt{n-s_j}+\sqrt{\frac{s_j}{\log_j s_j}}\right)}$. As the cost of each node is at most $\left(\mathsf{dom}_{n,n-s_j}(\ell, s)\right) + O(1)$, we immediately have that

$$T(s_j, n - s_j) \leq n^{O\left(\sqrt{n-s_j}+\sqrt{\frac{s_j}{\log_j s_j}}\right)} \cdot \left(\mathsf{dom}_{n,n-s_j}(\ell, s) + O(1)\right). \qquad \square$$

Together with Lemma 104, we obtain

$$T(s_j, n - s_j) = n^{O\left(\sqrt{n-s_j}+\sqrt{\frac{s_j}{\log_j s_j}}\right)}.$$

This completes the proof of Lemma 112.

Chapter 4

Conclusion and Open Problems

We have studied several clustering problems and parity games. We proposed and analyzed new algorithms for the Euclidean k-center Problem, several constrained clustering problems, and parameterized parity games.

Our algorithms make progress towards efficiently solving the mentioned problems, as we showed that an optimal solution to the Euclidean k-center Problem can deterministically be computed, proposed, among a bicriteria approximation algorithm for fair clustering, several constant-factor approximation algorithms for constrained clustering problems, and designed a fixed-parameter-tractable algorithm for parity games.

In the following we discuss open questions in regard to the problems discussed in this dissertation.

4.1 Clustering

With the abundance of clustering problems that arise when one of the clustering objectives is combined with any set of constraints, it is not hard to find open problems, as only in very few cases, like the unconstrained k-center problem, approximation algorithms with matching hardness results are known. In a lot of constrained cases, especially for k-median and k-means it is still an open question to find any constant-factor approximation algorithm at all.

We will specifically discuss open questions which are closely related to the clustering problems discussed in this dissertation.

4.1.1 The Euclidean k-Center Problem

We showed that rational instances of the smallest enclosing ball problem can be solved in weakly polynomial time. We used this knowledge to reduce the Euclidean k-center problem to the problem of finding the correct partition. We also showed that we can find the correct partition with a run time that is, in case we have $k = 1$, or both k and the dimension m are bounded by a fixed constant, polynomial in the size of the input. This was done by showing that we can compute a set of candidate partitionings, which contains an optimal partitioning and solving the problem for each of the candidate partitionings. It has already been known that the Euclidean k-center problem is easy for $m = 1$ and that it is an NP-hard problem for any $m \geq 2$ (and general k). Therefore an obvious open question is if we can find an algorithm for the Euclidean k-center problem, whose run time does not exponentially depend on the dimension m and is polynomial in case k is bounded by a fixed constant. Even for $k = 2$ this is an interesting question. One way to address this question would be to find a smaller set of candidate partitionings. It would also be interesting to find an algorithm that improves the overall run time for the general Euclidean k-center problem and is possibly even fixed-parameter-tractable for the combined parameters k and m.

4.1.2 Fair Clustering

We have studied several types of fair clustering problems for different clustering objectives. We obtained essentially fair approximation algorithms for the fair k-center/k-supplier/facility location/k-median/k-means problem and approximation algorithms for the fair k-center/k-supplier problem with exact fairness, which improved on the best previously known approximation ratio. Besides improving the approximation guarantees and/or obtaining better hardness results for any of the fair clustering problems, it is still an open task to find any true approximation algorithm for the fair k-center/k-supplier/facility location/k-median/k-means problem with relaxed fairness. For the fair facility location/k-median/k-means problem with exact fairness it is also still an open question if true approximation algorithms exist. As we have shown in Section 2.2.1, we could create an approximation algorithm for the fair facility location/k-median/k-means problem with the help of an approximation algorithm for the fairlet decomposition problem. We have also shown how we could, given an approximation algorithm for the capacitated clustering problem with the same objective, compute an approximation to the fairlet decomposition problem. This directly ties the task to find an approximation algorithm for capacitated clustering problems to finding approximation algorithms for fair clustering problems with exact fairness and specifically makes finding approximation algorithms for the capacitated facility location/k-median/k-means problem very interesting related problems.

4.1.3 Private Clustering

We have studied the k-center problem with capacities, fairness and outliers and have coupled these constraints with privacy; in addition, we proposed the strongly private k-center problem. An obvious open question is to improve the approximation guarantee of the coupling process; this is in particular interesting when combining more than two constraints as in the private capacitated and fair k-center problem. Another straightforward direction would be to study the generalization of privacy to arbitrary lower bounds, where each cluster has its individual lower bound on the number of necessary points to assign to it when opened. It would also be interesting to study general methods to add other constraints to clustering problems. And of course, extending our methods to other clustering objectives is open, too. Our algorithms rely on the threshold graph; removing it seems difficult at first glance. However, in Section 2.3.4, we demonstrated how to add privacy to capacitated *facility location*, albeit under a restriction: The method only works if the lower bound ℓ and all upper bounds $u(c)$ satisfy $\ell \leq u(c)/2$. If this is not true, then it induces a capacity violation (by a factor of at most 2). We raise the question whether adding privacy to facility location can be done without this condition. The method in Section 2.3.4 does not easily extend to variants with a restricted number of centers; so the next question then would be whether it can be combined with the idea we developed for k-center, in order to add privacy for an objective like k-median or k-means.

4.2 Parity Games

We have studied parity games and developed algorithms for two sets of parameters. A fixed-parameter-tractable algorithm for the combined parameter the minimum number of nodes owned by one of the players and the number of different priorities as well as algorithms for several out-degree based parameters.

The problem of solving parity games belongs to the class $\mathsf{NP} \cap \mathsf{coNP}$. It has been said that such problems are *well characterized* [60] and that it seems very unlikely that they are NP-complete. Yet it is still a major open question if parity games can be solved in polynomial time. Other open questions include if non-bipartite parity games are fixed-parameter-tractable for the parameter of the minimum number of nodes owned by one of the players as well as finding fixed-parameter-tractable algorithms for other reasonable parameters for parity games.

Appendix A

Approximation Algorithm for the k-Center Problem [47]

We will show that the 2-approximation algorithm for the k-center problem by Gonzalez [47] also works in case we have $P \subseteq L$. We will start by stating how the algorithm works.

The algorithms consists of an *initialization* followed by $k-1$ *expanding* phases. In the initialization the algorithm simply assigns all points to the same cluster with one of the points arbitrary selected as the center c_1. In every expanding phase the algorithm creates one additional cluster and assigns some points from the existing clusters to the new cluster. The creation of the additional cluster in the i-th expanding phase starts by selecting its center c_{i+1}. c_{i+1} is chosen as the point from the existing clusters with the highest distance to its center, with ties broken arbitrarily, i.e., $c_{i+1} = \arg\max_{j \in P} d(j, \phi_i(j))$, where ϕ_i denotes the cluster assignment function before the i-th expanding phase. A point $j \in P$ will then be assigned to c_{i+1} if $d(j, c_{i+1})$ is less than $d(j, \phi_i(j))$.

Gonzalez showed that this procedure creates a partition of the points into k clusters in polynomial time. We will repeat a slight alteration of the proof of the obtained approximation ratio.

Let S denote the centers by the method described above. Let $j_{max} = \arg\max_{j \in P} d(j, \phi_k(j))$ be the point with the highest distance to its assigned center and let $r = d(j_{max}, \phi_k(j_{max}))$. Without loss of generality we can assume $r > 0$ and therefore j_{max} is not one of the selected centers. As during the procedure every point is always assigned to its closest open center and j_{max} was not selected as a center we know that in every phase the selected center had a distance of at least r to its current center and therefore to all previous centers. This implies that S together with j_{max} form a set of $k+1$ points with a pairwise distance between two of these points of at least r. As in every cluster with at most k clusters at least 2 of these points must be part of the same cluster, the optimal solution must contain a cluster which contains two points with a distance of at least r two each other. Therefore the distance between the center of that cluster and one of the points must be at least $r/2$, which shows that our computed solution is in fact a 2-approximation.

Appendix B

Facts about the k-Means Cost Function

We use some well-known facts about the k-means function when extending our results for k-median to k-means. The first one is that squared distances satisfy a relaxed triangle inequality:

Lemma 114. *It holds for all $x, y, z \in \mathbb{R}^m$ that*

$$||x - z||^2 \leq 2||x - z||^2 + 2||z - y||^2.$$

The next lemma is also a folklore statement which can be extremely useful. It implies that the best 1-means is always the centroid of a point set, and has further consequences, like Lemma 116 which we state below, a fact which is also commonly used in approximation algorithms for the k-means problem.

Lemma 115. *For any $P \subseteq \mathbb{R}^m$, and $z \in \mathbb{R}^m$,*

$$\sum_{j \in P} ||j - z||^2 = \sum_{j \in P} ||j - \mu(P)||^2 + |P| \cdot ||\mu(P) - z||^2,$$

where $\mu(P) = \frac{1}{|P|} \sum_{j \in P} j$ is the centroid of P.

One corollary of Lemma 115 is that the optimum cost of the best discrete solution is not much more expensive than the best choice of centers from $\mathbb{R}^m$.

Lemma 116. *Let $P \subseteq \mathbb{R}^m$ be a set of point in the Euclidean space, and let $C^* \subseteq \mathbb{R}^m$ be a set of k points that minimizes the k-means objective, i.e., it minimizes*

$$\sum_{j \in P} \min_{c \in C} ||j - c||^2$$

over all choices of $C \subseteq \mathbb{R}^m$ with $|C| = k$. Furthermore, let $\hat{C}$ be the set of centers that minimizes the k-means objective over all choices of $C \subseteq P$ with $|C| = k$, i.e., the best choice of centers from P itself. Then it holds that

$$\sum_{j \in P} \min_{c \in \hat{C}} ||j - c||^2 \leq 2 \sum_{j \in P} \min_{c \in C^*} ||j - c||^2.$$

Thus, restricting the set of centers to the input point set increases the cost of an optimal solution by a factor of at most 2.

Appendix C

Fair k-Center Analysis by Chierichetti et al. [29]

We will briefly explain the method and analysis by Chierichetti et al. [29] for the special case of the fair k-center problem, where the points are colored in two different colors, with exactly half of the points being colored in each color. We will show a small miscalculation in their analysis, wherefore instead of the claimed 3-approximation, their analysis only yields a 4-approximation. As mentioned in Section 2.2.1 the general method introduced by Chierichetti et al.[29] consists of two steps. In the first step computes a fairlet decomposition and in the second step a clustering is computed on a set containing a representative of each fairlet. After these two steps each representative is replaced by its fairlet to obtain the final clustering. They correctly showed that the maximal radius in this final clustering is bound by the maximal radius of the clustering on the representatives and the cost of the fairlet decomposition. They also showed that it is possible to compute a clustering on the representatives that has a radius of at most $2\,\mathsf{opt}$. n the case where each fairlet contains exactly 2 points, Chierichetti et al, defined the cost of a fairlet as the distance between the two points and defined the cost of a fairlet decomposition as the maximal cost of any of its fairlets. As mentioned in Section 2.2.1, they then showed how to compute a minimum cost fairlet decomposition by solving a minimum cost perfect matching problem.

Yet, because they defined the cost of a fairlet as the distance between the two points, which is equivalent to restricting the center of a fairlet to one of the points, the cost of their fairlets is only bound by the maximal diameter of a fair clustering. The following example (see Figure C.1) shows that it can happen that this is twice the radius of the optimal solution. Let P contain two red (r_1 and r_2) and two blue points (b_1 and b_2) located in the 1-dimensional euclidean space. Let r_1 and r_2 be located at 0, b_1 at 1 and b_2 at 2 and let the pairwise distance between two of the points be the euclidean distance of their locations. The optimal fair solution with 1 cluster then chooses b_1 as its center and has a radius of 1, but in every fairlet decomposition one of the fairlets must contain only b_2 and either r_1 or r_2. in both cases the cost of this fairlet ant therefore the fairlet decomposition is $2 = 2\,\mathsf{opt}$.

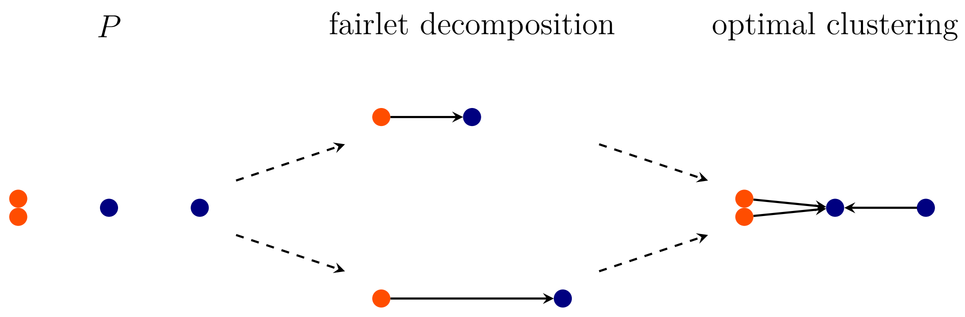

Figure C.1: Example for the fairlet decomposition analysis by Chierichetti et al.[29]

Appendix D

Notation

Object	**Notation**
A directed graph	$G = (V_0 \uplus V_1, E)$
Number of nodes	$n = \lvert V_0 \uplus V_1 \rvert$
Number of edges	$m = \lvert E \rvert$
Nodes in G belonging to player $i \in \{0, 1\}$	V_i
minimum number of nodes	
controlled by one player	$k = \min_{i \in \{0,1\}} \lvert V_i \rvert$
Priority function	$p : V_0 \uplus V_1 \to \mathbb{N}_0$
A parity game	$G = (V_0 \uplus V_1, E, p)$
Number of distinct priorities	p
Largest priority	$p_{\max}$
Set of nodes with the largest priority	$V_{p_{\max}}$
Maximum out-degree	$\Delta_{\max}$
Average out-degree	Δ_{avg}
Number of nodes with out-degree at most j	s_j
Set of out-neighbors of v	$N^+_G(v) = \{w \in V_0 \uplus V_1 \mid (v, w) \in E\}$
Set of in-neighbors of v	$N^-_G(v) = \{u \in V_0 \uplus V_1 \mid (u, v) \in E\}$
A strategy for player i	s_i
Winning set of player i on G	$win_i(G)$
i-reachability set of a set A	$reach_i(A)$

Table D.2: Objects relevant for our Parity Games results and their notation

Object	**Notation**
Set of points	P
Set of possible locations	L
Distance function	$d : (P \cup L) \times (P \cup L) \to \mathbb{R}_{\geq 0}$
Opening costs	f_i
A point	j or p_j
A center	i or c_i
Number of points	n
Number of allowed clusters	k
Set of chosen locations/ set of centers	S
Assignment of points to centers	$\phi : P \to S$
A clustering	$\mathcal{C} = (S, \phi)$
A cluster in a clustering	$P(i) = \{p \in P \mid \phi(p) = c_i\}$
Value of the optimal clustering	opt
Euclidean space	$\mathbb{R}^m$
Dimension of an Euclidean space	m
Set of colors	Col
A color	h or col_h
Color assignment	$col : P \to Col$
Points with color col_h in P'	$col_h(P') = \{j \in P' \mid col(j) = col_h\}$
Amount	
-of points with color col_h in P'	$\text{mass}_h(P') = \lvert col_h(P')\rvert$
-of color col_h assigned to i by (x, y)	$\text{mass}_h(x, i) = \sum_{j \in col_h(P)} x_{ij}$
-assigned to i by (x, y)	$\text{mass}(x, i) = \sum_{j \in P} x_{ij}$
Ratio of points with color col_h in a set P'	$r_h(P') = \frac{\lvert col_h(P')\rvert}{\lvert P'\rvert}$
Value of the optimal clustering	opt
Relaxed boarders for the ratio	$\ell_h = \frac{p_1^h}{q_1^h}, u_h = \frac{p_2^h}{q_2^h} \in \mathbb{Q}_{\geq 0}$
A fairlet decomposition	$\mathcal{F}$
A fairlet	F_i
Lower bound for points per cluster	ℓ
Upper bound for points per cluster	u
Maximum number of outliers	o
Threshold	τ
Threshold graph with threshold τ on P	$G_\tau = (P \cup L, E_\tau)$
Edges in the threshold graph	$E_\tau = \{(i, j) \mid i \in L, j \in P, d(i, j) \leq \tau\}$
Sphere with radius r centered at c	$B(c, r) = \{x \in \mathbb{R}^m \mid dist(x, c) \leq r\}$
Boundary of $B(c, r)$	$\delta(B(c, r)) := \{x \in \mathbb{R}^m \mid dist(x, c) = r\}$
Smallest sphere	
-centered at c containing S	$B(c, S) = B(c, \max_{p \in S} dist(p, c))$
-containing $S \subseteq \mathbb{R}^m$	$B(S)$
-containing T on its boundary	$B_\Delta(T)$
Circumcenter of a set T,	$cc(T)$ (the center of $B_\Delta(T)$)
Affine hull of a set T	$aff(T)$

Table D.1: Objects relevant for our Clustering results and their notation

Bibliography

[1] Karen Aardal, Pieter L. van den Berg, Dion Gijswijt, and Shanfei Li. Approximation algorithms for hard capacitated k-facility location problems. *European Journal of Operational Research*, 242:358–368, 2015. `doi:10.1016/j.ejor.2014.10.011`.

[2] Ankit Aggarwal, Louis Anand, Manisha Bansal, Naveen Garg, Neelima Gupta, Shubham Gupta, and Surabhi Jain. A 3-approximation algorithm for the facility location problem with uniform capacities. *Mathematical Programming*, 141:527–547, 2013. `doi:10.1007/s10107-012-0565-4`.

[3] Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Samir Khuller, Rina Panigrahy, Dilys Thomas, and An Zhu. Achieving anonymity via clustering. *ACM Transactions on Algorithms*, 6:49:1–49:19, 2010. `doi:10.1145/1798596.1798602`.

[4] Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. Approximation algorithms for k-anonymity. In *Proceedings of the International Conference on Database Theory (ICDT 2005)*, November 2005. URL: `http://ilpubs.stanford.edu:8090/645/`.

[5] Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and Euclidean k-median by primal-dual algorithms. In Chris Umans, editor, *Proceedings of the 58th IEEE Symposium on Foundations of Computer Science (FOCS 2017)*, pages 61–72. IEEE Computer Society, 2017. `doi:10.1109/FOCS.2017.15`.

[6] Sara Ahmadian and Chaitanya Swamy. Improved approximation guarantees for lower-bounded facility location. In Thomas Erlebach and Giuseppe Persiano, editors, *10th International Workshop on Approximation and Online Algorithms (WAOA 2012)*, volume 7846 of *Lecture Notes in Computer Science (LNCS)*, pages 257–271. Springer Berlin Heidelberg, 2012. `doi:10.1007/978-3-642-38016-7_21`.

[7] Sara Ahmadian and Chaitanya Swamy. Approximation algorithms for clustering problems with lower bounds and outliers. In Ioannis Chatzigiannakis, Michael

Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 69:1–69:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.69.

[8] Hyung-Chan An, Aditya Bhaskara, Chandra Chekuri, Shalmoli Gupta, Vivek Madan, and Ola Svensson. Centrality of trees for capacitated k-center. *Mathematical Programming*, 154:29–53, 2015. doi:10.1007/s10107-014-0857-y.

[9] Hyung-Chan An, Mohit Singh, and Ola Svensson. LP-based algorithms for capacitated facility location. *SIAM Journal on Computing*, 46:272–306, 2017. doi:10.1137/151002320.

[10] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.

[11] Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The hardness of approximation of euclidean k-means. In Lars Arge and János Pach, editors, *31st International Symposium on Computational Geometry (SoCG 2015)*, volume 34 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 754–767. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.SOCG.2015.754.

[12] Mihai Bādoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proc. STOC 2002*, pages 250–257, 2002.

[13] Manisha Bansal, Naveen Garg, and Neelima Gupta. A 5-approximation for capacitated facility location. In Leah Epstein and Paolo Ferragina, editors, *Algorithms – ESA 2012*, volume 7501 of *Lecture Notes in Computer Science (LNCS)*, pages 133–144. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-33090-2_13.

[14] Judit Bar-Ilan, Guy Kortsarz, and David Peleg. How to allocate network centers. *Journal of Algorithms*, 15:385–415, 1993. doi:10.1006/jagm.1993.1047.

[15] A. Ben-Hur, D. Horn, H.T. Siegelmann, and V. Vapnik. Support vector clustering. *J. Machine Learning Res.*, 2:125–137, 2001.

[16] Ioana O. Bercea, Martin Groß, Samir Khuller, Aounon Kumar, Clemens Rösner, Daniel R. Schmidt, and Melanie Schmidt. On the Cost of Essentially Fair Clusterings. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*, volume 145 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:22, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: http://drops.dagstuhl.de/opus/volltexte/2019/11233, doi:10.4230/LIPIcs.APPROX-RANDOM.2019.18.

[17] Dietmar Berwanger, Anuj Dawar, Paul Hunter, and Stephan Kreutzer. DAG-width and parity games. *Proc. STACS 2006*, 3884:524–536, 2006. URL: `http://dx.doi.org/10.1007/11672142_43`, `doi:10.1007/11672142_43`.

[18] Dietmar Berwanger and Erich Grädel. Fixed-point logics and solitaire games. *Theory Comput. Syst.*, 37(6):675–694, 2004. URL: `http://dx.doi.org/10.1007/s00224-004-1147-5`.

[19] Dietmar Berwanger, Erich Grädel, Łukasz Kaiser, and Roman Rabinovich. Entanglement and the complexity of directed graphs. *Theoret. Comput. Sci.*, 463:2–25, 2012. URL: `http://dx.doi.org/10.1016/j.tcs.2012.07.010`, `doi:10.1016/j.tcs.2012.07.010`.

[20] Henrik Björklund, Sven Sandberg, and Sergei Vorobyov. A discrete subexponential algorithm for parity games. *Proc. STACS 2003*, 2607:663–674, 2003. URL: `http://dx.doi.org/10.1007/3-540-36494-3_58`, `doi:10.1007/3-540-36494-3_58`.

[21] Y. Bulatov, S. Jambawalikar, P. Kumar, and S Sethia. Hand recognition using geometric classifiers. *Abstract of presentation for the DIMACS Workshop on Computational Geometry (Rutgers University)*, 2002.

[22] Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median and positive correlation in budgeted optimization. *ACM Transactions on Algorithms*, 13:23:1–23:31, 2017. `doi:10.1145/2981561`.

[23] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *STOC'17—Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 252–263. ACM, New York, 2017.

[24] Deeparnab Chakrabarty, Prachi Goyal, and Ravishankar Krishnaswamy. The non-uniform k-center problem. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 67:1–67:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. `doi:10.4230/LIPIcs.ICALP.2016.67`.

[25] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002.

[26] Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In S. Rao Kosaraju, editor, *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms (SODA 2001)*, pages 642–651, 2001. URL: `http://dl.acm.org/citation.cfm?id=365411.365555`.

[27] Danny Z. Chen, Jian Li, Hongyu Liang, and Haitao Wang. Matroid and knapsack center problems. *Algorithmica*, 75(1):27–52, 2016.

[28] Ke Chen. A constant factor approximation algorithm for k-median clustering with outliers. In Shang-Hua Teng, editor, *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008)*, pages 826–835. SIAM, 2008. URL: `http://dl.acm.org/citation.cfm?id=1347082.1347173`.

[29] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017 (NIPS 2017)*, pages 5036–5044, 2017. URL: `http://papers.nips.cc/paper/7088-fair-clustering-through-fairlets`.

[30] Marek Cygan, Mohammad Taghi Hajiaghayi, and Samir Khuller. LP rounding for k-centers with non-uniform hard capacities. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science—FOCS 2012*, pages 273–282. IEEE Computer Soc., Los Alamitos, CA, 2012.

[31] Marek Cygan and Tomasz Kociumaka. Constant factor approximation for capacitated k-center with outliers. In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 251–262. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014. `doi:10.4230/LIPIcs.STACS.2014.251`.

[32] Hu Ding, Lunjia Hu, Lingxiao Huang, and Jian Li. Capacitated center problems with two-sided bounds and outliers. In *Proceedings of the 15th International Symposium on Algorithms and Data Structures (WADS)*, pages 325–336, 2017.

[33] Hu Ding and Jinhui Xu. Solving the chromatic cone clustering problem via minimum spanning sphere. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 773–784, 2011.

[34] Hu Ding and Jinhui Xu. A unified framework for clustering constrained data without locality property. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1471–1490, 2015.

[35] Zvi Drezner. The p-centre problem—heuristic and optimal algorithms. *Journal of The Operational Research Society - J OPER RES SOC*, 35:741–748, 08 1984. `doi:10.1057/jors.1984.150`.

[36] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. Fairness through awareness. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012 (ITCS 2012)*, pages 214–226. ACM, 2012. `doi:10.1145/2090236.2090255`.

[37] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. FOCS 1991*, pages 368–377, 1991. URL: `http://dx.doi.org/10.1109/SFCS.1991.185392`, `doi:10.1109/SFCS.1991.185392`.

[38] John Fearnley and Oded Lachish. Parity games on graphs with medium tree-width. *Proc. MFCS 2011*, 6907:303–314, 2011. URL: `http://dx.doi.org/10.1007/978-3-642-22993-0_29`, `doi:10.1007/978-3-642-22993-0_29`.

[39] John Fearnley and Sven Schewe. Time and parallelizability results for parity games with bounded treewidth. *Proc. ICALP 2012*, 7392:189–200, 2012. URL: `http://dx.doi.org/10.1007/978-3-642-31585-5_20`, `doi:10.1007/978-3-642-31585-5_20`.

[40] Kaspar Fischer. The smallest enclosing ball of balls. Master's thesis, Institute of Theoretical Computer Schience, ETH Zürich, 2001.

[41] Kaspar Fischer, Bernd Gärtner, and Martin Kutz. Fast smallest-enclosing-ball computation in high dimensions. *Proc. ESA 2003*, 2832:630–641, 2003.

[42] Zachary Friggstad, Mohsen Rezapour, and Mohammad R. Salavatipour. Approximating connected facility location with lower and upper bounds via LP rounding. In *15th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 1:1–1:14, 2016.

[43] Jakub Gajarský, Michael Lampis, Kazuhisa Makino, Valia Mitsou, and Sebastian Ordyniak. Parameterized algorithms for parity games. In *Proc. MFCS 2015*, Lecture Notes Comput. Sci., pages 336–347, 2015.

[44] Bernd Gärtner. Fast and robust smallest enclosing balls. In *Proc. ESA 1999*, volume 1643 of *Lecture Notes Comput. Sci.*, pages 325–338, 1999.

[45] Bernd Gärtner and Sven Schönherr. An efficient, exact, and generic quadratic programming solver for geometric optimization. In *Proc. SoCG 2000*, pages 110–118 (electronic), 2000.

[46] Ashish Goel, Piotr Indyk, and Kasturi Varadarajan. Reductions among high dimensional proximity problems. In *Proc. SODA 2001*, pages 769–778, 2001.

[47] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985. `doi:10.1016/0304-3975(85)90224-5`.

[48] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes Comput. Sci.* Springer, 2002.

[49] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2 of *Algorithms and Combinatorics: Study and Research Texts.* Springer-Verlag, Berlin, 1988. URL: `http://dx.doi.org/10.1007/978-3-642-97881-4`, `doi:10.1007/978-3-642-97881-4`.

[50] Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31:228–248, 1999. doi:10.1006/jagm.1998.0993.

[51] P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, s1-10(1):26–30, 1935. URL: http://dx.doi.org/10.1112/jlms/s1-10.37.26, doi:10.1112/jlms/s1-10.37.26.

[52] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, pages 3315–3323, 2016.

[53] Dorit S. Hochbaum and David B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM*, 33(3):533–550, 1986. doi:10.1145/5925.5933.

[54] Wen-Lian Hsu and George L. Nemhauser. Easy and hard bottleneck location problems. *Discrete Applied Mathematics*, 1:209–215, 1979. doi:10.1016/0166-218X(79)90044-1.

[55] R. Z. Hwang, R. C. T. Lee, and R. C. Chang. The slab dividing approach to solve the Euclidean P-center problem. *Algorithmica*, 9(1):1–22, 1993. URL: https://doi.org/10.1007/BF01185335, doi:10.1007/BF01185335.

[56] Hiroshi Imai and Mary Inaba. The number of partitions of n points induced by the voronoi diagram via the conjugacy generated by k points. *Proc. 1st Japanese-Hungarian Symp. on Discrete Mathematics and Its Applications*, page 83–90, 03 1999.

[57] Hiroshi Imai, Mary Inaba, and Naoki Katoh. Variance-based k-clustering algorithms by voronoi diagrams and randomization. *IEICE Transactions on Information and Systems*, E83D, 06 2000.

[58] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing (STOC 2002)*, pages 731–740. ACM, 2002. doi:10.1145/509907.510012.

[59] Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Journal of the ACM*, 48(2):274–296, 2001.

[60] David S. Johnson. The NP-completeness column: finding needles in haystacks. *ACM Trans. Algorithms*, 3(2):Art. 24, 21, 2007. URL: https://doi.org/10.1145/1240233.1240247, doi:10.1145/1240233.1240247.

[61] Marcin Jurdziński. Deciding the winner in parity games is in UP ∩ co-UP. *Inform. Process. Lett.*, 68(3):119–124, 1998. URL: http://dx.doi.org/10.1016/S0020-0190(98)00150-1, doi:10.1016/S0020-0190(98)00150-1.

[62] Marcin Jurdziński. Small progress measures for solving parity games. *Proc. STACS 2000*, 1770:290–301, 2000. URL: `http://dx.doi.org/10.1007/3-540-46541-3_24`, `doi:10.1007/3-540-46541-3_24`.

[63] Marcin Jurdziński and Ranko Lazić. Succinct progress measures for solving parity games. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, page 9. IEEE, [Piscataway], NJ, 2017.

[64] Marcin Jurdziński, Mike Paterson, and Uri Zwick. A deterministic subexponential algorithm for solving parity games. *SIAM J. Comput.*, 38(4):1519–1532, 2008. URL: `http://dx.doi.org/10.1137/070686652`, `doi:10.1137/070686652`.

[65] Samir Khuller, Robert Pless, and Yoram J. Sussmann. Fault tolerant k-center problems. *Theoretical Computer Science*, 242(1-2):237–245, 2000.

[66] Samir Khuller and Yoram J. Sussmann. The capacitated k-center problem. *SIAM Journal on Discrete Mathematics*, 13:403–418, 2000. `doi:10.1137/S0895480197329776`.

[67] Valerie King, Orna Kupferman, and Moshe Y. Vardi. On the complexity of parity word automata. *Proc. FOSSACS 2001*, 2030:276–286, 2001.

[68] Ton Kloks and Hans L. Bodlaender. On the treewidth and pathwidth of permutation graphs, 1992. `http://www.cs.uu.nl/research/techreps/repo/CS-1992/1992-13.pdf`.

[69] Madhukar R. Korupolu, C. Greg Plaxton, and Rajmohan Rajaraman. Analysis of a local search heuristic for facility location problems. *Journal of Algorithms*, 37(1):146–188, 2000.

[70] Ravishankar Krishnaswamy, Shi Li, and Sai Sandeep. Constant approximation for k-median and k-means with outliers via iterative rounding. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2018)*, pages 646–659. ACM, 2018. `doi:10.1145/3188745.3188882`.

[71] Piyush Kumar, Joseph S. B. Mitchell, and E. Alper Yıldırım. Approximate minimum enclosing balls in high dimensions using core-sets. *ACM J. Exp. Algorithmics*, 8:29 pp. (electronic), 2003.

[72] Euiwoong Lee, Melanie Schmidt, and John Wright. Improved and simplified inapproximability for k-means. *Information Processing Letters*, 120:40–43, 2017. `doi:10.1016/j.ipl.2016.11.009`.

[73] Retsef Levi, David B. Shmoys, and Chaitanya Swamy. LP-based approximation algorithms for capacitated facility location. In *Integer programming and combinatorial optimization*, volume 3064 of *Lecture Notes in Comput. Sci.*, pages 206–218. Springer, Berlin, 2004. URL: `https://doi.org/10.1007/978-3-540-25960-2_16`, `doi:10.1007/978-3-540-25960-2_16`.

[74] Jian Li, Ke Yi, and Qin Zhang. Clustering with diversity. In *Automata, languages and programming. Part I*, volume 6198 of *Lecture Notes in Comput. Sci.*, pages 188–200. Springer, Berlin, 2010. URL: `https://doi.org/10.1007/978-3-642-14165-2_17`, `doi:10.1007/978-3-642-14165-2_17`.

[75] Shanfei Li. An improved approximation algorithm for the hard uniform capacitated k-median problem. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*, volume 28 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 325–338. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014. `doi:10.4230/LIPIcs.APPROX-RANDOM.2014.325`.

[76] Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013. `doi:10.1016/j.ic.2012.01.007`.

[77] Shi Li. Approximating capacitated k-median with $(1+\varepsilon)k$ open facilities. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016)*, pages 786–796. SIAM, 2016. `doi:10.1137/1.9781611974331.ch56`.

[78] Shi Li. On uniform capacitated k-median beyond the natural LP relaxation. *ACM Transactions on Algorithms*, 13:22:1–22:18, 2017. `doi:10.1145/2983633`.

[79] Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. *SIAM Journal on Computing*, 45:530–547, 2016. `doi:10.1137/130938645`.

[80] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4-5):498–516, 1996.

[81] Robert McNaughton. Infinite games played on finite graphs. *Ann. Pure Appl. Logic*, 65(2):149–184, 1993. URL: `http://dx.doi.org/10.1016/0168-0072(93)90036-D`, `doi:10.1016/0168-0072(93)90036-D`.

[82] N. Megiddo, A. Tamir, E. Zemel, and R. Chandrasekaran. An $O(n\log^2 n)$ algorithm for the kth longest path in a tree with applications to location problems. *SIAM J. Comput.*, 10(2):328–337, 1981. URL: `https://doi.org/10.1137/0210023`, `doi:10.1137/0210023`.

[83] Nimrod Megiddo. The weighted Euclidean 1-center problem. *Math. Oper. Res.*, 8(4):498–504, 1983. URL: `https://doi.org/10.1287/moor.8.4.498`.

[84] Nimrod Megiddo and Kenneth J. Supowit. On the complexity of some common geometric location problems. *SIAM J. Comput.*, 13(1):182–196, 1984. URL: `https://doi.org/10.1137/0213014`, `doi:10.1137/0213014`.

[85] Matthias Mnich, Heiko Röglin, and Clemens Rösner. New deterministic algorithms for solving parity games. *Discrete Optim.*, 30:73–95, 2018. URL: `https://doi.org/10.1016/j.disopt.2018.06.001`, `doi:10.1016/j.disopt.2018.06.001`.

[86] Jan Obdržálek. Fast μ-calculus model checking when tree-width is bounded. *Proc. CAV 2003*, 2725:80–92, 2003. URL: `http://dx.doi.org/10.1007/978-3-540-45069-6_7`, `doi:10.1007/978-3-540-45069-6_7`.

[87] Jan Obdržálek. Clique-width and parity games. *Proc. CSL 2007*, 4646:54–68, 2007. URL: `http://dx.doi.org/10.1007/978-3-540-74915-8_8`, `doi:10.1007/978-3-540-74915-8_8`.

[88] Anthony L. Peressini, Francis E. Sullivan, and J. J. Uhl, Jr. *The mathematics of nonlinear programming.* Undergraduate Texts in Mathematics. Springer-Verlag, New York, 1988.

[89] Andrea Romei and Salvatore Ruggieri. A multidisciplinary survey on discrimination analysis. *The Knowledge Engineering Review*, 29:582–638, 2014. `doi:10.1017/S0269888913000039`.

[90] Clemens Rösner and Melanie Schmidt. Privacy Preserving Clustering with Constraints. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 96:1–96:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. `doi:10.4230/LIPIcs.ICALP.2018.96`.

[91] Sven Schewe. Solving parity games in big steps. *Proc. FSTTCS 2007*, 4855:449–460, 2007. URL: `http://dx.doi.org/10.1007/978-3-540-77050-3_37`, `doi:10.1007/978-3-540-77050-3_37`.

[92] Raimund Seidel. Linear programming and convex hulls made easy. In *Proceedings of the Sixth Annual Symposium on Computational Geometry*, SCG '90, pages 211–215, New York, NY, USA, 1990. ACM. URL: `http://doi.acm.org/10.1145/98524.98570`, `doi:10.1145/98524.98570`.

[93] David B. Shmoys, Éva Tardos, and Karen Aardal. Approximation algorithms for facility location problems. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 265–274, 1997.

[94] Colin Stirling. Local model checking games. *Proc. CONCUR 1995*, 962:1–11, 1995. URL: `http://dx.doi.org/10.1007/3-540-60218-6_1`, `doi:10.1007/3-540-60218-6_1`.

[95] Zoya Svitkina. Lower-bounded facility location. *ACM Transaction on Algorithms*, 6:69:1–69:16, 2010. `doi:10.1145/1824777.1824789`.

[96] Indrė Žliobaitė, Faisal Kamiran, and Toon Calders. Handling Conditional Discrimination. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, pages 992–1001. IEEE Computer Society, 2011. `doi:10.1109/ICDM.2011.72`.

[97] Jens Vöge and Marcin Jurdziński. A discrete strategy improvement algorithm for solving parity games. *Proc. CAV 2000*, 1855:202–215, 2000. URL: `http://dx.doi.org/10.1007/10722167_18`, `doi:10.1007/10722167_18`.

[98] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, pages 577–584, 2001.

[99] Emo Welzl. Smallest enclosing disks (balls and ellipsoids). *New results and new trends in computer science (Graz, 1991)*, 555:359–370, 1991.

[100] D. B. Yudin and A. S. Nemirovskiĭ. Informational complexity and effective methods for the solution of convex extremal problems. *Èkonom. i Mat. Metody*, 12(2):357–369, 1976.

[101] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, volume 28 of *Proceedings of Machine Learning Research*, pages 325–333. PMLR, 2013. URL: `http://proceedings.mlr.press/v28/zemel13.html`.

[102] Guanglu Zhou, Kim-Chuan Toh, and Jie Sun. Efficient algorithms for the smallest enclosing ball problem. *Comput. Optim. Appl.*, 30(2):147–160, 2005. URL: `https://doi.org/10.1007/s10589-005-4565-7`, `doi:10.1007/s10589-005-4565-7`.

[103] Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoret. Comput. Sci.*, 200(1-2):135–183, 1998. URL: `http://dx.doi.org/10.1016/S0304-3975(98)00009-7`, `doi:10.1016/S0304-3975(98)00009-7`.

Index

γ sphere separated, 27
i-closed set, 83

approximation, 3
  algorithm, 3, 5, 108
  bicriteria, 11
  ratio, 3, 11
assignment problem
  fair, 5, 15, 55

black box algorithm, 36, 46

centroid, 35
clustering, 2, 4, 20
  capacitated, 2, 15, 40, 67
    soft, 15
  fair, 2, 8, 15, 31, 71
    essentially, 10, 45
    exact, 15, 51
    fractional, 10
    relaxed, 15
  private, 2, 12, 15, 58
  strongly private, 12, 15, 75
  with outliers, 2, 15, 59
clustering problem
  k-center, 2, 14, 108
    euclidean, 6, 14, 20
  k-means, 2, 14, 110
  k-median, 2, 14
  k-supplier, 14
  facility location, 2, 14, 78
combinatorial optimization, 4
convex body
  centered, 25
convex set
  unbounded, 23

dominion, 82, 91

euclidean space, 6
exact cover by 3-sets, 55

fairlet, 10, 32
  decomposition, 32, 72
fairness violation, 11, 46
fixed parameter tractable, 2, 4
  algorithm, 4, 17
  problem, 4
flow
  maximum, 39, 61
  min-cost, 49
  network, 39, 51, 53

Gaussian elimination, 27
graph
  directed, 16

integrality gap, 5, 56

kernelization, 2, 18, 84

linear program, 44
  integer, 44
local search, 5

optimization problem, 7
  convex, 7, 23
oracle polynomial time, 25

parameterized problem, 3
parity games, 2, 16, 81
  bipartite, 17, 86
partitioning, 20
priority function, 16

recurrence relation, 98, 100

smallest enclosing ball, 20

smallest enclosing ball problem, 7, 20
sphere, 20

threshold graph, 60
thresholding, 12, 59
tree
    binary, 103

unsupervised learning, 4

voronoi
    diagram, 21
    partitioning, 21
    region, 21

weak membership problem, 25
weakly supervised rounding, 10, 51
winning set, 82