Algorithms For Clustering Problems: Theoretical Guarantees and Empirical Evaluations

THÈSE N^{O} 8546 (2018)

PRÉSENTÉE LE 13 JUILLET 2018

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

LABORATOIRE DE THÉORIE DU CALCUL 2

PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Ashkan NOROUZI FARD

acceptée sur proposition du jury:

Prof. F. Eisenbrand, président du jury
Prof. O. N. A. Svensson, directeur de thèse
Prof. D. Shmoys, rapporteur
Prof. C. Mathieu, rapporteuse
Prof. M. Kapralov, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2018

Be happy for this moment.
This moment is your life.
—Omar Khayyam

To my family,
Farideh, Alireza and Anahita...

Acknowledgements

Ph.D. is a long, exhausting and boring journey unless one have great friends and colleagues that make it fun. I would like to express my deepest appreciation to all those who enabled and facilitated this journey for me.

First and foremost, I would like to acknowledge with much appreciation the crucial role that Ola had. Not only is he the greatest advisor that one can hope for, but he is also a good friend. He was always very patient with me trying new ideas even when he did not agree with them, and helped me understand the strengths and weaknesses of different approaches. His guidance was indispensable during my whole time at EPFL, whether in research or while writing the thesis.
I would also like to express my sincere gratitude to Chantal. She was always there when I needed help and when I had any questions. She fixed any mess that I created, all without ever dropping her smile.

Moreover, I would like to thank the jury members of my thesis, Prof. David Shmoys, Prof. Michael Kapralov, Prof. Clair Mathieu, and the president of the jury Prof. Friedrich Eisenbrand, for their careful reading of my thesis and the insightful discussion we had during and after the private defense. I am also grateful to all my co-authors for enabling me to grow as a researcher throughout our work together.

Furthermore, I would like to thank my labmates both at EPFL and Cornell for all the good times that I had. They made it possible for me to work everyday without getting bored and demotivated.
Without my friends I would not have managed to survive my studies. They were always there for me when I needed to blow off some steam, and do something stupid. I would like to thank all my friends in Sharif university who made my bachelor studies the best period of my life, and my friends in Lausanne who were the main reason that I did not quit Ph.D.
Last but not least, I would like to thank my family for their continuous support and unconditional love.

Abstract

Clustering is a classic topic in combinatorial optimization and plays a central role in many areas, including data science and machine learning. In this thesis, we first focus on the dynamic facility location problem (i.e., the facility location problem in evolving metrics). We present a new LP-rounding algorithm for facility location problems, which yields the first constant factor approximation algorithm for the dynamic facility location problem. Our algorithm installs competing exponential clocks on clients and facilities, and connects every client by the path that repeatedly follows the smallest clock in the neighborhood. The use of exponential clocks gives rise to several properties that distinguish our approach from previous LP-roundings for facility location problems. In particular, we use *no clustering* and we enable clients to connect through paths of *arbitrary lengths*. In fact, the clustering-free nature of our algorithm is crucial for applying our LP-rounding approach to the dynamic problem.

Furthermore, we present both empirical and theoretical aspects of the k-means problem. The best known algorithm for k-means with a provable guarantee is a simple local-search heuristic that yields an approximation guarantee of $9+\epsilon$, a ratio that is known to be tight with respect to such methods. We overcome this barrier by presenting a new primal-dual approach that enables us (1) to exploit the geometric structure of k-means and (2) to satisfy the hard constraint that at most k clusters are selected without deteriorating the approximation guarantee. Our main result is a 6.357-approximation algorithm with respect to the standard LP relaxation. Our techniques are quite general and we also show improved guarantees for the general version of k-means where the underlying metric is not required to be Euclidean and for k-median in Euclidean metrics.

We also improve the running time of our algorithm to almost linear running time and still maintain a provable guarantee. We compare our algorithm with K-Means++ (a widely studied algorithm) and show that we obtain better accuracy with comparable and even better running time.

Key words: Linear Programming, Approximation Algorithms, Clustering, Facility Location Problem, k-Median, k-Means

Résumé

Le partitionnement de données est un sujet classique dans l'optimisation combinatoire. Il joue un rôle central dans de nombreux domaines, dont la science des données et l'apprentissage automatique (Machine Learning). Dans cette thèse, nous nous concentrons d'abord sur le problème de l'emplacement dynamique d'installations (c'est-à-dire l'emplacement d'installations dans les métriques évoluantes). Nous présentons un nouvel algorithme d'arrondissement d'optimisation linéaire pour les problèmes de l'emplacement d'installations, qui donne, pour la première fois, un algorithme d'approximation à une constante multiplicative près pour le problème de l'emplacement dynamique d'installations. Notre algorithme pose les minuteurs exponentiels concurrents sur les clients et les installations. Puis, il lie chaque client par une chaîne qui suit le minuteur le moins avancé dans le voisinage. L'utilisation d'un minuteur exponentiel donne lieu à plusieurs caractéristiques favorables qui distingue notre méthode des méthodes précédentes pour l'arrondissement d'optimisation linéaire pour les problèmes de l'emplacement d'installations. En particulier, nous n'utilisons pas de partitionnement et nous permettons aux clients de se connecter par des chaînes de longueur arbitraire. En fait, l'absence de partitionnement dans notre algorithme est essentielle afin d'appliquer notre méthode d'arrondissement au problème dynamique.

De plus, nous présentons les aspects empiriques et théoriques du partitionnement en k-moyennes. Le meilleur algorithme connu pour le k-moyennes avec une garantie pour la plus petite borne démontrable est une simple heuristique de recherche locale qui donne une garantie approximative de $9 + \epsilon$ connue pour ne pas être améliorable par ce genre de méthodes. Nous surmontons cet obstacle en présentant une nouvelle approche primal-dual nous permettant (1) d'exploiter la structure géométrique de k-moyennes, et (2) de satisfaire la contrainte stricte d'avoir un maximum de k partitions sans dégrader la garantie d'approximation. Notre résultat principal est un algorithme 6.357-approximation par rapport à la relaxation standard d'optimisation linéaire. Nos techniques sont plutôt générales. Nous montrons également que notre méthode améliore les garanties pour la version générale de k-moyennes lorsque la métrique sous-jacente n'est pas forcement euclidienne, ou quand nous utilisons le k-médianes dans les métriques euclidiennes.

De plus, nous améliorons la complexité en temps de notre algorithme qui devient presque linéaire tout en maintenant une garantie démontrable. Nous comparons notre algorithme avec K-Means++ (un algorithme largement étudié) et montrons que nous obtenons une meilleure précision avec une complexité en temps comparable, voire meilleure.

Résumé

mots clés : programmation linéaire, algorithme d'approximation, regroupement, Problème de localisation d'installation, k-Median, k-Means

Contents

List of Figures

List of Tables

1 Introduction

Discrete optimization problems are everywhere, from the simple daily tasks that we face, to the algorithms used in high-tech products such as phones and computers. Clustering is one of the famous classes of problems in discrete optimization. Clustering is the task of dividing data into groups of similar objects. Each group, called a cluster, ideally consists of objects that are similar and dissimilar to objects of other groups. For example, assume we are given a set of photos of animals and we want to cluster them according to the type of these animals. In this instance, we want each cluster to have exactly one type of animal, e.g., one cluster for cats, one cluster for dogs, and so on.

Clustering problems play a central role in many areas, including data mining and knowledge discovery, data compression, pattern recognition and classification, and detection of abnormal data. Most of the clustering problems are NP-hard. This means that unless P=NP, there is no algorithm that can solve these problems optimally and efficiently. The notion of speed in theoretical computer science is mostly measured asymptotically with respect to the size of the input n. The running time of an efficient algorithm is formally defined to be polynomial in n. The NP-hardness of these problems have motivated a line of research on finding tight approximation guarantees for such clustering problems. In other words, the goal is to design an algorithm that is fast and finds a "good" solution, despite the fact that it might not find the optimum solution. In this thesis, we are in particular interested in improving the quality of such algorithms in two directions:

1. Designing algorithms with better approximation guarantees: This improves the approximation guarantees that we can achieve in polynomial time by ensuring that the quality of the solution that we get is good even in the worst-case scenario.
2. Designing fast algorithms that can be used in practice: As time is a scarce resource, and given the huge size of datasets these days, improving the running time of algorithms with good approximation guarantees is of great practical importance. Such algorithms not only can be used in practice, but also guarantee a good quality solution in any scenario.

The long line of research into clustering problems, both from the practical and theoretical side, has culminated in a rich literature of algorithms and techniques for this family of problems.

In this thesis we consider three basic clustering problems:

- Dynamic facility location problem, a natural generalization of the facility location problem,
- The k-means and k-median problems, two well-known and well-studied problems.

For these problems, we present algorithms based on linear programs that improve the approximation guarantee of the previous algorithms. Moreover, for the k-means problem, we improve the running time of our algorithm and evaluate its performance in practice.

This chapter serves as a general introduction for the remaining chapters of this thesis, and it introduces the basic notions that will be repeatedly used. In Section 1.1, we define approximation algorithms. Then in Section 1.2, we focus on linear programs (LP) and their dual forms and explain how we can use them in order to design algorithms. Following this, we summarize the contributions of this thesis and give a general outline of the remaining chapters in Section 1.3.

The remaining chapters of this thesis are self-contained, and can be read independently of each other: Each chapter has its own introduction that is the basis for the corresponding problem of interest, and covers the previous works and approaches that were previously used in the literature to tackle it.

1.1 Approximation Algorithms

One of the main challenges of theoretical computer science during the recent decades has been designing efficient algorithms for optimization problems. Due to the intractability of finding optimal solutions efficiently for NP-hard optimization problems (assuming P $\neq$ NP), there has been much interest in designing approximation algorithms.

Definition 1.1.1. *[60] An α-approximation algorithm for an optimization problem is a polynomial-time algorithm that for all instances of the problem produces a solution whose value is within a factor of α of the value of an optimal solution.*

In the remainder of this section, the facility location problem serves as an example of optimization problem as it is closely related to the problems that we study in this thesis. We use this problem to give examples for the concept of approximation algorithms and Linear Programs. In this problem, we have a set of clients $\mathcal{D}$ and a set of facilities $\mathcal{F}$.

There is an opening cost f_i associated with any facility $i \in \mathcal{F}$, and a metric d on the set of clients and facilities, which we refer to as the *connection cost*. The goal is to open a subset $S \subseteq \mathcal{F}$ and to assign each client to an open facility, minimizing the total opening cost and connection cost, i.e., $\min \sum_{i \in S} f_i + \sum_{j \in \mathcal{D}} \min_{i \in S} d(i,j)$. We also assume that the cost function d is metric. For an example on the concept of approximation algorithms, by the above definition, an algorithm is a 2-approximation for the facility location problem, if for any instance of this problem, it returns a solution that costs at most 2 times higher than the cost of the best possible solution.[1] In the following section we focus on Linear Programming and explain their importance in designing approximation algorithms.

1.2 Linear Programming

Linear Programming (LP) plays a central role in designing approximation algorithms. During the past decades, many different techniques have been developed based on LP. First we define integer and linear programs, and then we formulate the facility location problem using them.

Definition 1.2.1. *[60] Each linear program or integer program is formulated in terms of a number of* decision variables*. The variables are constrained by a number of linear inequalities or equalities called* constraints*. Any assignment of real numbers to variables such that all the constraints are satisfied is called a* feasible solution*. For an integer program all the values assigned to the variables should be integer. In addition to the constraints, linear and integer programs are defined by a linear function of the decision variables called the* objective function*. The linear and the integer program seeks to find a feasible solution that either minimizes or maximizes this objective function. Such a solution is called an* optimal solution*.*

The standard Integer Programming (IP) of the facility location problem has two sets of variables: a variable y_i for each facility $i \in \mathcal{F}$ and a variable x_{ij} for each facility-client pair $i \in \mathcal{F}, j \in \mathcal{D}$. The intuition behind these variables is that y_i should indicate whether facility i is opened and x_{ij} should indicate whether client j is connected to facility i. The standard IP can now be formulated as follows.

$$\begin{aligned}
\min \quad & \textstyle\sum_{i \in \mathcal{F}} y_i \cdot f_i + \sum_{i \in \mathcal{F}, j \in \mathcal{D}} x_{ij} \cdot d(i,j) & & \\
\text{s.t.} \quad & \textstyle\sum_{i \in F} x_{ij} \geq 1 & \forall j \in \mathcal{D} & \quad (1.2.1)\\
& x_{ij} \leq y_i & \forall i \in \mathcal{F}, j \in \mathcal{D} & \quad (1.2.2)\\
& x, y \in \{0,1\}. & & \quad (1.2.3)
\end{aligned}$$

The first set of constraints (1.2.1) is that each client should be connected to at least one

[1] After a long series of works, Li [45] designed a 1.488-approximation algorithm for the facility location problem.

facility; the second set of constraints (1.2.2) enforces that clients can only be connected to opened facilities; and the third constraint (1.2.3) is that each client is either connected to a facility or not and similarly each facility is either opened or closed. Notice that solving the above IP gives us an exact solution to the facility location problem. Unfortunately, solving IPs is NP-hard, and we do not know any algorithm that efficiently solve the above IP. Therefore, we relax the third set of constraints and get the following LP.

$$\begin{aligned}
\min \quad & \textstyle\sum_{i\in\mathcal{F}} y_i \cdot f_i + \sum_{i\in\mathcal{F}, j\in\mathcal{D}} x_{ij} \cdot d(i,j) & & \\
\text{s.t.} \quad & \textstyle\sum_{i\in F} x_{ij} \geq 1 & \forall j \in \mathcal{D} & (1.2.4)\\
& x_{ij} \leq y_i & \forall i \in \mathcal{F}, j \in \mathcal{D} & (1.2.5)\\
& 0 \leq x, y \leq 1. & & (1.2.6)
\end{aligned}$$

We remark that this is a relaxation of the original problem, as we have relaxed the constraint that x and y should take Boolean values to any value between zero and one. For future reference, we let OPT_{LP} denote the value of an optimal solution to this relaxation. By solving this LP, we get a fractional solution to the facility location problem. Notice that the cost of such a solution (OPT_{LP}) is no higher than the optimum cost (OPT) and we can solve any LP in polynomial time. Therefore any algorithm that rounds the fractional solution to an actual (integral) solution and increases the cost by at most a factor γ has an approximation guarantee of γ.

A line of research has been dedicated to designing polynomial time algorithms that round such fractional solutions. For the classic facility location problem, a large number of LP-based algorithmic techniques (and their combination) have been successfully applied, including LP-rounding [56, 24, 20, 45], primal-dual methods [40], or dual fitting [38, 37].

Designing algorithms by using the dual form of an LP has also proven to be useful for the facility location problem. The dual form of an LP is also an LP with many interesting properties. In order to obtain a dual form of an LP, we associate a variable for each constraint in the LP. Let α_j be the variable to the constraint $\sum_{i\in F} x_{ij} \geq 1$ for each client j. Similarly let β_{ij} be the variable associated with the constraint $x_{ij} \leq y_i$ for each client j and facility i. We can in general derive a dual linear program for any given linear program, but we will not go into the details of how to do so. Here is the dual form of the above mentioned natural LP relaxation of the facility location problem.

$$\begin{aligned} \max \quad & \textstyle\sum_{j\in\mathcal{D}} \alpha_j & & \\ \text{s.t.} \quad & \textstyle\sum_{j\in\mathcal{D}} \beta_{ij} \le f_i & \forall i \in \mathcal{F} & (1.2.7)\\ & \alpha_j - \beta_{ij} \le c_{ij} & \forall i \in \mathcal{F}, j \in \mathcal{D} & (1.2.8)\\ & \beta, \alpha \ge 0. & & (1.2.9) \end{aligned}$$

Consider any feasible solution to above dual and let $cost_{dual}$ be its cost. By weak duality, it is known that $cost_{Dual} \le \mathrm{OPT}_{LP}$. Consider an algorithm that constructs a feasible dual solution together with an integral solution of cost $cost_{answer}$ for the facility location problem such that $cost_{answer} \le \gamma cost_{dual}$, then we get that:

$$cost_{answer} \le \gamma cost_{Dual} \le \gamma \mathrm{OPT}_{LP} \le \gamma \mathrm{OPT},$$

where OPT is the optimum solution to the facility location problem. Therefore this algorithm is a γ-approximation algorithm for the facility location problem. In this thesis, we use both LP rounding and the primal-dual approach in order to design approximation algorithms.

1.3 Overview of Our Contribution

We consider various clustering problems similar to the facility location problem. In Chapter 2, we focus on the dynamic facility location problem – a generalization of the facility location problem. The facility location problem does not consider a dynamic environment, that is, a setting in which the distances between clients and facilities can change with time. Motivated by diverse networks such as web links, nation- or world-wide social networks, online social networks (Facebook or Twitter for example), we study the dynamic facility location problem. This enables us to model and understand such evolving networks better.

In dynamic facility location, we are given a set of facilities F, clients C, and a temporally changing metric on them. We denote by $d_t(i, j)$ the metric distance between client j and facility i at time t. We are also given a switching cost g, the total number of time steps T, and an opening cost f_i for each facility i. The goal is to output, for each time step t, a subset of open facilities A_t and an assignment $\phi_t : C \to A_t$ of clients to facilities so as to minimize:

$$\sum_{1\le t\le T, i\in A_t} f_i + \sum_{1\le t\le T, j\in C} d_t(\phi_t(j), j) + \sum_{1\le t<T, j\in C} \mathbb{1}\{\phi_t(j) \ne \phi_{t+1}(j)\} \cdot g, \qquad (1.3.1)$$

where $\mathbb{1}\{p\}$ is the indicator function of a proposition p, i.e., it takes value 1 if p is true and 0 otherwise. Informally speaking, in this problem we have T time steps and in each

time step we are given an instance of the facility location problem. The goals are to find a good solution to each of these instances and also – for the sake of stability – to make sure that the number of clients that switch from the facility that is serving them to another facility is small. Therefore, the cost function is the summation of the opening cost and the connection cost for all those time steps, plus the switching cost of the clients.

Eisenstat et al. [27] were the first to consider this problem. They designed an LP relaxation for this problem and presented a $O(\log nT)$-approximation algorithm by rounding the primal solution. We also solve the same LP relaxation for this problem and propose a novel primal rounding scheme by using a technique called exponential clocks. This enables us to get a 14-approximation algorithm.

In Chapter 3 we consider the k-means and the k-median problems. The k-means problem has been well-studied theoretically, experimentally and practically [36]. In this problem, given a set $\mathcal{D}$ of n points in $\mathbb{R}^\ell$ and an integer k, the task is to select a set S of k *cluster centers* in $\mathbb{R}^\ell$, so that $\sum_{j\in\mathcal{D}} c(j,S)$ is minimized, where $c(j,S)$ is the squared Euclidean distance between j and its nearest center in S. We show that for any $\epsilon > 0$, there is a $(\rho_{\mathsf{mean}} + \epsilon)$-approximation algorithm for the k-means problem, where $\rho_{\mathsf{mean}} \approx 6.357$. Moreover, the integrality gap of the standard LP is at most ρ_{mean}. This improves over the best known approximation algorithm for the k-means problem that was a $(9+\epsilon)$-approximation algorithm based on local search, due to Kanungo et al. [41].

We further extend our result to two other problems. In the first extension, we consider a variant of the k-means problem in which each $c(j,S)$ corresponds to the squared distance in an arbitrary (possibly non-Euclidean) metric on $\mathcal{D}\cup\mathcal{F}$. For this problem, the best-known approximation algorithm was a 16-approximation, thanks to Gupta and Tangwongsan [33]. We improve over this result and show that for any $\epsilon > 0$, there is a $(9+\epsilon)$-approximation algorithm for the k-means problem in general metrics. Moreover, the integrality gap of the standard LP is at most 9.

In the second extension, we consider the Euclidean k-median problem. Here we are given a set $\mathcal{D}$ of n points in $\mathbb{R}^\ell$ corresponding to clients and a set $\mathcal{F}$ of m points in $\mathbb{R}^\ell$ corresponding to facilities. The task is to select a set S of at most k facilities from $\mathcal{F}$ so as to minimize $\sum_{j\in\mathcal{D}} c(j,S)$, where $c(j,S)$ is now the (non-squared) Euclidean distance from j to its nearest facility in S. For this problem, no approximation better than the general 2.675-approximation algorithm of Byrka et al. [21] was known. We show that for any $\epsilon > 0$, there is a $(\rho_{\mathsf{med}} + \epsilon)$-approximation algorithm for the Euclidean k-median problem, where $\rho_{\mathsf{med}} \approx 2.633$. Moreover, the integrality gap of the standard LP is at most ρ_{med}.

Our approaches for both k-means and k-median are similar. They consist of two main steps:

1. In the first stage, we design a novel primal-dual algorithm that produces a series of solutions with some interesting and useful properties; even though each of them might open a different number of facilities – not exactly k.

2. In the second stage, we carefully inspect these series of solutions and find a solution that opens exactly k facilities by losing a factor $1+\epsilon$ in the approximation guarantee.

Afterwards, in Chapter 4, we focus on designing fast, practical and provably good algorithms for the k-means problem. We modify our algorithm and improve its run time, so that it runs in time $O(nkd) + O(d \cdot \text{poly}(k))$ and show that it outperforms one of the best known algorithms for this problem (K-Means++) in accuracy with comparable or even slightly better running time.

2 The Dynamic Facility Location Problem

This chapter is based on a joint work with Hyung-Chan An and Ola Svensson, presented in the the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) 2015 [3]; also invited to a special issue of ACM Transactions on Algorithms [4].

2.1 Introduction

In this chapter, we focus on the (dynamic) facility location problem. As discussed before the facility location problem is an extensively studied combinatorial optimization problem, which can also be understood as a problem of identifying closely related groups of nodes in networks.

Let us first recall the definition of the facility location problem and dynamic facility location problem. In the first problem, we are given a single metric on a set of *clients* and *facilities*, where each facility is associated with an *opening cost*; the aim of the problem is to choose a subset of facilities to *open* and connect every client to one of these open facilities and minimize the solution cost. The solution cost is defined as the sum of the opening costs of the chosen facilities and the *connection cost* given by the total distance between every client and the facility it is connected to.

The dynamic facility location problem is a generalization of this classic problem to temporally evolving metrics, proposed by Eisenstat, Mathieu, and Schabanel [27]. The temporal aspect of the problem is modeled by T metrics given on the same set of clients and facilities, each representing the metric at time step $t \in \{1, \ldots, T\}$. The problem asks us to find a feasible connection of the clients for each time step, but minimizing a new objective function: in addition to the classic opening and connection costs, we incur a fixed amount of *switching cost* every time a client changes its connected facility between two consecutive time steps. This modification was introduced to favor "stable" solutions, as Eisenstat et al. proposed this problem in order to study the dynamics of evolving systems. Given a temporally changing social/transportation network, the

dynamic facility location problem is to discover a temporal evolution of groups that is not too sensitive to transient changes in the metric. A more comprehensive discussion on the study of the dynamics of evolving networks can be found later in this section. (Also see Eisenstat et al. [27].)

For the classic facility location problem, a large number of algorithmic techniques (and their combination) have been successfully applied, including LP-rounding [56, 24, 20, 45], filtering [47], primal-dual methods [40], dual fitting [38, 37], local search [11, 23], and greedy improvement [23]. LP-rounding approaches, in particular, have their merit that they easily extend to other related problems with similar relaxations. Interestingly, a common algorithmic tool is shared by these traditional LP-roundings: In rounding an LP solution, we cannot afford to open enough number of facilities to ensure that every client can find an open facility among the ones it is fractionally connected to in the LP solution. Hence, these LP-rounding algorithms define a *short* "fall-back" path for each client, and guarantee that at least this fall-back path will always lead to an open facility, even if the (randomized) rounding fails to give an open facility in the direct neighborhood. The fall-back paths are constructed based on a certain clustering of the LP-solution, where the algorithm opens at least one facility in each cluster. These clustering decisions, unfortunately, are very *sensitive* to small changes in the input. As a result, when these traditional LP-rounding techniques are applied to the dynamic problem, they can generate an excessive number of switches between two consecutive time steps whose LP connection variables are only slightly different.

In this chapter, we present a novel LP-rounding approach based on exponential clocks for facility location problems. Exponential clocks were previously used to give a new approximation algorithm for the multiway cut problem by Buchbinder, Naor, and Schwartz [19]. Several interesting properties distinguish our algorithm from previous LP-rounding approaches for facility location problems. First, our algorithm enables a client to be connected along an *arbitrarily long path* in the LP support, in contrast to the traditional fall-back paths. Although, at a glance, it might appear counter-intuitive that enabling longer paths helps, the use of exponential clock guarantees that the probability that a long path is actually used rapidly diminishes to zero as we consider longer paths. On the other hand, this small probability of using long paths is still sufficient to eliminate the need of fall-back paths. Thus leading to the second property our algorithm: it does not rely on any clustering. Our algorithm, consequently, becomes "stable" with respect to small changes in the LP solution. For the dynamic problem, *separately* applying our new LP-rounding for each time step, but with *shared* randomness, ensures that our algorithm makes similar connection decisions for any two time steps whose LP connection variables are similar. Our approach thereby yields the first constant approximation algorithm for the dynamic facility location problem. We also note that our algorithm is a Lagrangian-preserving constant approximation algorithm for the (classic) facility location problem, although with a worse approximation ratio than the smallest known.

Eisenstat et al. [27] propose $O(\log nT)$-approximation algorithms for the dynamic problem that avoid the stability issue in a different way: They connect every client directly to one of the randomly opened facilities to which the client is fractionally connected in the LP solution, where the random choices are made based on exponential distributions. Such a direct connection keeps the algorithm from relying on the triangle inequality, yet any algorithm that does not assume the triangle inequality cannot achieve a sublogarithmic approximation under complexity-theoretic assumptions. Eisenstat et al. in fact consider two versions of the dynamic facility location problem for both of which they presented logarithimic approximation algorithms: in one version, the facility opening decision is global – paying the opening cost makes the facility available at *every* time step. In the other version, considered in here, more flexibility is given to the facility opening decision: a facility is opened for a specific set of time steps, and *hourly* opening cost is paid for each time step. They show that the first version does not admit a $o(\log T)$-approximation algorithm even for the metric case, and they leave the open question about whether a constant approximation algorithm is possible for the second case, which we now answer positively.

Related work. A huge amount of data is gathered by observing social networks such as face-to-face contact in a primary school [57], where these networks evolve over time. Different tools have been suggested and analyzed in order to understand the dynamic structure of these networks [52, 58, 54]. The dynamic facility location problem is a new tool for analyzing temporal aspects of such networks, introduced by Eisenstat et al. [27]. In this chapter, we present a constant approximation algorithm for this problem.

Apart from the offline and the dynamic versions of the facility location problem discussed so far, the online setting is a well studied one (see [31] for a survey). In this setting, the clients arrive one at a time and we need to connect them to facilities on the fly. The study of this online setting was started by Meyerson [50], who achieved a competitive ratio of $O(\log n)$. Later, Fotakis [30] showed an asymptotically tight competitive ratio of $\Theta(\log n/\log\log n)$. Online problems have also been studied under varying assumptions to give constant competitive algorithms: Anagnostopoulos, Bent, Upfal, and Hentenryck [5] studied the case where the clients are drawn from a known distribution; Fotakis [29] presented an algorithm for the case where the reassignment of clients is allowed; Divéki and Imreh [26] considered a setting that allows us to move facilities.

Finally, the facility leasing problem is a variant of the facility location problem introduced by Anthony and Gupta [6], who considered a family of leasing problems. While the facility leasing problem also aims at connecting clients to open facilities over multiple time steps, there exist major differences from the dynamic facility location problem, such as the existence of switching costs. Nagarajan and Williamson [51] present a 3-approximation algorithm for the facility leasing problem.

Overview of our approach. The standard LP relaxation for the classic problem consists of two types of decision variables: *opening variables* indicating whether each facility is to be opened, and *connection variables* that indicate whether each pair of client and facility is to be connected. Our algorithm for the dynamic problem starts by solving the natural extension of the standard LP [27], which augments the LP with a new set of *switching variables* that reflect the ℓ_1-distances between the connection variables of consecutive time steps. Once we obtain an optimal solution, we apply the preprocessing of Eisenstat et al. to ensure that the connection variables of each client does not change too often compared to the switching cost paid: each time a client changes its (fractional) connection variables, at least one half of the switching cost is paid by the LP solution.

The second step of our algorithm is then to install competing exponential clocks, i.e., to sample a value from an exponential distribution, on every client and facility. These exponential clocks are said *competing*, as the random choices made by our algorithm are based on comparing the clocks on subsets of nodes and choosing the best (i.e., smallest-valued) one. After sampling these clocks, our algorithm considers the LP solution of each time step *separately* to construct an assignment for that time step, but based on a single set of exponential clocks shared across all time steps.

At each time step, in order to determine which facility is to be connected to a given client j, our algorithm constructs a path called *connection path*. The path starts from j, and iteratively proceeds to the smallest-clock node among the neighborhood of the current node. If this "smallest-following" path enters a cycle, we stop and connect j to the last facility seen. Under this random process of connecting j, the path may become very *long*, which is one of the unusual characteristics of our algorithm discussed earlier. However, observe from the construction that the sequence of clock values that this smallest-following path witnesses keep decreasing: in other words, in order for this path to grow, it has to see in the neighborhood a clock that beats everything seen so far. As such, as the path becomes longer, the probability that the path continues will rapidly diminish (Lemma 2.4.4). For any given edge e, this key observation implies that most of the paths that start from a distant node will fail to reach e; in fact, we show via a counting argument that the expected number of connection paths passing through e is within a constant factor of its connection variable (Corollary 2.4.6). This bounds the connection cost (and opening cost, with some additional arguments: see Lemma 2.4.3) within a constant factor of the LP cost.

Finally, in order to bound the switching cost, recall that the exponential clocks are shared by all time steps. Hence, if the LP solution did not change at all between two time steps, the random construction of connection paths would not change, either. Now, if the LP solution did change slightly, for example around a single client, an obstacle in the analysis would be that the change in the single client's connection may lead to the switches of multiple clients' connections, whereas the LP solution only pays a constant fraction (namely 1/2) of a single switch cost. Recall that, however, the connection

paths tend to be short; thus, a local change in the connection variable cannot affect the connection of a distant client with high probability, and indeed we show that a change in a single client's connection globally causes only constant number of switches in expectation (Lemma 2.4.7). This yields the last piece of analysis to establish that our algorithm is a constant approximation algorithm.

Our new LP-rounding approach for facility location problems raises several interesting research directions. In general, we envision that a further understanding of the techniques using exponential clocks will be fruitful for these problems. A more specific question is how far the approximation guarantee of our current analysis can be pushed. We know that, by incorporating more case analyses, the guarantee on the connection costs can be improved. However, it remains an interesting open problem to understand if a different analysis can lead to bounds that compete with the best known ratios for the classic facility location problem. Further, while the use of connection paths is important for the stability of the solution, a potential improvement of our algorithm when applied to the classic facility location problem is in connecting each client to the closest opened facility instead of always following the connection path.

2.2 Preliminaries

2.2.1 Facility location in evolving metrics

In this section, we recall the dynamic facility location problem's formal definition and adjust the LP relaxation presented before, to model this problem.

Problem definition. In dynamic facility location, we are given a set of facilities F, clients C, and a temporally changing metric on them. We denote by $d_t(i, j)$ the metric distance between client j and facility i at time t. We are also given a switching cost g, the total number of time steps T, and an opening cost f_i for each facility i. The goal is to output, for each time step t, a subset of open facilities A_t and an assignment $\phi_t : C \to A_t$ of clients to facilities so as to minimize:

$$\sum_{1 \le t \le T, i \in A_t} f_i + \sum_{1 \le t \le T, j \in C} d_t(\phi_t(j), j) + \sum_{1 \le t < T, j \in C} \mathbb{1}\{\phi_t(j) \neq \phi_{t+1}(j)\} \cdot g, \tag{2.2.1}$$

where $\mathbb{1}\{p\}$ is the indicator function of proposition p, i.e., it takes value 1 if p is true and 0 otherwise. In words, the objective function consists of the hourly opening costs for each open facility, the connection costs of each client, and the switching costs.

Linear programming relaxation. We first introduce the standard linear programming relaxation for the classic facility location problem (or, equivalently, the dynamic version with a single time step). We then formulate the relaxation for the dynamic facility location problem, introduced in [27], which is a natural generalization of the relaxation for the classic facility location problem.

In the standard LP-relaxation of the classic facility location problem, we have a variable y_i for each facility $i \in F$ and a variable x_{ij} for each facility $i \in F$ and client $j \in C$. The intuition of these variables is that y_i should take value 1 if i is opened and 0 otherwise; x_{ij} should take value 1 if client j is connected to facility i and 0 otherwise. The set of feasible solutions to the relaxation is now described by $\mathbb{P}_{\mathsf{FL}} = \{(x, y) \mid \sum_{i \in F} x_{ij} = 1,\ \forall j \in C;\ x_{ij} \leq y_i,\ \forall i \in F, j \in C;\ \text{and } x, y \geq \mathbf{0}\}$. The first set of inequalities says that each client should be connected to a facility and the second set says that if a client j is connected to a facility i then that facility should be opened. In this terminology, the standard LP relaxation of the classic facility location problem is the following:

$$\begin{aligned} \text{minimize } & \sum_{i \in F} y_i f_i + \sum_{i \in F, j \in C} x_{ij} d(i, j) \\ \text{subject to } & (x, y) \in \mathbb{P}_{\mathsf{FL}}. \end{aligned}$$

We now adapt the above relaxation to the dynamic facility location problem. Let $[T] = \{1, \ldots, T\}$ and $[T) = \{1, \ldots, T-1\}$. For each time step $t \in [T]$, the relaxation has a variable y_i^t for each facility $i \in F$ and a variable x_{ij}^t for each facility $i \in F$ and client $j \in C$. These variables indicate which facilities should be opened at time t and where to connect clients at this time step. In other words, (x^t, y^t) should be a solution to the classic facility location problem and our relaxation will constrain that $(x^t, y^t) \in \mathbb{P}_{\mathsf{FL}}$ for each $t \in [T]$. To take into account the switching costs, our relaxation will also have a non-negative variable z_{ij}^t for each client $j \in C$, facility $i \in F$ and time $t \in [T]$. The intuition of z_{ij}^t is that it should take value 1 if client j was connected to facility i at time t but not at time $t+1$. The relaxation of the dynamic facility location problem introduced in [27] is

$$\begin{aligned} \text{minimize } & \sum_{i \in F, t \in [T]} y_i^t f_i + \sum_{i \in F, j \in C, t \in [T]} x_{ij}^t d_t(i, j) + \sum_{i \in F, j \in C, t \in [T)} z_{ij}^t \cdot g \\ \text{subject to } & (x^t, y^t) \in \mathbb{P}_{\mathsf{FL}} && \forall t \in [T], \\ & z_{ij}^t \geq x_{ij}^t - x_{ij}^{t+1} \text{ and } z_{ij}^t \geq 0, && \forall i \in F, j \in C, t \in [T). \end{aligned}$$

2.2.2 Exponential clocks

We refer to independent exponential random variables as exponential clocks. The probability density function of an exponential distribution with rate parameter $\lambda > 0$ is $f_\lambda(x) = \lambda e^{-\lambda x}$ for $x \geq 0$. If a random variable X has this distribution, we write $X \sim \text{Exp}(\lambda)$. We use the following well-known properties of the exponential distribution:

1. If $X \sim \text{Exp}(\lambda)$, then $\frac{X}{c} \sim \text{Exp}(\lambda c)$ for any $c > 0$.
2. Let $X_1, X_2, ..., X_n$ be independent exponential clocks with rate parameters $\lambda_1, \lambda_2, ..., \lambda_n$, then
   (a) $\min\{X_1, ..., X_n\} \sim \text{Exp}(\lambda_1 + ... + \lambda_n)$.
   (b) $\Pr[X_i = \min\{X_1, ..., X_n\}] = \frac{\lambda_i}{\lambda_1 + ... + \lambda_n}$.
3. Exponential clocks are memoryless, that is, if $X \sim \text{Exp}(\lambda)$, for any $n, m > 0$:

$$\Pr(X > m + n | X > m) = \Pr(X > n).$$

Note that the memorylessness property implies that if we have a set of exponential clocks then after observing the minimum of value say v, the remaning clocks, subtracted by v, are still exponentially distributed with their original rates.

2.2.3 Preprocessing

The first preprocessing is from the $O(\log nT)$-approximation algorithm by Eisenstat et al. [27]. Losing a factor of 2 in the cost, this simple preprocessing lets us assume that the LP pays one switching cost each time a client changes its fractional connection.

Lemma 2.2.1 ([27])**.** *Given an LP solution, we can, by increasing its cost by at most a factor of* 2*, obtain in polynomial time a feasible solution* (x, y, z) *satisfying:*

- *If we let* $Z^t = \{j \in C \mid x^t_{ij} \neq x^{t+1}_{ij}$ *for some* $i \in F\}$ *denote the set of clients that changed its fractional connection between time step* t *and* $t + 1$*, then* $\sum_{t=1}^{T-1} |Z^t| \leq \sum_{t=1}^{T-1} \sum_{i \in F, j \in C} z^t_{ij}$.

The second preprocessing is obtained by using the standard trick of duplicating facilities, while being careful that if the connection variables of a client remain the same between two consecutive time steps, they remain so even after the preprocessing.

Observation 2.2.2. *Without loss of generality, we assume that* (x, y, z) *satisfies the following:*

1. *For any facility* $i \in F$*, client* $j \in C$*, and time step* $t \in [T]$*,* $x^t_{ij} \in \{0, y^t_i\}$.

2. *For each facility $i \in F$, there exists $o_i \in [0,1]$ such that $y_i^t \in \{0, o_i\}$ for each time step $t \in [T]$.*

For formal proofs of the above statements, we refer the reader to section 2.5.

2.3 Description of Our Algorithm

Given a preprocessed solution (x, y, z) to the linear programming relaxation that satisfies the properties of Lemma 2.2.1 and Observation 2.2.2, our algorithm proceeds by first making a random choice and then opening facilities and connecting clients in each time step.

Random choice: Sample independently an exponential clock $Q_i \sim \text{Exp}(o_i)$ for each facility $i \in F$ and an exponential clock $R_j \sim \text{Exp}(1)$ for each client $j \in C$.

Opening and connecting: At each time step $t \in [T]$, open facilities and connect clients as follows. Consider the clients in the non-decreasing order of their sampled clocks (R_j's). When client $j \in C$ is considered, find the facility $i = \arg\min_{i:x_{ij}^t>0} Q_i$ of the smallest clock among the facilities that j is connected to in the support of x^t. Similarly, let $j' = \arg\min_{j':x_{ij'}^t>0} R_{j'}$ be the client with the smallest clock in the neighborhood of i. The connection of j at time t is now determined as follows: if $j = j'$ then open i and connect j to i; otherwise, connect j to the same facility as j'.

Note that, with probability 1, all clocks will have distinct values so we make that simplifying assumption. We also remark that the procedure is well-defined. Indeed, if j connects to the same facility as j' then $R_{j'} < R_j$ (since both j and j' are adjacent to i) , and therefore j' was already connected to a facility when j was considered.

2.3.1 An alternative presentation of our algorithm

In this section, we rewrite the *opening and connecting* step of the algorithm by using graph terminology, simplifying the presentation of our analysis. The new *opening and connecting* step for time step t reads as follows.

Let $\mathcal{SG}(x^t)$ be the *support graph* of x^t: i.e., $\mathcal{SG}(x^t)$ is an undirected bipartite graph on vertex set $F^t \cup C$ that has an edge $\{i, j\}$ if and only if $x_{ij}^t > 0$, where $F^t := \{i \in F \mid \exists j \; x_{ij}^t > 0\}$. Then we construct a directed bipartite graph $\mathcal{CG}(x^t)$, called *connection graph*, which represents the random choices made by the algorithm. $\mathcal{CG}(x^t)$ is a directed graph on the same vertex set $F^t \cup C$, where every vertex has exactly one outgoing arc directed towards the vertex with the smallest clock among its neighborhood in $\mathcal{SG}(x^t)$.

Note that the underlying undirected graph of $\mathcal{CG}(x^t)$ is therefore a subgraph of $\mathcal{SG}(x^t)$. Now the algorithm finds all length-2 cycles in $\mathcal{CG}(x^t)$, and opens every facility that appears on any of these cycles.

Once the algorithm determines the set of facilities to be opened, it produces an assignment of the clients to the open facilities. For each client j, the algorithm defines its *connection path* $P_j(x^t)$ as follows: the path starts from j, and follows the unique outgoing arc in the connection graph until it stops just before it is about to visit a vertex that it has already visited. Note that this path is well-defined, since every vertex has exactly one outgoing arc in the connection graph and there are finitely many vertices in the graph. The algorithm assigns j to the facility that appears latest on $P_j(x^t)$: if $P_j(x^t)$ ends at a facility, j is assigned to that facility; if $P_j(x^t)$ ends at a client, the second-to-last vertex of the path is a facility, and j is assigned to that facility. Figure 2.1 illustrates an execution of our algorithm.

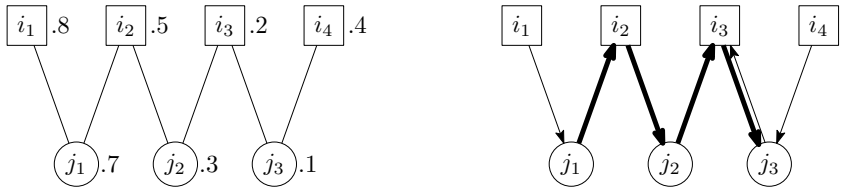


Figure 2.1 – An example of the execution of our algorithm. A support graph is shown on the left, where squares represent facilities and circles clients. Each node is annotated with its exponential clock value. The corresponding connection graph is shown on the right, where the bold edges represent $P_{j_1}(x^t)$. Our algorithm connects j_1 to i_3 in this case.

Following is a useful observation in the analysis of our algorithm.

Observation 2.3.1. *$\mathcal{CG}(x^t)$ does not contain a simple cycle with more than two arcs.*

Proof. Suppose that $\mathcal{CG}(x^t)$ contains a simple cycle $\langle i_1, j_1, i_2, \ldots, i_k, j_k(, i_1)\rangle$ for $k \geq 2$. Note that $i_1 \neq i_2$, $\{j_1, i_1\}, \{j_1, i_2\} \in \mathcal{SG}(x^t)$, and $(j_1, i_2) \in \mathcal{CG}(x^t)$; hence, $Q_{i_2} < Q_{i_1}$. Repeating this argument yields $Q_{i_1} < Q_{i_1}$, obtaining contradiction. □

Each connection path therefore ends only when it reaches a length-2 cycle, and this is why the new presentation of the algorithm is guaranteed to assign every client to an open facility. Observation 2.3.2 easily follows from the fact that the connection graph is merely a graph representation reflecting the random choices made by the original algorithm.

Observation 2.3.2. *The two versions of our algorithm are equivalent: they open the same set of facilities and produce the same assignment.*

Proof. First we verify that both versions of our algorithm open the same set of facilities. As the connection graph is bipartite, every length-2 cycle contains exactly one facility and one client. Let $\{(i,j),(j,i)\}$ be a length-2 cycle where $i \in F$ and $j \in C$; the new version opens i in this case. When the original version considers j's connection, it selects i in the neighborhood and subsequently finds that j has the smallest clock in the neighborhood of i. Hence, the original version also opens i.

Note that this in fact is the only case in which the original version opens a facility: $i' \in F$ is opened if and only if, for some client j', i' had the smallest clock in the neighborhood of j' and vice versa. This emerges as a length-2 cycle in the connection graph, and therefore the new version also opens i'.

Now we show by induction that, for all $k \in \mathbb{N}$, a client j whose connection path consists of k arcs is assigned to the same facility by both versions of our algorithm. Suppose $k = 1$. In this case, $P_j(x^t) = \{(j,i)\}$ and $(i,j) \in \mathcal{CG}(x^t)$. Thus, the original version opens i and assigns j to i, which is consistent with the decision made by the new version. Suppose $k = 2$. In this case, $P_j(x^t) = \{(j,i),(i,j')\}$ and $(j',i) \in \mathcal{CG}(x^t)$. The new version assigns j to i; the original version assigns j to the same facility as j', and assigns j' to facility i. Hence, the decisions are consistent in this case as well.

Suppose that k is an odd integer greater than two, and let $(j,i_1),(i_1,j_1)$ be the first two arcs of $P_j(x^t)$ and (j_2,i_2) be the last. Then the outgoing arc from i_2 in the connection graph has to be towards j_2, since it has to be towards a vertex that is already visited, creating a length-2 cycle. (See Observation 2.3.1.) Thus, we can obtain $P_{j_1}(x^t)$ by removing the first two arcs from $P_j(x^t)$, and the original version of the algorithm assigns j_1 to i_2 from the induction hypothesis. It assigns j also to i_2, which is consistent with the decision made by the new version of our algorithm.

The final case where k is even follows from a symmetric argument. □

2.4 Analysis

In this section, we analyze our algorithm described in Section 4.3 for the dynamic facility location problem. Throughout this section, let X_{ij}^t denote the random indicator variable that takes value 1 if the algorithm connects client j to facility i at time step t and let Y_i^t be the random indicator variable that takes value 1 if facility i is opened during time step t. All probabilities and expectations in this section are over the random outcomes of the exponential clocks. With this notation and by linearity of expectation, the expected

cost of the returned solution equals

$$\underbrace{\sum_{t=1}^{T} \mathbb{E}\left[\sum_{i\in F} f_i \cdot Y_i^t + \sum_{i\in F, j\in C} d_t(i,j) X_{ij}^t\right]}_{(i)} +$$
$$g \cdot \underbrace{\sum_{t=1}^{T-1} \mathbb{E}\left[\sum_{j\in C} \mathbb{1}\{X_{ij}^t \neq X_{ij}^{t+1} \text{ for some } i \in F\}\right]}_{(ii)},$$

where the term (i) expresses the expected opening cost and connection cost at time step t and the second term (ii) expresses the expected number of clients who changed, between time steps t and $t+1$, the facility to which they are connected. Note that analyzing (i) is simply the problem of analyzing our algorithm for the uncapacitated facility location problem. To analyze (ii) we crucially rely on the fact that the random choices of our algorithm (i.e., the sampling of the exponential clocks) are based on a single set of exponential clocks shared by all time steps. This will enable us to prove that the expected number of clients that change connection is proportional to the cost that the LP pays towards the switching cost.

More specifically, we prove the following lemma:

Lemma 2.4.1. *For any time step $t \in [T]$, we have*

$$\mathbb{E}\left[\sum_{i\in F} f_i Y_i^t\right] \leq \sum_{i\in F} f_i y_i^t, \tag{2.4.1}$$

$$\mathbb{E}\left[\sum_{i\in F, j\in C} d_t(i,j) X_{ij}^t\right] \leq 6 \sum_{i\in F, j\in C} d_t(i,j) x_{ij}^t, \tag{2.4.2}$$

$$g \cdot \mathbb{E}\left[\sum_{j\in C} \mathbb{1}\{X_{ij}^t \neq X_{ij}^{t+1} \textit{ for some } i \in F\}\right] \leq 7g|Z^t|. \tag{2.4.3}$$

The above lemma implies that our algorithm is a 14-approximation algorithm for the dynamic facility location, from the following argument. The preprocessing of Lemma 2.2.1 incurs a factor of 2. Combining this with the above lemma gives us that the opening costs are approximated within a factor of 2, the connection cost within a factor of 12, and the switching cost within a factor 14 since $\sum_{t\in[T)} |Z^t| \leq \sum_{i\in F, j\in C, t\in[T)} z_{ij}^t$. Hence, we have the following[1]:

Theorem 2.4.2. *There is a randomized* 14*-approximation algorithm for the dynamic facility location problem.*

[1] As can be seen from the argument, we could also have *temporally changing opening cost* without changing our algorithm: we can allow the input to specify f_i^t, the opening cost of facility i at time step t.

We prove Inequalities (2.4.1), (2.4.2), and (2.4.3) of Lemma 2.4.1 in Sections 2.4.1, 2.4.2, and 2.4.3, respectively.

2.4.1 Bounding the opening cost

We show that the probability to open a facility i at time step t equals y_i^t.

Lemma 2.4.3. *For any time step $t \in [T]$ and facility $i \in F$, $\mathbb{E}[Y_i^t] \leq y_i^t$.*

Proof. If $i \notin F^t$, it does not appear in $\mathcal{SG}(x^t)$ and cannot be opened: $\mathbb{E}[Y_i^t] = 0 \leq y_i^t$. Suppose $i \in F^t$, and let j be the facility to which i has its outgoing arc in the connection graph: i.e., $(i,j) \in \mathcal{CG}(x^t)$. Let $F(j)$ be the set of facilities that are adjacent to j in $\mathcal{SG}(x^t)$. Observe that i will be opened if and only if $(j,i) \in \mathcal{CG}(x^t)$, or, in other words, $Q_i = \min_{i' \in F(j)} Q_{i'}$. Note that this event is independent from the event $(i,j) \in \mathcal{CG}(x^t)$, since the facility-clocks are independent from the client-clocks. Thus, from Property 2b of exponential clocks, we have $\mathbb{E}[Y_i^t] = \frac{y_i^t}{\sum_{i' \in F(j)} y_{i'}^t} = y_i^t$, where the last equality follows from $1 = \sum_{i' \in F(j)} x_{i'j}^t = \sum_{i' \in F_j} y_{i'}^t$. (See Observation 2.2.2.)

□

2.4.2 Bounding the connection cost

In this section we bound the connection cost for a fixed time step $t \in [T]$, i.e., $\mathbb{E}\left[\sum_{i \in F, j \in C} d_t(i,j) X_{ij}^t\right]$. As the time step t is fixed throughout this section, we simplify the notation by letting $(x,y) = (x^t, y^t)$ and we also abbreviate $\mathcal{SG}(x^t), \mathcal{CG}(x^t)$, and $P_j(x^t)$ by $\mathcal{SG}, \mathcal{CG}$, and P_j, respectively.

By the triangle inequality, the connection cost of a client is at most the sum of distances of the edges in its connection path. Therefore we have that

$$\mathbb{E}\left[\sum_{i \in F, j \in C} d_t(i,j) X_{ij}^t\right] \leq \mathbb{E}\left[\sum_{j \in C} d_t(P_j)\right],$$

where $d_t(P_j)$ denotes the total distance of the edges in the path P_j. Note that the right-hand side can be further rewritten by summing over all the edges in $\mathcal{SG}$ and counting the expected number of connection paths that use this edge (note that the paths do not repeat a vertex). That is, we obtain the following bound on the connection cost

$$\sum_{\{i,j\} \in \mathcal{SG}} d_t(i,j) \mathbb{E}[|\{j' \in C \mid (i,j) \text{ or } (j,i) \text{ is in } P_{j'}\}|].$$

To analyze this, we first bound the probability, over the randomness of the exponential

clocks, that a connection path starts with a given prefix. We then show that the expected number of connection paths that traverses an edge $\{i, j\}$ is at most $6x_{ij}$, which then implies Inequality (2.4.2) of Lemma 2.4.1.

The probability of a prefix

Consider a client j_0 and its connection path P_{j_0}. We use the notation $\text{prefix}(P_{j_0})$ to denote the set of all the prefixes of this path, i.e., the subpaths of P_{j_0} that start at j_0. We also let $C(i)$ denote the set of clients adjacent to a facility i in $\mathcal{SG}$. Similarly, let $F(j)$ denote the set of facilities adjacent to client j in $\mathcal{SG}$. We further abbreviate $C(i_1) \cup \cdots \cup C(i_\ell)$ by $C(i_1, \ldots, i_\ell)$ and $F(j_1) \cup \cdots \cup F(j_\ell)$ by $F(j_1, \ldots, j_\ell)$. Finally, for a subset $F' \subseteq F$ of facilities, we let $y(F') = \sum_{i \in F'} y_i$. Using this notation, we now bound the probability that a given subpath appears as a prefix of a connection path.

Lemma 2.4.4. *We have*

$$\Pr[\langle j_0, i_1, j_1, i_2, \ldots, i_k, j_k, i_{k+1}\rangle \in \text{prefix}(P_{j_0})] \leq \prod_{\ell=1}^{k} \frac{1}{|C(i_1, i_2, \ldots, i_\ell)|} \cdot \prod_{\ell=0}^{k} \frac{y_{i_{\ell+1}}}{y(F(j_0, j_1, \ldots, j_\ell))},$$

$$\Pr[\langle j_0, i_1, j_1, i_2, \ldots, i_k, j_k\rangle \in \text{prefix}(P_{j_0})] \leq \prod_{\ell=1}^{k} \frac{1}{|C(i_1, i_2, \ldots, i_\ell)|} \cdot \prod_{\ell=0}^{k-1} \frac{y_{i_{\ell+1}}}{y(F(j_0, j_1, \ldots, j_\ell))}.$$

Proof. We start by analyzing $\Pr[\langle j_0, i_1, j_1, i_2, \ldots, i_k, j_k, i_{k+1}\rangle \in \text{prefix}(P_{j_0})]$. First note that $\langle j_0, i_1, j_1, i_2, \ldots, i_k, j_k, i_{k+1}\rangle \in \text{prefix}(P_{j_0})$ if and only if all the arcs (j_0, i_1), (i_1, j_1), $\ldots, (j_k, i_{k+1})$ exist in $\mathcal{CG}$. The algorithm uses two independent sources of randomness: the client-clocks and facility-clocks. We use the randomness of the client-clocks to bound the probability that all the arcs $(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)$ exist in $\mathcal{CG}$. Note that for (i_ℓ, j_ℓ) to exist, j_ℓ has to have the smallest clock of the clients in $C(i_\ell)$. Moreover, as $j_{\ell-1} \in C(i_\ell)$, we have $R_{j_\ell} < R_{j_{\ell-1}}$. By repeating this argument, we have that a necessary condition for all arcs to exist in $\mathcal{CG}$ is that $R_{j_\ell} < R_{j_{\ell-1}} < \ldots < R_{j_1}$, which implies that the arcs exist only if

$$R_{j_\ell} = \min\{R_j \mid j \in C(i_1, i_2, \ldots, i_\ell)\} \qquad \text{for } \ell = 1, \ldots, k.$$

To bound the probability that these conditions hold, we use the well-known properties of the exponential distribution. Note first that all client-clocks are distributed according to the exponential distribution with the same rate (which is 1) and therefore the probability

of $R_{j_k} = \min\{R_j \mid j \in C(i_1, i_2, \ldots, i_k)\}$ is

$$\frac{1}{|C(i_1, i_2, \ldots, i_k)|}. \tag{2.4.4}$$

Now, by the memorylessness property, if we condition on the event that j_k has the smallest exponential clock of all clients in $C(i_1, i_2, \ldots, i_k)$, then the clocks of the clients different from j_k, subtracted by R_{j_k}, are still distributed according to the exponential distribution with same rates. Therefore, the probability of $R_{j_{k-1}} = \min\{R_{k-1} \mid j \in C(i_1, i_2, \ldots, i_{k-1})\}$ is $1/|C(i_1, i_2, \ldots, i_{k-1})|$ even if we condition on the event $R_{j_k} = \min\{R_j \mid j \in C(i_1, i_2, \ldots, i_k)\}$.[2] Note that $j_k \notin C(i_1, \ldots, i_{k-1})$ and more generally $j_\ell \notin C(i_1, \ldots, i_{\ell-1})$ (otherwise, as $R_{j_\ell} < R_{j_{\ell-1}} < \ldots < R_{j_1}$, one of the facilities in $\{i_1, \ldots, i_{\ell-1}\}$ would have its outgoing arc to j_ℓ or another client of smaller clock). By repeating this argument, we get that the probability that all the arcs $(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)$ exist in $\mathcal{CG}$ is at most

$$\prod_{\ell=1}^{k} \frac{1}{|C(i_1, i_2, \ldots, i_\ell)|}.$$

We then use the randomness of the facility-clocks to bound the probability that all the arcs (j_0, i_1), (j_1, i_2), $\ldots$, (j_k, i_{k+1}) exist. Similarly to above, we have that these arcs exist only if

$$Q_{i_{\ell+1}} = \min\{Q_i \mid i \in F(j_0, j_1, \ldots, j_\ell)\} \qquad \text{for } \ell = 0, \ldots, k.$$

As the clock Q_i of a facility is distributed according to the rate $o_i = y_i$, we have that $\Pr[Q_{i_{\ell+1}} = \min\{Q_i \mid i \in F(j_0, j_1, \ldots, j_\ell)\}$ equals

$$\frac{y_{i_{\ell+1}}}{y(F(j_0, j_1, \ldots, j_\ell))}$$

and again by using the memorylessness property, all these arcs exist with probability at most

$$\frac{y_{i_1}}{y(F(j_0))} \cdot \frac{y_{i_2}}{y(F(j_0, j_1))} \cdots \frac{y_{i_{k+1}}}{y(F(j_0, j_1, \ldots, j_k))}.$$

The bound now follows since the client-clocks and facility-clocks are independent.

Let us now calculate an upper bound on $\Pr[\langle j_0, i_1, j_1, i_2, \ldots, i_k, j_k\rangle \in \text{prefix}(P_{j_0})]$. Similarly to above we have that the probability that all the arcs $(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)$ exist in $\mathcal{CG}$ is at most (2.4.4). We now bound the probability that all the arcs $(j_0, i_1), (j_1, i_2), \ldots, (j_{k-1}, i_k)$ exist in $\mathcal{CG}$. By using analogous arguments to above, this is

[2] Since the exponential clocks have all the same rate, an equivalent (more combinatorial) point of view is the following: choose a random permutation of the clients in $C(i_1, i_2, \ldots, i_k)$. The probability that a certain client is the first (smallest clock) is 1 over the cardinality of the set; and even after conditioning on this event all permutations of the remaining clients are equally likely.

at most

$$\frac{y_{i_1}}{y(F(j_0))} \cdot \frac{y_{i_2}}{y(F(j_0, j_1))} \cdots \frac{y_{i_k}}{y(F(j_0, j_1, \ldots, j_{k-1}))}$$

and the statement again follows from the independence of the facility-clocks and client-clocks.

□

Expected number of connection paths traversing an edge

We now bound the expected number of connection paths that traverse an edge in the support graph. When we say that a path visits k clients before going through arc (j, i) we mean that it visits k clients different from j before going through the arc.

Lemma 2.4.5. *Consider a facility $i \in F$ and a client $j \in C$. For any integer $k \geq 1$, the expected number of connection paths that visits k clients before going through arc (i, j) is at most*

$$\frac{x_{ij}}{2^{\max(0,k-2)}}$$

and, for any integer $k \geq 0$, the expected number of connection paths that visits k clients before going through the arc (j, i) is at most

$$\frac{x_{ij}}{2^{\max(0,k-1)}}.$$

Proof. We divide the proof into the following cases: $k = 0, k = 1$, and $k \geq 2$.

Case $k = 0$. In this case there is no path that visits 0 clients before going through the arc (i, j). The expected number of paths that visits 0 clients before going through the arc (j, i) is equal to the probability of that $\langle j, i \rangle \in \text{prefix}(P_j)$, which by Lemma 2.4.4 is at most $y_i / y(F(j)) = y_i = x_{ij}$.

Case $k = 1$. Note that any prefix of a connection path that visits 1 client before going through the arc (i, j) must be of the form $\langle j_0, i_1, j_1 \rangle$ where $i_1 = i, j_1 = j$ and $j_0 \in C(i_1)$. Hence, by linearity of expectation, we have that the expected number of such paths is at most

$$\sum_{j_0 \in C(i_1)} \Pr[\langle j_0, i_1, j_1 \rangle \in \text{prefix}(P_{j_0})] \leq \sum_{j_0 \in C(i_1)} \frac{1}{|C(i_1)|} \frac{y_{i_1}}{y(F(j_0))} = y_{i_1} = x_{i_1 j_1},$$

where the inequality follows from Lemma 2.4.4 and the equalities follow from $y(F(j_0)) = \sum_{i \in F} x_{ij_0} = 1$ and $x_{i_1 j_1} = y_{i_1}$ since $i_1 \in F(j_1)$.

Let us now consider the expected number of connection paths whose prefix visits 1 client before going through the arc (j, i). If we let $j_1 = j$ and $i_2 = i$, any such prefix has the form $\langle j_0, i_1, j_1, i_2\rangle$ where $i_1 \in F(j_1)$ and $j_0 \in C(i_1)$. Hence, again by linearity of expectation and by Lemma 2.4.4, we have the upper bound of

$$\begin{aligned}\sum_{i_1 \in F(j_1)} \sum_{j_0 \in C(i_1)} &\Pr[\langle j_0, i_1, j_1, i_2\rangle \in \text{prefix}(P_{j_0})] \\ &\leq \sum_{i_1 \in F(j_1)} \sum_{j_0 \in C(i_1)} \frac{1}{|C(i_1)|} \frac{y_{i_1}}{y(F(j_0))} \frac{y_{i_2}}{y(F(j_0, j_1))} \\ &\leq y_{i_2} \sum_{i_1 \in F(j_1)} \sum_{j_0 \in C(i_1)} \frac{1}{|C(i_1)|} \frac{y_{i_1}}{y(F(j_1))} = y_{i_2} = x_{i_2 j_1},\end{aligned}$$

where the last inequality follows from $y(F(j_0, j_1)) \geq y(F(j_1))$ and $y(F(j_0)) = 1$.

Case $k \geq 2$: We start by analyzing the expected number of connection paths that have prefixes that visit k clients before going through arc (i, j). For notational convenience let $i_k = i$ and $j_k = j$. Any such prefix with nonzero probability has the form

$$\langle j_0, i_1, j_1, i_2, \ldots, j_{k-1}, i_k, j_k\rangle,$$

where $j_\ell \in C(i_{\ell+1})$ for $\ell = 0, 1, \ldots, k-1$ and $i_\ell \in F(j_\ell)$ for $\ell = 1, \ldots, k$. Moreover, we have that $i_\ell \notin F(j_{\ell+1})$, i.e., $i_\ell \in F(j_\ell) \setminus F(j_{\ell+1})$. Indeed, as the client-clocks decrease along a path, if $i_\ell \in F(j_{\ell+1})$ then i_ℓ would have its outgoing arc to $j_{\ell+1}$ (or a client of even smaller clock) instead of j_ℓ. By linearity of expectation, we can thus upper bound the expected number of such prefixes by the following sum:

$$\sum_{j_{k-1} \in C(i_k)} \sum_{\substack{i_{k-1} \in F(j_{k-1}) \setminus F(j_k) \\ j_{k-2} \in C(i_{k-1})}} \cdots \sum_{\substack{i_1 \in F(j_1) \setminus F(j_2) \\ j_0 \in C(i_1)}} \Pr[\langle j_0, i_1, j_1, i_2, \ldots, j_{k-1}, i_k, j_k\rangle \in \text{prefix}(P_{j_0})].$$

By Lemma 2.4.4 we have that $\Pr[\langle j_0, i_1, j_1, i_2, j_2, \ldots, j_{k-1}, i_k, j_k\rangle \in \text{prefix}(P_{j_0})]$ is at most

$$\begin{aligned}&\prod_{\ell=1}^{k} \frac{1}{|C(i_1, i_2, \ldots, i_\ell)|} \cdot \prod_{\ell=0}^{k-1} \frac{y_{i_{\ell+1}}}{y(F(j_0, j_1, \ldots, j_\ell))} \\ &\leq \prod_{\ell=1}^{k} \frac{1}{|C(i_\ell)|} \cdot \frac{y_{i_k}}{y(F(j_0))} \cdot \prod_{\ell=0}^{k-2} \frac{y_{i_{\ell+1}}}{y(F(j_\ell, j_{\ell+1}))} \\ &\leq \frac{y_{i_k}}{|C(i_k)| y(F(j_0))} \frac{y_{i_{k-1}}}{|C(i_{k-1})| y(F(j_0, j_1))} \prod_{\ell=1}^{k-2} \frac{y_{i_\ell}}{|C(i_\ell)| y(F(j_\ell, j_{\ell+1}))} \\ &\leq \frac{y_{i_k}}{|C(i_k)|} \frac{y_{i_{k-1}}}{|C(i_{k-1})|} \prod_{\ell=1}^{k-2} \frac{y_{i_\ell}}{|C(i_\ell)| y(F(j_\ell, j_{\ell+1}))}.\end{aligned}$$

Substituting in this bound, we get that the expected number of connection paths is at most

$$\sum_{j_{k-1}\in C(i_k)} \sum_{\substack{i_{k-1}\in F(j_{k-1})\setminus F(j_k)\\ j_{k-2}\in C(i_{k-1})}} \cdots \sum_{\substack{i_1\in F(j_1)\setminus F(j_2)\\ j_0\in C(i_1)}} \frac{y_{i_k}}{|C(i_k)|}\frac{y_{i_{k-1}}}{|C(i_{k-1})|}\prod_{\ell=1}^{k-2}\frac{y_{i_\ell}}{|C(i_\ell)|y(F(j_\ell,j_{\ell+1}))},$$

which, by rearranging terms, equals

$$\sum_{j_{k-1}\in C(i_k)} \frac{y_{i_k}}{|C(i_k)|}\left(\sum_{\substack{i_{k-1}\in F(j_{k-1})\setminus F(j_k)\\ j_{k-2}\in C(i_{k-1})}} \frac{y_{i_{k-1}}}{|C(i_{k-1})|}\cdots\left(\sum_{\substack{i_1\in F(j_1)\setminus F(j_2)\\ j_0\in C(i_1)}}\frac{y_{i_1}}{|C(i_1)|y(F(j_1,j_2))}\right)\right).$$

To analyze this expression, let us first consider the last term

$$\sum_{i_1\in F(j_1)\setminus F(j_2)}\sum_{j_0\in C(i_1)}\frac{1}{|C(i_1)|}\frac{y_{i_1}}{y(F(j_1,j_2))} = \sum_{i_1\in F(j_1)\setminus F(j_2)}\frac{y_{i_1}}{y(F(j_1,j_2))} = \frac{y(F(j_1)\setminus F(j_2))}{y(F(j_1,j_2))}.$$

Recall that $F(j_1)$ and $F(j_2)$ are two sets such that $y(F(j_1)) = y(F(j_2)) = 1$. Let $s = y(F(j_1)\cap F(j_2))$. Then

$$\frac{y(F(j_1)\setminus F(j_2))}{y(F(j_1,j_2))} = \frac{1-s}{2-s} \le \frac{1}{2}.$$

In general, we have for $\ell = 1,\dots k-2$ that the ℓ-th last term is bounded by

$$\frac{y(F(j_\ell)\setminus F(j_{\ell+1}))}{y(F(j_\ell,j_{\ell+1}))} \le \frac{1}{2}.$$

Thus, repeating the same arguments for the $k-2$ last terms enable us to upper bound the expected number of paths by

$$\frac{1}{2^{\max(0,k-2)}}\sum_{j_{k-1}\in C(i_k)}\frac{y_{i_k}}{|C(i_k)|}\left(\sum_{\substack{i_{k-1}\in F(j_{k-1})\setminus F(j_k)\\ j_{k-2}\in C(i_{k-1})}}\frac{y_{i_{k-1}}}{|C(i_{k-1})|}\right)$$

and using that $y(F(j_{k-1})\setminus F(j_k)) \le 1$ this is at most $y_{i_k}\frac{1}{2^{\max(0,k-2)}} = x_{i_k j_k}\frac{1}{2^{\max(0,k-2)}}$ as required.

Let us now bound the expected number of connections paths whose prefix visits k clients before going through the arc (j,i). For notational convenience, let now $j_k = j$ and $i_{k+1} = i$. By the same arguments as above, any such prefix with nonzero probability has the form $\langle j_0, i_1, j_1, i_2, \dots, j_{k-1}, i_k, j_k, i_{k+1}\rangle$, where $j_\ell \in C(i_{\ell+1})$ for $\ell = 0,1,\dots,k$ and $i_\ell \in F(j_\ell)\setminus F(j_{\ell+1})$ for $\ell = 1,\dots,k-1$. By linearity of expectation, we can thus bound

the expected number of such prefixes by

$$\sum_{\substack{i_k \in F(j_k) \\ j_{k-1} \in C(i_k)}} \sum_{\substack{i_{k-1} \in F(j_{k-1}) \setminus F(j_k) \\ j_{k-2} \in C(i_{k-1})}} \cdots \sum_{\substack{i_1 \in F(j_1) \setminus F(j_2) \\ j_0 \in C(i_1)}} \Pr[\langle j_0, i_1, j_1, \ldots, j_{k-1}, i_k, j_k, i_{k+1} \rangle \in \text{prefix}(P_{j_0})].$$

Similarly to above, by applying Lemma 2.4.4 and rearranging the terms, we have the following upper bound

$$y_{i_{k+1}} \sum_{\substack{i_k \in F(j_k) \\ j_{k-1} \in C(i_k)}} \frac{y_{i_k}}{|C(i_k)|} \Big(\sum_{i_{k-1} \in F(j_{k-1}) \setminus F(j_k) j_{k-2} \in C(i_{k-1})} \frac{y_{i_{k-1}}}{|C(i_{k-1})| y(F(j_{k-1}, j_k))} \cdots \Big(\sum_{\substack{i_1 \in F(j_1) \setminus F(j_2) \\ j_0 \in C(i_1)}} \frac{y_{i_1}}{|C(i_1)| y(F(j_1, j_2))} \Big) \Big).$$

Again, we have that the last term is at most $1/2$, and repeating that argument now for the last $k-1$ terms allows us to upper bound the expected number of connection paths by

$$\frac{1}{2^{\max(0,k-1)}} y_{i_{k+1}} \sum_{\substack{i_k \in F(j_k) \\ j_{k-1} \in C(i_k)}} \frac{y_{i_k}}{|C(i_k)|}.$$

As $y(F(j_k)) = 1$, this equals $\frac{1}{2^{\max(0,k-1)}} y_{i_{k+1}} = \frac{1}{2^{\max(0,k-1)}} x_{i_{k+1} j_k}$ as required.

□

The following corollary follows from the fact that the expected number of connection paths that traverse arc (i,j) is at most $\sum_{k=1}^{\infty} \frac{x_{ij}}{2^{\max(0,k-2)}} = 3x_{ij}$ and the expected number of connection paths that traverse arc (j,i) is at most $\sum_{k=0}^{\infty} \frac{x_{ij}}{2^{\max(0,k-1)}} = 3x_{ij}$.

Corollary 2.4.6. *The expected number of connections paths that traverse the arc (i,j) (and respectively arc (j,i)) is at most $3x^t_{ij}$. Hence, the expected number of connection paths that traverse the edge $\{i,j\}$ in any direction is at most $6x^t_{ij}$.*

2.4.3 Bounding the switching cost

In this section, we prove (2.4.3) of Lemma 2.4.1. Lemma 2.4.7 shows (2.4.3) for a special case where $|Z^t| = 1$.

Lemma 2.4.7. *For some client $k \in C$, suppose that $(x^A, y^A), (x^B, y^B) \in \mathbb{P}_{\mathsf{FL}}$ satisfy $x^A_{ij} = x^B_{ij}$ for all $i \in F$ and $j \in C \setminus \{k\}$. If we use a single set of exponential clocks to construct both of their corresponding connection graphs, the expected number of clients whose connection paths are different is at most 7.*

Proof. We first compare $\mathcal{CG}(x^A)$ and $\mathcal{CG}(x^B)$ to characterize their difference. Recall that every node has exactly one outgoing arc in a connection graph. For any $j \in C \setminus \{k\}$, its

neighborhood in $\mathcal{SG}(x^A)$ and in $\mathcal{SG}(x^B)$ are the same; hence, the unique outgoing arc from j is the same in both connection graphs (note that we use a single set of exponential clocks to define both connection graphs). The only client whose outgoing arc can be different in $\mathcal{CG}(x^A)$ and $\mathcal{CG}(x^B)$ is k.

Now suppose that a facility $i \in F$ has different outgoing arcs in $\mathcal{CG}(x^A)$ and $\mathcal{CG}(x^B)$: $(i, j^A) \in \mathcal{CG}(x^A)$ and $(i, j^B) \in \mathcal{CG}(x^B)$ for $j^A \neq j^B$. This implies $\{i, j^A\} \in \mathcal{SG}(x^A)$ and $\{i, j^B\} \in \mathcal{SG}(x^B)$. We have that at least one of these two edges is absent in the other support graph, since otherwise both edges are in both support graphs and the choice of outgoing arc from i should have been consistent in both connection graphs. Suppose that $\{i, j^A\} \notin \mathcal{SG}(x^B)$; in this case, $j^A = k$, since $\{i, j^A\} \in \mathcal{SG}(x^A)$ and k is the only client whose neighborhood can be different in the two support graphs. If $\{i, j^B\} \notin \mathcal{SG}(x^A)$, $j^B = k$. In sum, if a facility $i \in F$ has different outgoing arcs in $\mathcal{CG}(x^A)$ and $\mathcal{CG}(x^B)$, one of the two outgoing arcs is towards k.

This characterization leads to the following claim:

Claim 2.4.8. *For a client $j \in C$, if $P_j(x^A)$ is different from $P_j(x^B)$, at least one of them contains k.*

Proof. Let v be the last vertex of the maximal common prefix of $P_j(x^A)$ and $P_j(x^B)$; the outgoing arcs of v are different in $\mathcal{CG}(x^A)$ and $\mathcal{CG}(x^B)$ from the choice. If v is a client, $v = k$, hence the claim follows. If v is a facility, one of its two outgoing arcs in $\mathcal{CG}(x^A)$ and $\mathcal{CG}(x^B)$ is towards k. Assume without loss of generality that $(v, k) \in \mathcal{CG}(x^A)$. In this case, either $(v, k) \in P_j(x^A)$, or $(v, k) \notin P_j(x^A)$ because k was already visited by $P_j(x^A)$. In both cases, $k \in P_j(x^A)$. □

The connection path of k automatically contains k; for any other client $j \in C \setminus \{k\}$, its connection path contains k if and only if it contains an arc (i, k) for some $i \in F$. Therefore, the lemma follows from the following claim (and its analogue for $\mathcal{CG}(x^B)$).

Claim 2.4.9. *The expected number of connection paths in $\mathcal{CG}(x^A)$ that contain (i, k) for some $i \in F$ is at most 3.*

Proof. For each $i \in F$, the expected number of connection paths that contain (i, k) is at most $3x^A_{ik}$ from Corollary 2.4.6. Thus, the expected number of connection paths that contain (i, k) for *any* $i \in F$ is at most $\sum_{i \in F} 3x^A_{ik} = 3 \sum_{i \in F} x^A_{ik} = 3$, since $(x^A, y^A) \in \mathbb{P}_{\mathsf{FL}}$. □

For some $j \in C$, if at least one of $P_j(x^A)$ and $P_j(x^B)$ contains k, one of the following is true: $j = k$ (there is one such connection path), $(i, k) \in P_j(x^A)$ for some $i \in F$ (in expectation, there are at most three such j), or $(i, k) \in P_j(x^B)$ for some $i \in F$ (again,

there are at most three such j in expectation). Thus, the expected number of clients whose connection paths are different is at most $1 + 3 + 3 = 7$. □

For the general case where $|Z^t| > 1$, Corollary 2.4.10 shows (2.4.3) by applying Lemma 2.4.7 multiple times.

Corollary 2.4.10. *For some $K \subset C$, suppose that $(x^A, y^A), (x^B, y^B) \in \mathbb{P}_{\mathsf{FL}}$ satisfy $x^A_{ij} = x^B_{ij}$ for all $i \in F$ and $j \in C \setminus K$. If we use a single set of exponential clocks to construct both of their corresponding connection graphs, the expected number of clients that are assigned to different facilities is at most $7|K|$.*

Proof. Let us denote the elements of K as $k_1, \ldots, k_{|K|}$. Note that Lemma 2.4.7 considers two sets of connection variables that are different in the neighborhood of only *one* client, whereas this Corollary considers the case where the connection variables are different around $|K|$ clients. In order to apply Lemma 2.4.7, we can construct a series of solutions $(x^0, y^0), (x^1, y^1), \ldots, (x^{|K|}, y^{|K|})$, which starts with $(x^0, y^0) = (x^A, y^A)$ and gradually looks more similar to (x^B, y^B) until it ends with $(x^{|K|}, y^{|K|}) = (x^B, y^B)$. In particular, we ensure that $x^{\ell-1}$ and x^ℓ are different only in the neighborhood of k_ℓ. By applying Lemma 2.4.7 on each consecutive pair of these solutions, we obtain that the expected number of clients whose connection paths are different in $\mathcal{CG}(x^A)$ and $\mathcal{CG}(x^B)$ is at most $7|K|$. Note that, if a client is assigned to different facilities, its connection paths have to be different. □

2.5 Preprocessing of the LP Solution

In this section, we present the two preprocessings we apply to the LP solution.

2.5.1 First preprocessing

The first preprocessing formalized by Lemma 2.2.1 is due to Eisenstat et al. [27]. For the sake of completeness, we present their preprocessing in this section under our notation.

Let $(\bar{x}, \bar{y}, \bar{z})$ be a given LP solution. In polynomial time, we output a feasible solution (x, y, z) satisfying the following:

- The cost of (x, y, z) is at most twice the cost of $(\bar{x}, \bar{y}, \bar{z})$.
- If we let $Z^t = \{j \in C \mid x^t_{ij} \neq x^{t+1}_{ij} \text{ for some } i \in F\}$ denote the set of clients that changed its fractional connection between time step t and $t + 1$, then

$$\sum_{t=1}^{T-1} |Z^t| \leq \sum_{t=1}^{T-1} \sum_{i \in F, j \in C} z^t_{ij}. \tag{2.5.1}$$

Note that this will prove Lemma 2.2.1. The vector x is constructed from $\bar{x}$ as follows:

1. For each client j, we decide on a set $L_j = \{t_0^j, t_1^j, \ldots, t_{\iota(j)}^j = T+1\}$ of *boundary time steps* that divide the entire time span into *time intervals* $[t_0^j, t_1^j), \cdots, [t_{\iota(j)-1}^j, t_{\iota(j)}^j)$:

   - Let $t_0^j = 1$ and $\iota = 1$.
   - Now select t_ι^j to be the largest $t \in (t_{\iota-1}^j, T+1]$ such that $\sum_{i\in F}(\min_{t_{\iota-1}^j \le u < t} \bar{x}_{ij}^u) \ge 1/2$.
   - If $t_\iota^j = T+1$ we are done selecting boundaries: $\iota(j) \leftarrow \iota$, and $L_j \leftarrow \{t_0^j, \ldots, t_{\iota(j)}^j\}$.
     Otherwise, we increment ι and repeat the previous step to select the next boundary.

   Note that each client defines its own division of the time span.

2. Once we have decided on these time intervals, we redefine the connection variables so that, for each client j, its connection variables do not change within each time interval defined by itself.
   For each client j and each of its time interval $[t_k^j, t_{k+1}^j)$, we set

$$x_{ij}^t := \frac{\min_{t_k^j \le u < t_{k+1}^j} \bar{x}_{ij}^u}{\sum_{i'\in F}(\min_{t_k^j \le u < t_{k+1}^j} \bar{x}_{i'j}^u)} \qquad \text{for each } i \in F \text{ and } t \in [t_k^j, t_{k+1}^j).$$

   Note that the right-hand side is not dependent on t, achieving the desired property.

By construction we have $x_{ij}^t \le \bar{x}_{ij}^t / \sum_{i'\in F}(\min_{t_k^j \le u < t_{k+1}^j} \bar{x}_{i'j}^u) \le 2\bar{x}_{ij}^t$. Therefore the connection cost of x is at most twice the connection cost of $\bar{x}$. Moreover, if we let $y = 2\bar{y}$, then $x_{ij}^t \le y_i^t$ for all $i \in F, j \in C$ and $t \in [T]$. Finally, by construction $\sum_{i\in F} x_{ij}^t = 1$ for all $j \in C$ and $t \in [T]$, hence we can conclude that

$$(x^t, y^t) \in \mathbb{P}_{\mathsf{FL}} \qquad \text{for all } t \in [T].$$

We finish the description of the preprocessing by specifying z. Towards this aim we use the following fact from [27]:

Fact 2.5.1. *For each client $j \in C$ and its time interval $[t_{k-1}^j, t_k^j)$ where $1 \le k < \iota(j)$,*

$$\sum_{t_{k-1}^j \le t < t_k^j} \sum_{i\in F} \bar{z}_{ij}^t > 1/2.$$

Note that this implies that $\sum_{j\in C}(\iota(j)-1) \le \sum_{t=1}^{T-1} \sum_{i\in F, j\in C} 2\bar{z}_{ij}^t$. On the other hand, recall that the fractional connection of a client j does not change within each of its time

interval from constuction; thus, $\sum_{t=1}^{T-1} |Z^t| \leq \sum_{j \in C} (\iota(j) - 1)$. We have, as a result,

$$\sum_{t=1}^{T-1} |Z^t| \leq \sum_{t=1}^{T-1} \sum_{i \in F, j \in C} 2\bar{z}_{ij}^t.$$

Therefore, if we can obtain (feasible) z such that $\sum_{t=1}^{T-1} \sum_{i \in F, j \in C} z_{ij}^t = \sum_{t=1}^{T-1} \sum_{i \in F, j \in C} 2\bar{z}_{ij}^t$, (2.5.1) would follow, and the switching cost would be at most twice the original switching cost.

We first set $z_{ij}^t := x_{ij}^t - x_{ij}^{t+1}$ for all $i \in F, j \in C$ and $t \in [T)$, and we have $\sum_{t=1}^{T-1} \sum_{i \in F, j \in C} z_{ij}^t \leq \sum_{j \in C} (\iota(j) - 1)$ from construction, which in turn implies $\sum_{t=1}^{T-1} \sum_{i \in F, j \in C} z_{ij}^t \leq \sum_{t=1}^{T-1} \sum_{i \in F, j \in C} 2\bar{z}_{ij}^t$. Then we can increase z in an arbitrary manner until we achieve $\sum_{t=1}^{T-1} \sum_{i \in F, j \in C} z_{ij}^t = \sum_{t=1}^{T-1} \sum_{i \in F, j \in C} 2\bar{z}_{ij}^t$.

2.5.2 Second preprocessing

The second preprocessing formalized by Observation 2.2.2 follows from applying the standard technique of duplicating facilities. We present the details in this section.

Let us first review the standard technique that is used by multiple algorithms for the classic problem: it ensures $x_{ij} \in \{0, y_i\}$. Suppose there exists a facility i that is open by the fraction of .6 and connected to clients a, b, c, and d each by .1, .4, .4, and .6, respectively, in the LP solution. The standard technique in this case duplicates i into three copies, each of which is to be opened by .1, .3, and .2. Then a, b, c, and d is respectively connected to the first one, two, two, and three copies of the facility. Figure 2.2 illustrates this duplication. Note that this technique modifies the problem instance since it creates copies of facilities. However, we define the metric so that the copied facilities are exactly at the same position as the original facility: e.g., $d(i, j) = d(i_1, j)$ for all $j \in C$, and the copied facilities are also defined to have the same opening cost as the original. Thus, this modification does not change the cost of the LP solution, and if we find an (approximate) solution to the new instance, it translates back to the original instance by opening the original facilities instead of their duplicates. In general, the connection variables of each facility determine the set of "threshold" values ($\{.1,\ .4,\ .6\}$ in this case) to be used to split that facility, and the facility is split into multiple copies each of which is to be opened by the fraction equal to the difference of two consecutive threshold values (.1, $.4 - .1$, and $.6 - .4$ in this case).

For the dynamic problem, we additionally need to ensure that each copy of a facility is open by the same fraction (or by zero) at every time step (Property 2 of Observation 2.2.2). In order to obtain this property, every time step will share a single set of threshold values: we determine the thresholds for facility $i \in F$ by taking all its LP variables across all the time steps: $\{x_{ij}^t \mid j \in C, t \in [T]\} \cup \{y_i^t \mid t \in [T]\}$. Figure 2.3 shows an example: the

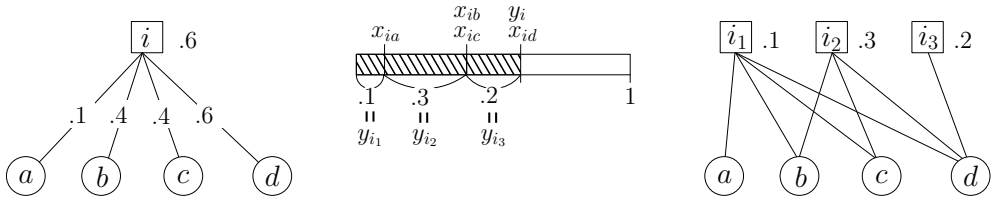


Figure 2.2 – The standard technique applied to the classic problem. (Part of) an LP solution is shown on the left. Numbers on the edges show the connection variables; next to the facilities (represented as squares) are the opening variables. Result of the preprocessing is shown on the right: connection variables are omitted, as they are equal to the incident opening variables.

set of threshold values is $\{.1, .3, .4, .7, .8\}$. Therefore we create five copies of the facility, each of which is to be opened respectively by .1, .2, .1, .3, and .1. Now the openings and connections of the facility are implemented by subsets of the first few copies of the facility.

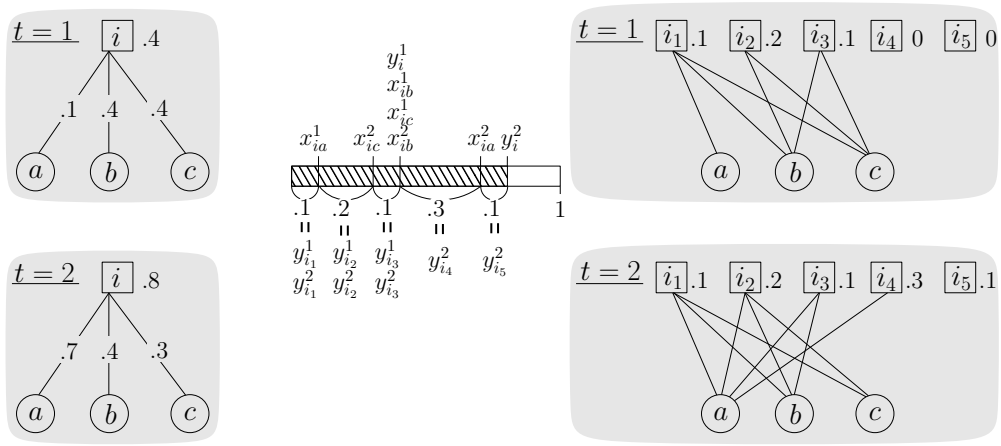


Figure 2.3 – The second preprocessing. $T = 2$.

Finally, note that the connection variables of the same value are implemented by connecting to the same set of copies. In Figure 2.3, $x_{ib}^1 = x_{ib}^2 = .4$ and both are implemented by connecting to $\{i_1, i_2, i_3\}$ in their respective time steps. Thus, if a client j has exactly the same set of connection variables in two time steps, this remains the case even after the preprocessing.

3 The k-Means and k-Median Problems

This chapter is based on a joint work with Sara Ahmadian, Ola Svensson and Justin Ward, published in 58th Annual Symposium on Foundations of Computer Science (FOCS) 2017 [2].

3.1 Introduction

In this chapter, we focus on one of the most widely considered clustering problems, the so-called *k-means* problem. Let us briefly recall the definition of this problem: Given a set $\mathcal{D}$ of n points in $\mathbb{R}^\ell$ and an integer k, the task is to select a set S of k *cluster centers* in $\mathbb{R}^\ell$, so that $\sum_{j\in\mathcal{D}} c(j,S)$ is minimized, where $c(j,S)$ is the squared Euclidean distance between j and its nearest center in S.

The k-means problem has been well-studied experimentally and practically [36]. One of the most commonly used heuristics for k-means is Lloyd's algorithm [48], which is based on iterative improvements. Despite its ubiquity in practice, Lloyd's algorithm has, in general, no worst-case guarantee and might not even converge in polynomial time [8, 59]. Arthur and Vassilvitskii [9] propose a randomized initialization procedure for Lloyd's algorithm, called k-means++; it leads to a $\Theta(\log k)$ expected approximation guarantee in the worst case. Under additional assumptions about the *clusterability* of the input dataset, Ostrovsky et al. [53] show that Lloyd's algorithm gives a PTAS for k-means clustering.

Various other PTASes have been developed for restricted instances of the k-means problem, under a variety of assumptions. For example, Awasthi, Blum, and Sheffet obtain a PTAS assuming the instance has certain stability properties [13], and there is a long line of work (beginning with [49]) that obtains better and better PTASes under the assumption that k is constant. Most recently, it has been shown that local search gives a PTAS under the assumption that the dimension ℓ of the dataset is constant [25, 32]. These last results generalize to the case in which the squared distances are from

the shortest path metric on a graph with forbidden minors [25] or from a metric with constant doubling dimension [32].

Under no additional assumptions, however, the best approximation for the general k-means problem has for some time remained a $(9+\epsilon)$-approximation algorithm based on local search, due to Kanungo et al. [41]. Their analysis shows that no natural local search algorithm performing a fixed number of swaps can improve upon this ratio. In terms of hardness, Awasthi, Charikar, Krishnaswamy, and Sinop [14] show that k-means is APX-hard, and so we cannot hope for a PTAS in the general case. Follow-up work by Lee, Schmidt, and Wright [44] shows that it is NP-hard to approximate this problem to within a factor better than 1.0013.

In summary, while k-means is perhaps the most widely used clustering problem in computer science, the only constant-factor approximation algorithm for the general case is based on simple local search heuristics. This is in stark contrast to many other well-studied clustering problems, such as facility location and k-median. Over the past several decades, work on these problems has been responsible for the refinement of a variety of core techniques in approximation algorithms such as dual fitting, primal-dual, and LP-rounding [56, 24, 20, 45, 47, 39, 38, 46, 37]. The development of these techniques has led to several breakthroughs giving the current best approximation guarantees for both facility location (a 1.488-approximation due to Li [45]) and k-median (a 2.675-approximation due to Byrka et al. [21]). In both cases, LP-based techniques now give significantly better results than previous local search algorithms [11, 23]. These techniques have not yet been able to attain similar improvements for k-means primarily because they have relied heavily on the triangle inequality, which does not hold in the case of k-means.

Our results In this work, we overcome this barrier by developing new techniques that enables us to exploit the standard LP formulation for k-means. We significantly narrow the gap between known upper and lower bounds by designing a new primal-dual algorithm for the k-means problem. We stress that our algorithm works in the general case that k and ℓ are part of the input, and requires no additional assumptions on the dataset.

Theorem 3.1.1. *For any $\epsilon > 0$, there is a $(\rho_{mean} + \epsilon)$-approximation algorithm for the k-means problem, where $\rho_{mean} \approx 6.357$. Moreover, the integrality gap of the standard LP is at most ρ_{mean}.*

We now describe our approach and contributions at a high level. Given a k-means instance, we apply standard discretization techniques (e.g., [28]) to obtain an instance of the *discrete k-means* problem, in which we are given a discrete set $\mathcal{F}$ of candidate centers in $\mathbb{R}^\ell$ and must select k centers from $\mathcal{F}$, rather than k arbitrary points in $\mathbb{R}^\ell$.

This step incurs an arbitrarily small loss in the approximation guarantee, with respect to the original k-means instance. Using Lagrangian relaxation, we can then consider the resulting discrete problem by using the standard linear programming formulation for facility location.

Our approach then starts with the framework of Jain and Vazirani [39] for the k-median problem. In their paper, they first present a *Lagrangian Multiplier Preserving* (LMP) 3-approximation algorithm for the facility location problem. Then they run binary search over the opening cost of the facilities and use the aforementioned algorithm to find two solutions: one that opens more than k facilities and one that opens less than k, such that the opening cost of facilities in these solutions are close to each other. These solutions are then combined to obtain a solution that opens exactly k facilities. This step results in losing another factor 2 in the approximation guarantee, which results in a 6-approximation algorithm for k-median. The factor 6 was later improved by Jain, Mahdian, and Saberi [38] who obtained a 4-approximation algorithm for k-median by developing an LMP 2-approximation algorithm for facility location.

One can see that the same approach gives a much larger constant factor for the k-means problem since one can no longer rely on the triangle inequality. We use two main ideas to overcome this obstacle: (1) we exploit the geometric structure of k-means and (2) we develop a new primal-dual approach. Specifically, we modify the primal-dual algorithm of Jain and Vazirani [39] into a parameterized version, which enables us to regulate the "agressiveness" of the opening strategy of facilities. By using properties of Euclidean metrics, we show that this leads to improved LMP approximation algorithms for k-means.

By the virtue of [7], these results already imply upper bounds on the integrality gaps of the standard LP relaxations, albeit with a rounding algorithm that might require exponential time. Our second main contribution is a new polynomial time algorithm that accomplishes the same task. Several new ideas are required to obtain this rounding. Specifically, instead of finding two solutions by binary search as in the framework of [39], we find a sequence of solutions such that the opening costs and the dual values of any two consecutive solutions are close in L^∞-norm. We show how to combine two appropriate such solutions to obtain a solution that opens exactly k facilities and loses only a factor $1 + \epsilon$ in the approximation guarantee.

Extensions to other problems In addition to the standard k-means problem, we show that our results also extend to the following two problems. In the first extension, we consider the Euclidean k-median problem. Here we are given a set $\mathcal{D}$ of n points in $\mathbb{R}^\ell$ and a set $\mathcal{F}$ of m points in $\mathbb{R}^\ell$ corresponding to facilities. The task is to select a set S of at most k facilities from $\mathcal{F}$ so as to minimize $\sum_{j\in\mathcal{D}} c(j, S)$, where $c(j, S)$ is now the (non-squared) Euclidean distance from j to its nearest facility in S. For this problem, no approximation better than the general 2.675-approximation algorithm of Byrka et

al. [21] for k-median was known.

Theorem 3.1.2. *For any $\epsilon > 0$, there is a $(\rho_{\mathsf{med}} + \epsilon)$-approximation algorithm for the Euclidean k-median problem, where $\rho_{\mathsf{med}} \approx 2.633$. Moreover, the integrality gap of the standard LP is at most ρ_{med}.*

In the second extension, we consider a variant of the k-means problem in which each $c(j, S)$ corresponds to the squared distance in an arbitrary (possibly non-Euclidean) metric on $\mathcal{D} \cup \mathcal{F}$. For this problem, the best-known approximation algorithm is a 16-approximation thanks to Gupta and Tangwongsan [33]. In this thesis, we obtain the following improvement:

Theorem 3.1.3. *For any $\epsilon > 0$, there is a $(9 + \epsilon)$-approximation algorithm for the k-means problem in general metrics. Moreover, the integrality gap of the standard LP is at most 9.*

We remark that the same hardness reduction as used for k-median [38] immediately yields a hardness result for the above generalization much stronger than what is known for the standard k-means problem: it is hard to approximate the k-means problem in general metrics within a factor $1 + 8/e - \epsilon \approx 3.94$ for any $\epsilon > 0$.

Outline of this chapter. In Section 3.2, we review the standard LP formulation that we use, as well as its Lagrangian relaxation. In Section 3.3, we show how to exploit the geometric structure k-means and Euclidean k-median to give improved LMP guarantees. In Section 3.4, we show the main ideas behind our new rounding approach by giving an algorithm that runs in quasi-polynomial time. Then, in order to obtain an algorithm that runs in polynomial time, we generalize the results in Sections 3.5, 3.6, and 3.7.

3.2 The Standard LP Relaxation and Its Lagrangian Relaxation

Here and in the remainder of this chapter, we shall consider the *discrete* k-means problem, where we are given a discrete set $\mathcal{F}$ of facilities (corresponding to candidate centers).[1] Henceforth, we will simply refer to the discrete k-means problem as the k-means problem.

Given an instance $(\mathcal{D}, \mathcal{F}, d, k)$ of the k-means problem or the k-median problem, let $c(j, i)$ denote the connection cost of client j if connected to facility i. That is, $c(j, i) = d(j, i)$ in the case of k-median and $c(j, i) = d(j, i)^2$ in the case of k-means. Let $n = |\mathcal{D}|$ and $m = |\mathcal{F}|$.

[1] As discussed in the introduction, it is well-known that a ρ-approximation algorithm for this case can be turned into a $(\rho + \epsilon)$-approximation algorithm for the standard k-means problem, for any constant $\epsilon > 0$ (see e.g., [28]).

Similar to the previously discussed LP for the facility location problem, the standard LP relaxation of these problems has two sets of variables: a variable y_i for each facility $i \in \mathcal{F}$ and a variable x_{ij} for each facility-client pair $i \in \mathcal{F}, j \in \mathcal{D}$. The intuition of these variables is that y_i should indicate whether facility i is opened and x_{ij} should indicate whether client j is connected to facility i. The standard LP relaxation can now be formulated as follows.

$$\begin{aligned}
\min \quad & \sum_{i \in \mathcal{F}, j \in \mathcal{D}} x_{ij} \cdot c(j,i) & & \\
\text{s.t.} \quad & \sum_{i \in \mathcal{F}} x_{ij} \geq 1 & \forall j \in \mathcal{D} & \\
& & & (3.2.1) \\
& x_{ij} \leq y_i & \forall j \in \mathcal{D}, i \in \mathcal{F} & \\
& & & (3.2.2) \\
& \sum_{i \in \mathcal{F}} y_i \leq k & & (3.2.3) \\
& x, y \geq 0\,. & & (3.2.4)
\end{aligned}$$

The first set of constraints says that each client should be connected to at least one facility; the second set of constraints enforces that clients can only be connected to opened facilities; and the third constraint says that at most k facilities can be opened. We remark that this is a relaxation of the original problem, as we have relaxed the constraint that x and y should take Boolean values to a non-negativity constraint. For future reference, we let OPT_k denote the value of an optimal solution to this relaxation.

A main difficulty for approximating the k-median and the k-means problems is the hard constraint that at most k facilities can be selected, i.e., constraint (4.2.4) in the above relaxation. A popular way of overcoming this difficulty, pioneered in this context by Jain and Vazirani [39], is to consider the Lagrangian relaxation where we multiply the constraint (4.2.4) times a Lagrange multiplier λ and move it to the objective. This results, for every $\lambda \geq 0$, in the following relaxation and its dual that we denote by $\mathsf{LP}(\lambda)$ and $\mathsf{DUAL}(\lambda)$, respectively.

$\mathsf{LP}(\lambda)$

$$\min \quad \sum_{i\in\mathcal{F}, j\in\mathcal{D}} x_{ij}\cdot c(j,i) + \lambda\cdot\left(\sum_{i\in\mathcal{F}} y_i - k\right)$$

s.t. (4.2.2), (4.2.3), and (4.2.5).

$\mathsf{DUAL}(\lambda)$

$$\max \quad \sum_{j\in\mathcal{D}} \alpha_j - \lambda\cdot k$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{D}} [\alpha_j - c(j,i)]^+ \le \lambda \qquad \forall i\in\mathcal{F} \qquad (3.2.5)$$

$$\alpha \ge 0.$$

Here, we have simplified the dual by noticing that the dual variables $\{\beta_{ij}\}_{i\in\mathcal{F}, j\in\mathcal{D}}$ corresponding to the constraints (4.2.3) of the primal can always be set $\beta_{ij} = [\alpha_j - c(j,i)]^+$; the notation $[a]^+$ denotes $\max(a, 0)$. Moreover, to see that $\mathsf{LP}(\lambda)$ remains a relaxation, note that any feasible solution to the original LP is a feasible solution to the Lagrangian relaxation of no higher cost. In other words, for any $\lambda \ge 0$, the optimum value of $\mathsf{LP}(\lambda)$ is at most OPT_k.

If we disregard the constant term $\lambda\cdot k$ in the objective functions, $\mathsf{LP}(\lambda)$ and $\mathsf{DUAL}(\lambda)$ become the standard LP formulation and its dual for the facility location problem where the opening cost of each facility equals λ and the connection costs are defined by $c(\cdot,\cdot)$. Recall that the facility location problem (with uniform opening costs) is defined as the problem of selecting a set $S \subseteq \mathcal{F}$ of facilities to open so as to minimize the opening cost $|S|\lambda$ plus the connection cost $\sum_{j\in\mathcal{D}} c(j,S)$. Jain and Vazirani [39] introduced the following method for addressing the k-median problem motivated by simple economics. On the one hand, if λ is selected to be very small, i.e., it is inexpensive to open facilities, then a good algorithm for the facility location problem will open many facilities. On the other hand, if λ is selected to be very large, then a good algorithm for the facility location problem will open few facilities. Ideally, we want to use this intuition to find an opening price that leads to the opening of exactly k facilities and thus a solution to the original, constrained problem.

To make this intuition work, we need the notion of *Lagrangian Multiplier Preserving* (LMP) approximations: We say that a ρ-approximation algorithm is LMP for the facility

location problem with opening costs λ if it returns a solution $S \subseteq \mathcal{F}$ satisfying

$$\sum_{j \in \mathcal{D}} c(j, S) \leq \rho(\mathrm{OPT}(\lambda) - |S|\lambda)\,,$$

where $\mathrm{OPT}(\lambda)$ denotes the value of an optimal solution to $\mathsf{LP}(\lambda)$ without the constant term $\lambda \cdot k$. The importance of this definition becomes apparent when either $\lambda = 0, |S| \leq k$ or $|S| = k$. In these cases, we can see that the value of the k-median or k-means solution is at most ρ times the optimal value of its relaxation $\mathsf{LP}(\lambda)$, and thus an ρ-approximation with respect to its standard LP relaxation since $\mathrm{OPT}(\lambda) - k \cdot \lambda \leq \mathrm{OPT}_k$ for any $\lambda \geq 0$.

3.3 Exploiting Euclidean Metrics via Primal-Dual Algorithms

In this section we show how to exploit the structure of Euclidean metrics to achieve better approximation guarantees. Our algorithm is based on the primal-dual algorithm for the facility location problem by Jain and Vazirani [39]. We refer to their algorithm as the JV algorithm. The main modification to their algorithm is that we permit for a more "aggressive" opening strategy of facilities. The amount of aggressiveness is measured by the parameter δ: we devise an algorithm $\mathsf{JV}(\delta)$ for each parameter $\delta \geq 0$, where a smaller δ results in a more aggressive opening strategy. We first describe $\mathsf{JV}(\delta)$, we then optimize δ for the considered objectives to obtain the claimed approximation guarantees.

We remark that the result in [7] (non-constructively) upper bounds the integrality gap of the standard LP relaxation of k-median in terms of the LMP approximation guarantee of JV. This readily generalizes to the k-means problem and $\mathsf{JV}(\delta)$. Consequently, our guarantees presented here upper bound the integrality gaps, as the theorems state in the introduction.

3.3.1 Description of $\mathsf{JV}(\delta)$

As alluded to above, the algorithm is a modification of JV, and Remark 3.3.2 below highlights the only difference. The algorithm consists of two phases: the dual-growth phase and the pruning phase.

Dual-growth phase: In this stage, we construct a feasible dual solution α to $\mathsf{DUAL}(\lambda)$. Initially, we set $\alpha = \mathbf{0}$ and let $A = \mathcal{D}$ denote the set of active clients (which is all clients at first). We then repeat the following until there are no active clients, i.e., $A = \emptyset$: increase the dual-variables $\{\alpha_j\}_{j \in A}$ corresponding to the active clients at a uniform rate until one of the following events occur (if several events happen at the same time, break ties arbitrarily):

Event 1: A dual constraint $\sum_{j\in\mathcal{D}}[\alpha_j - c(j,i)]^+ \leq \lambda$ becomes tight for a facility $i \in \mathcal{F}$. In this case we say that facility i is *tight* or *temporarily opened*. We update A by removing the active clients with a *tight edge* to i, that is, a client $j \in A$ is removed if $\alpha_j - c(j,i) \geq 0$. For future reference, we say that facility i is the *witness* of these removed clients.

Event 2: An active client $j \in A$ gets a tight edge, i.e., $\alpha_j - c(j,i) = 0$, to some already tight facility i. In this case, we remove j from A and let i be its witness.

This completes the description of the dual-growth phase. Before proceeding to the pruning phase, let us remark that the constructed α is indeed a feasible solution to $\mathsf{DUAL}(\lambda)$ by design. It is clear that α is non-negative. Now consider a facility $i \in \mathcal{F}$ and its corresponding dual constraint $\sum_{j\in\mathcal{D}}[\alpha_j - c(j,i)]^+ \leq \lambda$. On the one hand, the constraint is clearly satisfied if it never becomes tight during the dual-growth phase. On other hand, if it becomes tight, then all clients with a tight edge to it are removed from the active set of clients by Event 1. Moreover, if any client gets a tight edge to i in subsequent iterations it gets immediately removed from the set of active clients by Event 2. Therefore, the left-hand side of the constraint will never increase (nor decrease) after it becomes tight so the constraint remains satisfied. Having proved that α is a feasible solution to $\mathsf{DUAL}(\lambda)$, let us now describe the pruning phase.

Pruning phase: After the dual-growth phase (too) many facilities are temporarily opened. The pruning phase will select a subset of these facilities to open. In order to formally describe this process, we need the following notation. For a client j, let $N(j) = \{i \in \mathcal{F} : \alpha_j - c(j,i) > 0\}$ denote the facilities to which client j contributes to the opening cost. Similarly, for $i \in \mathcal{F}$, let $N(i) = \{j \in \mathcal{D} : \alpha_j - c(j,i) > 0\}$ denote the clients with a positive contribution toward i's opening cost. For a temporarily opened facility i, let

$$t_i = \max_{j\in N(i)} \alpha_j \,,$$

and by convention let $t_i = 0$ if $N(i) = \emptyset$ (this convention will be useful in future sections and will only be used when the opening cost λ of facilities are set to 0). Note that, if $N(i) \neq \emptyset$, then t_i equals the "time" that facility i was temporarily opened in the dual-growth phase. A crucial property of t_i that follows from the construction of α is the following.

Claim 3.3.1. *For a client j and its witness i, $\alpha_j \geq t_i$. Moreover, for any $j' \in N(i)$ we have $t_i \geq \alpha_{j'}$.*

A key ingredient for the pruning phase is the *client-facility graph* G and the *conflict*

graph H. The vertex set of G consists of all the clients and all facilities i such that $\sum_{j\in\mathcal{D}}[\alpha_j - c(j,i)]^+ = \lambda$ (i.e., the tight or temporarily open facilities). There is an edge between facility i and client j if $i \in N(j)$. The conflict graph H is defined based on the client-facility graph G and t as follows:

- The vertex set consists of all facilities in G.
- There is an edge between two facilities i and i' if some client j is adjacent to both of them in G and $c(i,i') \le \delta \min(t_i, t_{i'})$.

The pruning phase now finds a (inclusion-wise) maximal independent set IS of H and opens those facilities; clients are connected to the closest facility in IS.

Remark 3.3.2. *The only difference between the original algorithm JV and our modified JV(δ) is the additional condition $c(i,i') \le \delta \min(t_i, t_{i'})$ in the definition of the conflict graph. Notice that if we select a smaller δ, we will have fewer edges in H. Therefore a maximal independent set will likely grow in size, which results in a more "aggressive" opening strategy.*

3.3.2 Analysis of JV(δ) for the considered objectives

In the following subsections, we optimize δ and analyze the guarantees obtained by the algorithm JV(δ) for the objective functions: k-means objective in general metrics, standard k-means objective (in Euclidean metrics), and k-median objective in Euclidean metrics. The first analysis is very similar to the original JV analysis and can also serves as a motivation for the possible improvements in Euclidean metrics.

k-Means objective in general metrics

We consider the case when $c(j,i) = d(j,i)^2$ and d forms a general metric. We let $\delta = \infty$ so JV(δ) becomes simply the JV algorithm. We prove the following.

Theorem 3.3.3. *Let d be any metric on $\mathcal{D} \cup \mathcal{F}$ and suppose that $c(j,i) = d(j,i)^2$ for every $i \in \mathcal{F}$ and $j \in \mathcal{D}$. Then, for any $\lambda \ge 0$, Algorithm JV(∞) constructs a solution α to DUAL(λ) and returns a set IS of opened facilities such that*

$$\sum_{j\in D} c(j,\mathsf{IS}) \le 9 \cdot \left(\sum_{j\in\mathcal{D}} \alpha_j - \lambda|\mathsf{IS}|\right).$$

Proof. Consider any client $j \in \mathcal{D}$. We shall prove that

$$\frac{c(j,\mathsf{IS})}{9} \leq \alpha_j - \sum_{i \in N(j) \cap \mathsf{IS}} (\alpha_j - c(j,i)) = \alpha_j - \sum_{i \in \mathsf{IS}} [\alpha_j - c(j,i)]^+ . \tag{3.3.1}$$

The statement then follows by summing up over all clients and noting that any facility $i \in \mathsf{IS}$ was temporarily opened, thus we have $\sum_{j \in \mathcal{D}} [\alpha_j - c(j,i)]^+ = \lambda$.

To prove (3.3.1), we first note that $|\mathsf{IS} \cap N(j)| \leq 1$. Indeed, consider $i \neq i' \in N(j)$. Then (j,i) and (j,i') are edges in the client-facility graph G and as $\delta = \infty$, i and i' are adjacent in the conflict graph H. Hence, the temporarily opened facilities in $N(j)$ form a clique and at most one of them can be selected in the maximal independent set IS. We complete the analysis by considering the two cases $|\mathsf{IS} \cap N(j)| = 1$ and $|\mathsf{IS} \cap N(j)| = 0$.

Case $|\mathsf{IS} \cap N(j)| = 1$: Let i^* be the unique facility in $\mathsf{IS} \cap N(j)$. Then

$$\frac{c(j,\mathsf{IS})}{9} \leq c(j,\mathsf{IS}) \leq c(j,i^*) = \alpha_j - (\alpha_j - c(j,i^*)) = \alpha_j - \sum_{i \in N(j) \cap \mathsf{IS}} (\alpha_j - c(j,i)).$$

Notice the amount of slack in the above analysis (specifically, the first inequality). In the Euclidean case, we exploit this slack for a more aggressive opening and to improve the approximation guarantee.

Case $|\mathsf{IS} \cap N(j)| = 0$: Let i_1 be j's witness. First, if $i_1 \in \mathsf{IS}$ then by the same arguments as above we have the desired inequality; specifically, since j has a tight edge to i_1 but $i_1 \notin N(j)$ we must have $\alpha_j = c(i_1, j)$. Now consider the more interesting case when $i_1 \notin \mathsf{IS}$. As IS is a maximal independent set in H, there must be a facility $i_2 \in \mathsf{IS}$ that is adjacent to i_1 in H. By definition of H, there is a client j_1 such that (j_1, i_1) and (j_1, i_2) are edges in the client-facility graph G, i.e., $j_1 \in N(i_1) \cap N(i_2)$. By the definition of witness and $N(\cdot)$, we have

$$\alpha_j \geq c(j, i_1,) , \qquad \alpha_{j_1} > c(j_1, i_1) , \qquad \alpha_{j_1} > c(j_1, i_2) ,$$

and by the description of the algorithm (see Claim 3.3.1 in Section 3.3) we have $\alpha_j \geq t_{i_1} \geq \alpha_{j_1}$. Hence, using the triangle inequality and that $(a+b+c)^2 \leq 3(a^2+b^2+c^2)$,

$$\begin{aligned} c(j,\mathsf{IS}) \leq c(j,i_2) = d(j,i_2)^2 &\leq (d(j,i_1) + d(j_1,i_1) + d(j_1,i_2))^2 \\ &\leq 3(d(j,i_1)^2 + d(j_1,i_1)^2 + d(j_1,i_2)^2) \\ &= 3(c(j,i_1) + c(j_1,i_1) + c(j_1,i_2)) \leq 9\alpha_j . \end{aligned} \tag{3.3.2}$$

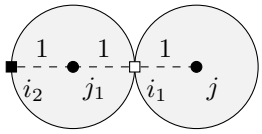
As $\sum_{i \in N(j) \cap \mathsf{IS}} (\alpha_j - c(j,i)) = 0$, this completes the proof of this case and thus the theorem.

□

k-Means objective in Euclidean metrics

Worst case configuration

The clients and the facilities are arranged on a line and we have $c(i_2, j) = d(i_2, j)^2 = 9\alpha_j$.



Better case in Euclidean space

The distance $d(j, i_2)$ is better than that the triangle inequality gives yielding a better bound.

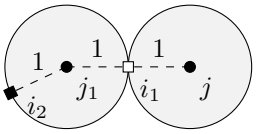


Figure 3.1 – The intuition how we improve the guarantee in the Euclidean case. In both cases, we have $\alpha_j = \alpha_{j_1} = 1$. Moreover, $i_1 \notin \mathsf{IS}, i_2 \in \mathsf{IS}$ and we are interested in bounding $c(j, i_2)$ as a function of α_j.

We begin with some intuition that illustrates our approach. From the standard analysis of JV (and our analysis of k-means in general metrics), it is clear that the bottleneck for the approximation guarantee comes from the connection-cost analysis of clients that need to do a "3-hop" as illustrated in the left part of Figure 3.1: client j is connected to open facility i_2 and the squared-distance is bounded by the path $j - i_1 - j_1 - i_2$. Furthermore, this analysis is tight when considering $\mathsf{JV} = \mathsf{JV}(\infty)$. Our strategy will now be as follows: Select δ to be a constant smaller than 4. This means that in the configurations of Figure 3.1, we will also open i_2 if the distance between i_1 and i_2 is close to 2. Therefore, if we do not open i_2, the distance between i_1 and i_2 is less than 2 (as in the right part of Figure 3.1) which enables us to obtain an approximation guarantee better than 9. However, this might result in a client contributing to the opening cost of many facilities in IS. Nonetheless, by using the properties of Euclidean metrics, we show that, even in this case, we are able to achieve a LMP approximation guarantee with ratio better than 9.

Specifically, define δ_{mean} to be the constant larger than 2 that minimizes

$$\rho_{\mathsf{mean}}(\delta) = \max\left\{(1+\sqrt{\delta})^2, \frac{1}{\delta/2-1}\right\},$$

which will be our approximation guarantee. It can be verified that $\delta_{\mathsf{mean}} \approx 2.3146$ and $\rho_{\mathsf{mean}} \approx 6.3574$. Let also $c(j, i) = d(j, i)^2$ where d is the underlying Euclidean metric. The proof uses the following basic facts about squared-distances in Euclidean metrics: given $x_1, x_2, \ldots, x_s \in \mathbb{R}^\ell$, we have that $\min_{y\in\mathbb{R}^\ell} \sum_{i=1}^s \|x_i - y\|_2^2$ is attained by the *centroid* $\mu = \frac{1}{s}\sum_{i=1}^s x_i$ and in addition we have the identity $\sum_{i=1}^s \|x_i - \mu\|_2^2 = \frac{1}{2s}\sum_{i=1}^s \sum_{j=1}^s \|x_i - x_j\|_2^2$.

Theorem 3.3.4. *Let d be a Euclidean metric on $\mathcal{D} \cup \mathcal{F}$ and suppose that $c(j, i) = d(j, i)^2$ for every $i \in \mathcal{F}$ and $j \in \mathcal{D}$. Then, for any $\lambda \geq 0$, Algorithm JV(δ_{mean}) constructs a solution*

α to DUAL(λ) and returns a set IS *of opened facilities such that*

$$\sum_{j\in S} c(j,\mathsf{IS}) \le \rho_{\mathit{mean}} \cdot \Big(\sum_{j\in\mathcal{D}} \alpha_j - \lambda|\mathsf{IS}|\Big)\,.$$

Proof. To simplify notation, we use δ instead of δ_{mean} throughout the proof. Consider any client $j \in \mathcal{D}$. We prove that

$$\frac{c(j,\mathsf{IS})}{\rho_{\mathsf{mean}}} \le \alpha_j - \sum_{i\in N(j)\cap\mathsf{IS}} (\alpha_j - c(j,i)) = \alpha_j - \sum_{i\in\mathsf{IS}} [\alpha_j - c(j,i)]^+\,.$$

Similarly to the proof of Theorem 3.3.3, the statement then follows by summing up over all clients. A difference compared to the standard analysis of JV is that in our algorithm we may open several facilities in $N(j)$, i.e., client j may contribute to the opening of several facilities. We divide our analysis into the three cases $|N(j)\cap\mathsf{IS}| = 1$, $|N(j)\cap\mathsf{IS}| > 1$, and $|N(j)\cap\mathsf{IS}| = 0$. For brevity, let S denote $N(j)\cap\mathsf{IS}$ and $s = |S|$.

Case $s = 1$: If we let i^* be the unique facility in S,

$$\frac{c(j,\mathsf{IS})}{\rho_{\mathsf{mean}}} \le c(j,\mathsf{IS}) \le c(j,i^*) = \alpha_j - (\alpha_j - c(j,i^*)) = \alpha_j - \sum_{i\in N(j)\cap\mathsf{IS}} (\alpha_j - c(j,i))\,.$$

Case $s > 1$: In this case, there are multiple facilities in IS that j is contributing to. We need to show that $\alpha_j - \sum_{i\in S}(\alpha_j - c(j,i)) \ge \frac{1}{\rho_{\mathsf{mean}}} c(j,\mathsf{IS})$.

The sum $\sum_{i\in S} c(j,i)$ is the sum of square distances from j to facilities in S which is at least the sum of square distances of these facilities from their centroid μ, i.e., $\sum_{i\in S} c(j,i) \ge \sum_{i\in S} c(i,\mu)$. Moreover, by the identity, $\sum_{i\in S} c(i,\mu) = \frac{1}{2s}\sum_{i,i'\in S} c(i,i')$, we get

$$\sum_{i\in S} c(j,i) \ge \frac{1}{2s} \sum_{i,i'\in S} c(i,i')\,.$$

As there is no edge between any pair of facilities in $S \subseteq \mathsf{IS}$, we must have

$$c(i,i') > \delta \cdot \min(t_i, t_{i'}) \ge \delta \cdot \alpha_j,$$

where the last inequality follows because j is contributing to both i and i' and hence $\min(t_i, t_{i'}) \ge \alpha_j$. By the above,

$$\sum_{i\in S} c(j,i) \ge \frac{\sum_{i,i'\in S} c(i,i')}{2s} \ge \frac{\sum_{i\ne i'\in S} \delta\cdot\alpha_j}{2s} = \delta \cdot \frac{s-1}{2} \cdot \alpha_j\,.$$

Hence,

$$\sum_{i\in S}(\alpha_j - c(j,i)) \leq \Big(s - \delta\cdot\frac{s-1}{2}\Big)\alpha_j = \Big(s(1-\tfrac{\delta}{2}) + \tfrac{\delta}{2}\Big)\alpha_j\,.$$

Now, since $\delta \geq 2$, the above upper bound is a non-increasing function of s. Therefore, since $s \geq 2$, we always have

$$\sum_{i\in S}(\alpha_j - c(j,i)) \leq (2-\tfrac{\delta}{2})\alpha_j\,. \tag{3.3.3}$$

We also know that $\alpha_j > c(j,i)$ for any $i \in S$. Therefore, $\alpha_j > c(j,\mathsf{IS})$ and, since $\delta \geq 2$:

$$(\tfrac{\delta}{2}-1)c(j,\mathsf{IS}) \leq (\tfrac{\delta}{2}-1)\alpha_j\,. \tag{3.3.4}$$

Combining Inequalities (3.3.3) and (3.3.4),

$$\sum_{i\in S}(\alpha_j - c(j,i)) + (\tfrac{\delta}{2}-1)c(j,\mathsf{IS}) \leq (2-\tfrac{\delta}{2})\alpha_j + (\tfrac{\delta}{2}-1)\alpha_j = \alpha_j\,.$$

We conclude the analysis of this case by rearranging the above inequality and recalling that $\rho_{\mathsf{mean}} \geq \frac{1}{\delta/2-1}$.

Case $s = 0$: Here, we claim that there exists a tight facility i such that

$$(1+\sqrt{\delta})\sqrt{\alpha_j} \geq d(j,i) + \sqrt{\delta t_i}\,. \tag{3.3.5}$$

To see that such a facility i exists, consider the witness $w(j)$ of j. By Claim 3.3.1, we have $\alpha_j \geq t_{w(j)}$ and since j has a tight edge to its witness $w(j)$, $\alpha_j \geq c(j,w(j) = d(j,w(j))^2$; or, equivalently, $\sqrt{\alpha_j} \geq \sqrt{t_{w(j)}}$ and $\sqrt{\alpha_j} \geq d(j,w(j))$ which implies that there is a tight facility, namely $w(j)$, satisfying (3.3.5).

Since IS is a maximal independent set of H, either $i \in \mathsf{IS}$, in which case $d(j,\mathsf{IS}) \leq d(j,i)$, or there is an $i' \in \mathsf{IS}$ such that the edge (i',i) is in H, in which case

$$d(j,\mathsf{IS}) \leq d(j,i) + d(i,i') \leq d(j,i) + \sqrt{\delta t_i}\,,$$

where the inequality follows from $d(i,i')^2 = c(i,i') \leq \delta\min(t_i,t_{i'})$ by the definition of H. In any case, we have by (3.3.5)

$$d(j,\mathsf{IS}) \leq (1+\sqrt{\delta})\sqrt{\alpha_j}\,.$$

Squaring both sides and recalling that $\rho_{\mathsf{mean}} \geq \frac{1}{(1+\sqrt{\delta})^2}$ completes the last case and the proof of the theorem.

□

k-Median objective in Euclidean metrics

We use a very similar approach as the one for k-means (in Euclidean metrics) to address the k-median objective in Euclidean metrics. In this section, we have $c(j,i) = d(j,i)$, i.e., the distances are *not* squared. Define,

$$\delta_{\mathsf{med}} = \sqrt{\tfrac{8}{3}} \qquad \text{and} \qquad \rho_{\mathsf{med}} = 1 + \sqrt{\tfrac{8}{3}} = \max\left(1 + \delta_{\mathsf{med}},\, 1/(\delta_{\mathsf{med}} - 1),\, 1/(\tfrac{3}{2}\delta_{\mathsf{med}} - 2)\right) .$$

We have $\delta_{\mathsf{med}} \approx 1.633$ and $\rho_{\mathsf{med}} \approx 2.633$.

Theorem 3.3.5. *Let d be a Euclidean metric on $\mathcal{D} \cup \mathcal{F}$ and suppose that $c(j,i) = d(j,i)$ for every $i \in \mathcal{F}$ and $j \in \mathcal{D}$. Then, for any $\lambda \geq 0$, Algorithm JV(δ_{med}) constructs a solution α to DUAL(λ) and returns a set IS of opened facilities such that*

$$\sum_{j \in S} c(j, \mathsf{IS}) \leq \rho_{med} \cdot \left(\sum_{j \in \mathcal{D}} \alpha_j - \lambda |\mathsf{IS}|\right) .$$

Proof. To simplify notation, we use δ instead of δ_{med} throughout the proof. Similarly to the proof of the previous theorem, we proceed by considering a single client j and prove

$$\frac{c(j, \mathsf{IS})}{\rho_{\mathsf{med}}} \leq \alpha_j - \sum_{i \in N(j) \cap \mathsf{IS}} (\alpha_j - c(j,i)) = \alpha_j - \sum_{i \in \mathsf{IS}} [\alpha_j - c(j,i)]^+ .$$

Let S denote $N(j) \cap \mathsf{IS}$ and $s = |S|$. We again proceed by case distinction on s. We first bound the number of cases.

Claim 3.3.6. *We have $s \leq 3$.*

Proof. Using the centroid property of squared distances in Euclidean space,

$$\sum_{j \in S} d(j,i)^2 \geq \frac{\sum_{i,i' \in S} d(i,i')^2}{2s} > \frac{\delta^2 (s-1) \alpha_j^2}{2} ,$$

where the last inequality follows from that $i, i' \in \mathsf{IS}$ are not adjacent in H so $d(i,i') > \delta \min(t_i, t_{i'})$ and $\min(t_i, t_{i'}) \geq \alpha_j$ since $i, i' \in S$. Since the left-hand side is upper bounded by $s\alpha_j^2$, we get $s > \frac{\delta^2(s-1)}{2}$. Therefore $s < \frac{\delta^2}{\delta^2 - 2} = 4$. □

We now proceed by considering the cases $s = 0, 1, 2, 3$.

Case $s = 0$: Consider the witness i_1 of j. We have $\alpha_j \geq t_{i_1}$ and $\alpha_j \geq c(j, i_1,)$. Since IS is a maximal independent set of H, either $i_1 \in \mathsf{IS}$, in which case $c(j, \mathsf{IS}) = d(j, \mathsf{IS}) \leq$

$d(j, i_1) \leq \alpha_j$, or there is an $i_2 \in \mathsf{IS}$ such that the edge (i_1, i_2) is in H, in which case

$$c(j, \mathsf{IS}) = d(j, \mathsf{IS}) \leq d(j, i_1) + d(i_1, i_2) \leq d(j, i_1) + \delta t_{i_1} \leq (1 + \delta)\alpha_j \,.$$

In any case, we have $c(j, \mathsf{IS})/\rho_{\mathsf{med}} \leq \alpha_j$ as required.

Case $s = 1$: If we let i^* be the unique facility in S,

$$\frac{c(j, \mathsf{IS})}{\rho_{\mathsf{med}}} \leq c(j, \mathsf{IS}) \leq c(j, i^*) = \alpha_j - (\alpha_j - c(j, i^*)) = \alpha_j - \sum_{i \in N(j) \cap \mathsf{IS}} (\alpha_j - c(j, i)) \,.$$

Case $s = 2$: Let $S = \{i_1^*, i_2^*\}$. We have

$$\begin{aligned} 2\alpha_j &= c(j, i_1^*) + c(j, i_2^*) + (\alpha_j - c(j, i_1^*)) + (\alpha_j - c(j, i_2^*)) \\ &\geq c(i_1^*, i_2^*) + \sum_{i \in S} (\alpha_j - c(j, i)) \\ &\geq \delta\alpha_j + \sum_{i \in S} (\alpha_j - c(j, i)) \,, \end{aligned}$$

where we used the triangle inequality and that $c(i_1^*, i_2^*) > \delta \min(t_{i_1^*}, t_{i_2^*}) \geq \delta\alpha_j$ since both i_1^* and i_2^* are in S and hence i_1^* and i_2^* are not adjacent in H. Rearranging the above inequality, we have

$$(\delta - 1)c(j, \mathsf{IS}) \leq \alpha_j - \sum_{i \in S} (\alpha_j - c(j, i)),$$

and the case follows because $\rho_{\mathsf{med}} \geq 1/(\delta - 1)$.

Case $s = 3$: Similar to the previous case,

$$\begin{aligned} 3\alpha_j &= \sum_{i \in S} c(j, i) + \sum_{i \in S} (\alpha_j - c(j, i)) \\ &\geq \frac{1}{2} \sum_{\{i, i'\} \subseteq \in S} c(i, i') + \sum_{i \in S} (\alpha_j - c(j, i)) \\ &\geq \frac{3\delta}{2} \cdot \alpha_j + \sum_{i \in S} (\alpha_j - c(j, i)) \,, \end{aligned}$$

using the triangle inequality. Rearranging the above inequality, we have

$$\left(\tfrac{3\delta}{2} - 2\right) c(j, \mathsf{IS}) \leq \alpha_j - \sum_{i \in S} (\alpha_j - c(j, i))$$

and the lemma follows because $\rho_{\mathsf{med}} \geq 1/(\frac{3\delta}{2} - 2)$.

□

3.4 Quasi-Polynomial Time Algorithm

In this section, we present a quasi-polynomial time algorithm that achieves the improved approximation guarantees explained in the previous section. We also introduce several of the ideas used in the polynomial time algorithm. Although the results obtained in this section are weaker (quasi-polynomial instead of polynomial), we believe that the easier quasi-polynomial algorithm serves as a good starting point before reading the more complex polynomial time algorithm. From now on, we concentrate on the k-means problem and we let $\rho = \rho_{\mathsf{mean}}$ denote the approximation guarantee and $\delta = \delta_{\mathsf{mean}}$ denote the parameter to our algorithm, where ρ_{mean} and δ_{mean} are defined as in Section 3.3.2 (it will be clear that the techniques presented here are easily applicable to the other considered objectives, as well). Throughout this section we fix $\epsilon > 0$ to be a small constant, and we assume for notational convenience and without loss of generality that $n \gg 1/\epsilon$. We also assume that the distances satisfy the following:

Lemma 3.4.1. *By losing a factor $(1 + 100/n^2)$ in the approximation guarantee, we can assume that the squared distance between any client j and any facility i satisfies: $1 \le d(i,j)^2 \le n^6$, where $n = |\mathcal{D}|$.*

The proof follows by standard discretization techniques and is presented in Section 3.10.

Our algorithm will produce a $(\rho + O(\epsilon))$-approximate solution. In the algorithm, we consider separately the two phases of the primal-dual algorithm from Section 3.3.2. Suppose that the first phase produces a set of values $\alpha = \{\alpha_j\}_{j \in \mathcal{D}}$ satisfying the following definition:

Definition 3.4.2. *A feasible solution α of $\mathsf{DUAL}(\lambda)$ is* good *if for every $j \in \mathcal{D}$ there exists a tight facility i such that $(1 + \sqrt{\delta} + \epsilon)\sqrt{\alpha_j} \ge d(j,i) + \sqrt{\delta t_i}$.*

Recall that for a dual solution α, t_i is defined to be the largest α-value out of all clients that are contributing to a facility i: $t_i = \max_{j \in N(i)} \alpha_j$ where $N(i) = \{j \in \mathcal{D} : \alpha_j - d(i,j)^2 > 0\}$.

As the condition of Definition 3.4.2 relaxes (3.3.5) by a tiny amount (regulated by ϵ), our analysis in Section 3.3 shows that as long as the first stage of the primal-dual algorithm produces an α that is good, the second stage will find a set of facilities IS such that $\sum_{j \in \mathcal{D}} d(j, \mathsf{IS})^2 = \sum_{j \in \mathcal{D}} c(j, \mathsf{IS}) \le (\rho + O(\epsilon))\big(\sum_{j \in \mathcal{D}} \alpha_j - \lambda |\mathsf{IS}|\big)$. If we could somehow find a value λ such that the second stage opened *exactly* k facilities, then we would obtain a $(\rho + O(\epsilon))$-approximation algorithm. In order to accomplish this, we first enumerate all potential values $\lambda = 0, 1 \cdot \epsilon_z, 2 \cdot \epsilon_z, \ldots, L \cdot \epsilon_z$, where ϵ_z is a small step size and L is large enough to guarantee that we eventually find a solution of size at most k (for a precise definition of L and ϵ_z, see (3.4.1) and (3.4.2)). Specifically, in Section 3.4.1, we give an algorithm that in time $n^{O(\epsilon^{-1} \log n)}$ generates a quasi-polynomial-length sequence of solutions $\alpha^{(0)}, \alpha^{(1)}, \ldots, \alpha^{(L)}$, where $\alpha^{(\ell)}$ is a good solution to $\mathsf{DUAL}(\ell \cdot \epsilon_z)$. We ensure that each consecutive set of values $\alpha^{(\ell)}, \alpha^{(\ell+1)}$ are *close* in the following sense:

Definition 3.4.3. *Two solutions α and α' are* close *if $|\alpha'_j - \alpha_j| \leq \frac{1}{n^2}$ for all $j \in \mathcal{D}$.*

Unfortunately, it might be the case that for a good solution $\alpha^{(\ell)}$ to $\mathsf{DUAL}(\lambda)$, the second stage of our algorithm opens more than k facilities, while for a good solution $\alpha^{(\ell+1)}$ to $\mathsf{DUAL}(\lambda + \epsilon_z)$, it opens fewer than k facilities. In order to obtain a solution that opens *exactly* k facilities, we must somehow interpolate between consecutive solutions in our sequence. In Section 3.4.2 we describe an algorithm that accomplishes this task. Specifically, for each pair of consecutive solutions $\alpha^{(\ell)}, \alpha^{(\ell+1)}$ we show that, since their α-values are nearly the same, we can control the way in which a maximal independent set in the associated conflict graphs changes. Formally, we show how to maintain a sequence of approximate integral solutions with cost bounded by $\alpha^{(\ell)}$ and $\alpha^{(\ell+1)}$, in which the number of open facilities decreases by at most one in each step. This ensures that some solution indeed opens exactly k facilities and it will be found in time $n^{O(\epsilon^{-1}\log n)}$.

3.4.1 Generating a sequence of close, good solutions

We first describe our procedure for generating a close sequence of good solutions. We select the following parameters

$$\epsilon_z = n^{-3-10\log_{1+\epsilon} n}\,, \tag{3.4.1}$$

$$L = 4n^7 \cdot \epsilon_z^{-1} = n^{O(\epsilon^{-1}\log n)}\,. \tag{3.4.2}$$

We now describe a procedure QuasiSweep that takes as input a good dual solution α^{in} of $\mathsf{DUAL}(\lambda)$ and outputs a good dual solution α^{out} of $\mathsf{DUAL}(\lambda + \epsilon_z)$ such that α^{in} and α^{out} are close. In order to generate the desired close sequence of solutions, we first define an initial solution for $\mathsf{DUAL}(0)$ by $\alpha_j = \min_{i\in\mathcal{F}} d(i,j)^2$ for $j \in \mathcal{D}$. Then, for $0 \leq \ell < L$, we call QuasiSweep with $\alpha^{\mathsf{in}} = \alpha^{(\ell)}$ to generate the next solution $\alpha^{(\ell+1)}$ in our sequence. We show that each $\alpha^{(\ell)}$ is a feasible dual solution of $\mathsf{DUAL}(\ell \cdot \epsilon_z)$ and that the following invariant holds throughout the generation of our sequence:

Invariant 1. *In every solution $\alpha^{(\ell)}$, ($0 \leq \ell \leq L$), every client $j \in \mathcal{D}$ has a tight edge to a tight facility $w(j) \in \mathcal{F}$ (its witness) such that $t_{w(j)} \leq (1+\epsilon)\alpha_j$.*

Note that this implies that each solution in our sequence is good. Indeed, consider a dual solution α that satisfies Invariant 1. Then, for any client j, we have some i ($= w(j)$) such that $\sqrt{\alpha_j} \geq d(i,j)$ (since j has a tight edge to $w(j)$) and $\sqrt{(1+\epsilon)\delta\alpha_j} \geq \sqrt{\delta t_i}$. Hence,

$$(1+\sqrt{\delta}+\epsilon)\sqrt{\alpha_j} \geq \left(1+\sqrt{(1+\epsilon)\delta}\right)\sqrt{\alpha_j} \geq d(i,j) + \sqrt{\delta t_i}\,,$$

and so α is good (here, for the first inequality we have used that $\sqrt{1+\epsilon} \leq 1 + \epsilon/2$ and $\sqrt{\delta} \leq 2$). We observe that our initial solution $\alpha^{(0)}$ has $t_i = 0$ for all $i \in \mathcal{F}$, and so Invariant 1 holds trivially. In our following analysis, we will show that each call to Sweep preserves Invariant 1.

We use the notion of *buckets* that partition the real line:

Definition 3.4.4. *For any value $v \in \mathbb{R}$, let*

$$B(v) = \begin{cases} 0 & \text{if } v < 1, \\ 1 + \lfloor \log_{1+\epsilon}(v) \rfloor & \text{if } v \geq 1. \end{cases}$$

We say that $B(v)$ is the index of the bucket containing v.

The buckets will be used to partition the α-values of the clients. Since α_j will always be at least 1 for each client (because its distance to any facility is at least 1), the definition gives the property that the α-values of any two clients j and j' in the same bucket differ by at most a factor of $1 + \epsilon$.

Description of QuasiSweep

We now formally describe the procedure QuasiSweep that, given the last previously generated solution α^{in} in our sequences produces a solution α^{out} returned next.

We initialize the algorithm by setting $\alpha_j = \alpha_j^{\mathsf{in}}$ for each $j \in \mathcal{D}$ and by increasing the opening prices of each facility from λ to $\lambda + \epsilon_z$. At this point, no facility is tight and therefore the solution α is not a good solution of $\mathsf{DUAL}(\lambda + \epsilon_z)$. We now describe how to modify α to obtain a solution α^{out} satisfying Invariant 1 (and hence into a good solution). The algorithm maintains a current set A of active clients and a current threshold θ. Initially, $A = \emptyset$, and $\theta = 0$. We slowly increase θ and whenever $\theta = \alpha_j$ for some client j, we add j to A. While $j \in A$, we increase α_j at the same rate as θ. We remove a client j from A, whenever the following occurs:

> j has a tight edge to some tight facility i with $B(\alpha_j) \geq B(t_i)$. In this case, we say that i is the *witness* of j.

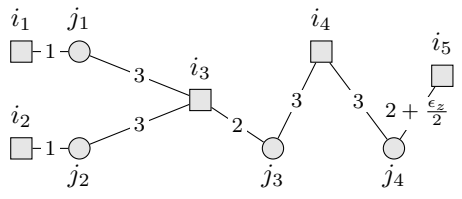
Note that if a client j satisfies this condition when it is added to A, then we remove j from A immediately after it is added. Then, α_j is not increased.

Increasing the α-values for clients in A, may cause the contributions to some facility i to exceed the opening cost $\lambda + \epsilon_z$. To prevent this from happening, we also decrease every value α_j with $B(\alpha_j) > B(\theta)$ at a rate of $|A|$ times the rate that θ is increasing. Observe

that while there exists any such $j \in N(i)$, the total contribution of the clients toward opening this i cannot increase, hence i cannot become tight. It follows that once any facility i becomes tight, $B(\alpha_j) \leq B(\theta)$ for every $j \in N(i)$ and so i is presently a witness for all clients $j \in N(i) \cap A$. As no such client will subsequently decrease, i remains tight until the end of QuasiSweep. Furthermore, any client j' that is added to A later will immediately be removed from A as soon as it has a tight edge to i. Thus, neither the total contribution to i nor t_i can subsequently increase, hence i remains a witness for all such j for the remainder of QuasiSweep.

We stop increasing θ once every client j has been added and removed from A. The procedure QuasiSweep then terminates and outputs $\alpha^{\mathsf{out}} = \alpha$. As we have just argued, the contributions to any tight facility can never increase, and every client that is removed from j will have a witness through the rest of QuasiSweep (in particular, in α^{out}). Thus, α^{out} is a feasible solution of $\mathsf{DUAL}(\lambda + \epsilon_z)$ in which every client has a witness, and as the values in the same bucket differ by at most a factor $1 + \epsilon$ (using that the α-value of each client is at least 1), the output of Sweep always satisfies Invariant 1.

This completes the description of QuasiSweep. For a small example of its execution see Figure 3.2. We now proceed to its analysis.



Input ($\lambda = 2$)

$\alpha^{\mathsf{in}}_{j_1} = \alpha^{\mathsf{in}}_{j_2} = 3$, $\alpha^{\mathsf{in}}_{j_3} = \alpha^{\mathsf{in}}_{j_4} = 4$.
Thus the tight facilities are i_1, i_2, i_3, i_4.

Output ($\lambda = 2 + \epsilon_z$)

$\alpha^{\mathsf{out}}_{j_1} = \alpha^{\mathsf{out}}_{j_2} = 3+\epsilon_z$, $\alpha^{\mathsf{out}}_{j_3} = 4-\epsilon_z$, $\alpha^{\mathsf{out}}_{j_4} = 4+\frac{3\epsilon_z}{2}$.
Thus the tight facilities are i_1, i_2, i_3, i_5.

Figure 3.2 – The instance has 4 clients and 5 facilities depicted by circles and squares, respectively. The number on an edge is the squared-distance of that edge and the squared-distances that are not depicted are all of value 5. Given the input solution α^{in} with $\lambda = 2$, QuasiSweep proceeds as follows. First the opening prices of facilities are increased to $2 + \epsilon_z$. Next the clients j_1, j_2 are added to the set A of active clients when the threshold $\theta = 3$. Then, until $\theta = 3 + \epsilon_z$, α_{j_1} and α_{j_2} increase at a uniform rate while the (significantly) larger dual values α_{j_3} and α_{j_4} are decreasing $|A| = 2$ times that rate. At the point $\theta = 3 + \epsilon_z$, both i_1 and i_2 become tight and the witnesses of j_1 and j_2 respectively. This causes these clients to be removed from A which stops their increase and the decrease of the larger values. When $\theta = 4 - 2\epsilon_z$, j_3 and j_4 are added to A and they start to increase at a uniform rate. Next, the facility i_3 becomes tight when $\theta = 4 - \epsilon_z$ and client j_3 is removed from A with i_3 as its witness. Finally, j_4 is removed from A when $\theta = 4 + 3\epsilon_z/2$ at which point i_5 becomes tight and its witness.

Closeness and running time

We begin by showing that QuasiSweep produces a close sequence of solutions.

Lemma 3.4.5. *For each client $j \in \mathcal{D}$, we have $|\alpha_j^{in} - \alpha_j^{out}| \leq 1/n^2$.*

Proof. We first note that the largest α-value at any time is at most $(\lambda + \epsilon_z) + n^6 \leq L\epsilon_z + n^6 = 4n^7 + n^6 \leq 5n^7$. This follows from the feasibility of α because, by Lemma 3.4.1, no squared-distance is larger than n^6 and the opening cost of any facility is at most $\lambda + \epsilon_z \leq L\epsilon_z$. Hence, $B(\alpha_j) \leq 1 + \lfloor \log_{1+\epsilon}(5n^7) \rfloor \leq 10 \log_{1+\epsilon}(n)$ for any client j and dual solution α.

We now prove the following claim:

Claim. *Any α_j can increase by at most $\epsilon_z n^{3b}$ while $B(\theta) \leq b$.*

Proof. The proof is by induction on $b = 0, 1, \ldots, 10 \log_{1+\epsilon}(n)$.

Base case $b = 0$: This case is trivially true because there are no clients j with $B(\alpha_j) = 0$, hence no clients can been added to A while $B(\theta) = 0$. Indeed, any client j had a tight edge to some facility in α^{in}, which by Lemma 3.4.1 implies $\alpha_j^{\text{in}} \geq 1$, and a client's α-value can decrease only while some smaller α-value is increasing.

Inductive step (assume true for $0, 1, \ldots, b-1$ and prove for b): Now, we suppose some α_j is increasing while $B(\theta) \leq b$. Note that we then must have $\alpha_j = \theta$. Let i be the witness of j in α^{in}, and let $N^{\text{in}}(i)$ be the set of clients contributing to i at this time. We further suppose that α_j is increased by at least ϵ_z while $B(\theta) \leq b$; otherwise the claim follows immediately, since $\epsilon_z \leq n^{3b}\epsilon_z$ for all $b \geq 0$.

First, suppose that $\alpha_j < \alpha_j^{\text{in}}$. Then, α_j was previously decreased. Moreover, since α_j was increased by at least ϵ_z while $B(\theta) \leq b$, we must have previously decreased α_j while $B(\alpha_j) \leq b$. In particular, at the last moment α_j was decreased, we must have had $B(\alpha_j) \leq b$, and since α_j was decreasing at this moment, we also had $B(\theta) < B(\alpha_j)$. Therefore α_j was decreased only while $B(\theta) < b$. Moreover, during this time, j's α-value was decreased at a rate of $|A|$, and so was decreased at most n times the amount that any other client was increased. By the induction hypothesis, this increase was at most $\epsilon_z \cdot n^{3b-3}$, and so α_j can be increased at most $\epsilon_z \cdot n^{3b-2}$ until $\alpha_j = \alpha_j^{\text{in}}$.

Now, we consider how much α_j may increase while $\alpha_j \geq \alpha_j^{\text{in}}$. For each $j' \in N^{\text{in}}(i)$ we must have initially had $B(\alpha_{j'}^{\text{in}}) \leq B(\alpha_j^{\text{in}}) \leq b$ since i is a witness for j in α^{in}, and so the α-value of j' was decreased by QuasiSweep only while $B(\theta) \leq b - 1$. Then, by the same argument as above, the α-value of any $j' \in N^{\text{in}}(i)$ can decrease at most $\epsilon_z n^{3b-2}$

throughout QuasiSweep. Thus, the total contribution to i from all $j' \neq j$ can decrease at most $(n-1) \cdot \epsilon_z \cdot n^{3b-2}$. After increasing α_j at most $(n-1) \cdot \epsilon_z \cdot n^{3b-2} + \epsilon_z$, i will again be tight. Moreover, at this moment, any client j' contributing to i was either already added to A (and potentially also removed) in which case $\alpha_{j'} \leq \theta = \alpha_j$ or it was not already added to A in which case $\alpha_{j'} \leq \alpha_{j'}^{\mathsf{in}}$. Hence, as $B(\alpha_{j'}^{\mathsf{in}}) \leq B(\alpha_j^{\mathsf{in}}) \leq B(\alpha_j)$, i is a witness for j, and j will be removed from A.

Altogether, the total amount α_j can increase while $B(\theta) \leq b$ is then the sum of these two increases, which is $\epsilon_z \cdot n^{3b-2} + (n-1) \cdot \epsilon_z \cdot n^{3b-2} + \epsilon_z \leq \epsilon_z \cdot n^{3b}$, as required. □

The claim immediately bounds the increase $\alpha_j^{\mathsf{out}} - \alpha_j^{\mathsf{in}}$ by $\frac{1}{n^3} \leq \frac{1}{n^2}$ as required (recall that $\epsilon_z = n^{-3-10\log_{1+\epsilon} n}$)). Moreover, as shown in the proof of the claim above, the α-value of every client decreases by no more than n times the maximum increase in the α-value of any client. Then, the desired bound $\frac{1}{n^2}$ on $\alpha_j^{\mathsf{in}} - \alpha_j^{\mathsf{out}}$ follows as well. □

Running time analysis The procedure QuasiSweep was presented in a continuous fashion. We now describe a polynomial time implementation of QuasiSweep. As presented, QuasiSweep maintains only the α-values of each client, the value of θ, and the set A of active clients. We suppose we are increasing θ at the speed of 1, so that the value of θ corresponds to the current time. Then, QuasiSweep changes each α-value at the speed of either 0, 1, or $-|A|$. Moreover, this speed does not change until one of the following events happens:

Event 1: Client j joins A: this can happen only if $\alpha_j = \theta$.

Event 2: θ changes buckets: this can only happen when θ has reached the border of a bucket.

Event 3: Facility i becomes tight: this can happen if (1) no client with a tight edge to i is decreasing, and (2) some client in A has a tight edge to i.

Event 4: Client j gains a tight edge to facility i: this can happen only if $j \in A$.

Event 5: Client $j \notin A$ changes bucket and enters the same bucket as θ: this can happen only if α_j is being decreased.

Note that we remove a client from A either immediately after it is added to A, at the time that some facility becomes tight, or at the time that it gains a tight edge to some (tight) facility. Therefore, we do not need to add an event for removing a client from A, since it only happens if one of the above events happen.

The polynomial time QuasiSweep now works as follows: In each step, we find the next time that any one of the above events happens, then increase/decrease each α-value

according to its current speed to obtain a new set of values at this time. Then, we update θ, A, and our set of speeds and continue. We need to show how we can efficiently compute the next event that happens, and also we need to prove that the number of such events is polynomial. In what follows, we compute the time until each event above happens, assuming that it is the next event that happens. Then the next event that actually happens is the event with the minimum such time (breaking ties arbitrarily).

We now consider each of the above events in turn:

- The time until the Event 1 may happen next is $\min_{\alpha_j > \theta} \alpha_j - \theta$. Also we have exactly n occurrences of this event.

- The time until Event 2 may happen next is the difference between θ and the border of the next bucket, i.e., $B_{next} - \theta$, where $B_{next} = (1+\epsilon)^{B(\theta)}$. We have at most $O(\epsilon^{-1}\log(n))$ such events.

- For Event 3, if some client with a tight edge to i is decreasing then (non-tight) facility i cannot become tight (due to the choice of the speed of decrease). If no decreasing client has a tight edge to facility i, then the time that i may become tight is

  $$\frac{(\lambda + \epsilon_z) - \sum_{j \in \mathcal{D}} [\alpha_j - d(i,j)^2]^+}{|N(i) \cap A|}.$$

  Notice that the numerator is the current slack of facility i and the denominator is the speed at which this slack decreases. Moreover, there are at most $m = |\mathcal{F}|$ such events, since if a facility becomes tight, it will stay tight (as we discussed in our description of QuasiSweep).

- The time until Event 4 may happen for some edge (j,i) is $d(j,i)^2 - \alpha_j$ if $\alpha_j < d(j,i)^2$, and there are at most nm such events, since if an edge becomes tight, it remains tight afterwards.

- Finally, Event 5 may happen only for those clients j with $B(\alpha_j) > B(\theta)$. For any such j, the time until Event 5 happens is $(\alpha_j - B_{next})/|A|$. This event can happen also at most n times, since once $B(\alpha_j) = B(\theta)$, j is is no longer decreased. Note that when α_j is decreasing, we consider it to change buckets at the moment that it lies on the lower border of it's current bucket (i.e., at the moment that $1 + \log_{1+\epsilon} \alpha_j = B_{next}$). It is easy to verify that still $(1+\epsilon)\alpha_j \leq \alpha_{j'}$ for any j and j' placed in the same bucket by this rule.

From the above, it is clear that the number of events are polynomial, and also that the next event can be computed in polynomial time.

We conclude the analysis of this section by noting that, as SWEEP is repeated $L = n^{O(\epsilon^{-1}\log n)}$ times, the total running time for producing the sequence $\alpha^{(0)}, \alpha^{(1)}, \ldots, \alpha^{(L)}$ is $n^{O(\epsilon^{-1}\log n)}$.

3.4.2 Finding a solution of size k

In this section we describe our algorithm for finding a solution of k facilities given a close sequence $\alpha^{(0)}, \alpha^{(1)} \ldots, \alpha^{(L)}$, where $\alpha^{(\ell)}$ is a good solution to $\mathsf{DUAL}(\epsilon_z \cdot \ell)$.

We associate with each dual solution $\alpha^{(\ell)}$ a client-facility graph and a conflict graph that are defined in exactly the same way as in Section 3.3.1: that is, the graph $G^{(\ell)}$ is a bipartite graph with all of $\mathcal{D}$ on one side and every tight facility in $\alpha^{(\ell)}$ on the other and $G^{(\ell)}$ contains the edge (j, i) if and only if $\alpha_j^{(\ell)} > c(j, i)$. Given each $G^{(\ell)}$, recall that $H^{(\ell)}$ is then a graph consisting of the facilities present in $G^{(\ell)}$, which contains an edge (i, i') if and only if $c(i, i') < \delta \min(t_i^{(\ell)}, t_{i'}^{(\ell)})$, where for each i, we have $t_i^{(\ell)} = \max\{\alpha_j^{(\ell)} : \alpha_j^{(\ell)} > c(j, i)\}$ (and again we adopt the convention that $t_i^{(\ell)} = 0$ if $\alpha_j^{(\ell)} \leq c(j, i)$ for all $j \in \mathcal{D}$). Thus, we have a sequence $G^{(0)}, \ldots, G^{(L)}$ of client-facility graphs and a sequence $H^{(0)}, \ldots, H^{(L)}$ of conflict graphs obtained from our sequence of dual solutions. The main goal of this section is to give a corresponding sequence of maximal independent sets of the conflict graphs so that the size of the solution (independent set) never decreases by more than 1 in this sequence. Unfortunately, this is not quite possible. Instead, starting with a maximal independent set $\mathsf{IS}^{(\ell)}$ of $H^{(\ell)}$, we shall slowly transform it into a maximal independent set $\mathsf{IS}^{(\ell+1)}$ of $H^{(\ell+1)}$ by considering maximal independent sets in a sequence of polynomially many intermediate conflict graphs $H^{(\ell)} = H^{(\ell,0)}, H^{(\ell,1)}, \ldots, H^{(\ell,p_\ell)} = H^{(\ell+1)}$. We shall refer to these independent sets as $\mathsf{IS}^{(\ell)} = \mathsf{IS}^{(\ell,0)}, \mathsf{IS}^{(\ell,1)}, \ldots, \mathsf{IS}^{(\ell,p_\ell)} = \mathsf{IS}^{(\ell+1)}$. This interpolation will allow us to ensure that the size of our independent set decreases by at most 1 throughout this sequence. It follows that at some point we find a solution IS of size exactly k: On the one hand, since $H^{(0)}$ contains all facilities and no edges we have $\mathsf{IS}^{(0)} = \mathcal{F}$, which by assumption is strictly greater than k. On the other hand, we must have $|\mathsf{IS}^{(L)}| \leq 1$. Indeed, as $\alpha^{(L)}$ is a good dual solution of $\mathsf{DUAL}(L\epsilon_z) = \mathsf{DUAL}(4n^7)$, we claim $H^{(L)}$ is a clique. First, note that any tight facility i has $t_i \geq \frac{L\epsilon_z}{n} = 4n^6$ which means that all clients have a tight edge to i when i becomes tight (as the maximum squared facility-client distance is n^6 by Lemma 3.4.1). Second, any two facilities i, i' have $d(i, i')^2 \leq 4n^6$ using the triangle inequality and facility-client distance bound. Combining these two insights, we can see that $H^{(L)}$ is a clique and so its independent set has size at most 1.

It remains to describe and analyze the procedure QUASIGRAPHUPDATE that will perform the interpolation between two conflict graphs $H^{(\ell)}$ and $H^{(\ell+1)}$ when given a maximal independent set $\mathsf{IS}^{(\ell)}$ of $H^{(\ell)}$ so that $|\mathsf{IS}^{(\ell)}| > k$. We run this procedure at most L times starting with $H^{(0)}, H^{(1)}$, and $\mathsf{IS}^{(0)} = \mathcal{F}$ until we find a solution of size k.

Description of QuasiGraphUpdate

Denote the input by $H^{(\ell)}, H^{(\ell+1)}$, and $\mathsf{IS}^{(\ell)}$ (the maximal independent set of $H^{(\ell)}$ of size greater than k). Although we are interested in producing a sequence of conflict graphs, it will be helpful to think of a process that alters some "hybrid" client-facility graph G, then uses G and the corresponding opening times t to construct a new conflict graph H after each alteration. To ease the description of this process, we duplicate each facility that appears both in $G^{(\ell)}$ and $G^{(\ell+1)}$ so as to ensure that these sets are disjoint. Let $\mathcal{V}^{(\ell)}$ and $\mathcal{V}^{(\ell+1)}$ denote the (now disjoint) sets of facilities in $G^{(\ell)}$ and $G^{(\ell+1)}$, respectively. Note that the duplication of facilities does not alter the solution space of the considered instance, as one may assume that at most one facility is opened at each location. Note that our algorithm will also satisfy this property, since $d(i, i')^2 = 0$ for any pair of co-located facilities i, i'.

Initially, we let G be the client-facility graph with bipartition $\mathcal{D}$ and $\mathcal{V}^{(\ell)} \cup \mathcal{V}^{(\ell+1)}$ that has an edge from client j to facility $i \in \mathcal{V}^{(\ell)}$ if (j, i) is present in $G^{(\ell)}$ and to $i \in \mathcal{V}^{(\ell+1)}$ if (j, i) is present in $G^{(\ell+1)}$. The opening time t_i of facility i is now naturally set to $t_i^{(\ell)}$ if $i \in \mathcal{V}^{(\ell)}$ and to $t_i^{(\ell+1)}$ if $i \in \mathcal{V}^{(\ell+1)}$. Informally, G is the union of the two client-facility graphs $G^{(\ell)}$ and $G^{(\ell+1)}$ where the client vertices are shared (see Figure 3.3). We then generate[2] the conflict graph $H^{(\ell,1)}$ from G and t. As the induced subgraph of $H^{(\ell,1)}$ on vertex set $\mathcal{V}^{\ell}$ equals $H^{(\ell)} = H^{(\ell,0)}$, we have that $\mathsf{IS}^{(\ell)}$ is also an independent set of $H^{(\ell,1)}$. We obtain a maximal independent set $\mathsf{IS}^{(\ell,1)}$ of $H^{(\ell,1)}$ by greedily extending $\mathsf{IS}^{(\ell)}$. Clearly, the independent set can only increase so we still have $|\mathsf{IS}^{(\ell,1)}| > k$.

To produce the remaining sequence, we iteratively perform changes, but construct and output a new conflict graph and maximal independent set after *each* such change. Specifically, we remove from G each facility $i \in \mathcal{V}^{(\ell)}$, one by one. At the end of the procedure (after $|\mathcal{V}^{(\ell)}|$ many steps), we have $G = G^{(\ell+1)}$ and so $H^{(\ell,p_\ell)} = H^{(\ell+1)}$. Note that at each step, our modification to G results in removing a single facility i from the associated conflict graph. Thus, if $\mathsf{IS}^{(\ell,s)}$ is an independent set in $H^{(\ell,s)}$ before a modification, then $\mathsf{IS}^{(\ell,s)} \setminus \{i\}$ is an independent set in $H^{(\ell,s+1)}$. We obtain a maximal independent set $\mathsf{IS}^{(\ell,s+1)}$ of $H^{(\ell,s+1)}$ by greedily extending $\mathsf{IS}^{(\ell,s)} \setminus \{i\}$. Then, for each step s, we have $|\mathsf{IS}^{(\ell,s+1)}| \geq |\mathsf{IS}^{(\ell,s)}| - 1$, as required.

Analysis

The total running time is $n^{O(\epsilon^{-1} \log n)}$ since the number of steps L (and the number of dual solutions in our sequence) is $n^{O(\epsilon^{-1} \log n)}$ and each step runs in polynomial time since it involves the construction of at most $O(|\mathcal{F}|)$ conflict graphs and maximal independent sets.

[2]Recall that a conflict graph is defined in terms of a client-facility graph G and t: the vertices are the facilities in G, and two facilities i and i' are adjacent if there is some client j that is adjacent to both of them in G and $d(i, i')^2 \leq \delta \min(t_i, t_{i'})$.

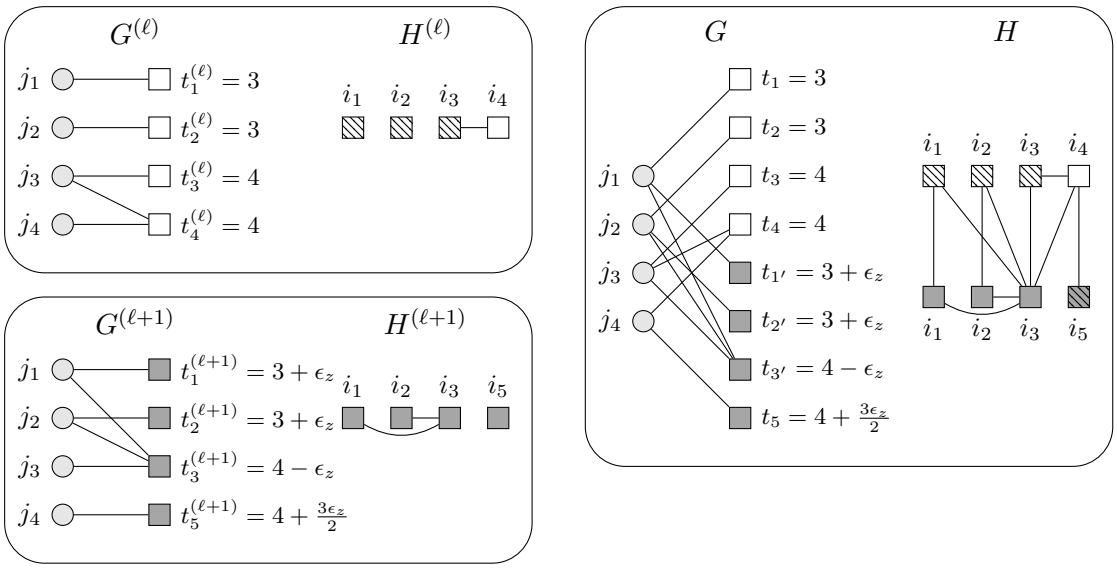


Figure 3.3 – An example of the "hybrid" client-facility graph and associated conflict graph used by QuasiGraphUpdate. $G^{(\ell)}$ and $G^{(\ell+1)}$ are the client-facility graphs of α^{in} and α^{out} of Figure 3.2. Next to the facilities, we have written the facility times (t_i's) of those solutions. As the squared-distance between any two facilities is 5 in the example of Figure 3.2, one can see that any two facilities with a common neighbor in the client-facility graph will be adjacent in the conflict graph. G is the "hybrid" client-facility graph of $G^{(\ell)}$ and $G^{(\ell+1)}$. When H is formed, we extend the given maximal independent set $\mathsf{IS}^{(\ell)}$ of $H^{(\ell)}$ to form a maximal independent set of H. The facilities in the relevant independent sets are indicated with stripes.

We proceed to analyze the approximation guarantee. Consider the first time that we produce some maximal independent set IS of size exactly k. Suppose that when this happened, we were moving between two solutions $\alpha^{(\ell)}$ and $\alpha^{(\ell+1)}$, i.e., $\mathsf{IS} = \mathsf{IS}^{(\ell,s)}$ is a maximal independent set of $H^{(\ell,s)}$ for some $1 \leq s \leq p_\ell$. That we may assume that $s \geq 1$ follows from $|\mathsf{IS}^{(0)}| > k$ and $\mathsf{IS}^{(\ell-1,p_\ell)} = \mathsf{IS}^{(\ell)} = \mathsf{IS}^{(\ell,0)}$ (recall that IS was selected to be the *first* independent set of size k).

To ease notation, we let $H = H^{(\ell,s)}$ and denote by G the "hybrid" client-facility graph that generated H. In order to analyze the cost of IS, let us form a hybrid solution α by setting $\alpha_j = \min(\alpha_j^{(\ell)}, \alpha_j^{(\ell+1)})$ for each client $j \in \mathcal{D}$. Note that $\alpha \leq \alpha^{(\ell)}$ is a feasible solution of $\mathsf{DUAL}(\lambda)$ where $\lambda = \ell \cdot \epsilon_z$ and, since $\alpha^{(\ell)}$ and $\alpha^{(\ell+1)}$ are close, $\alpha_j \geq \alpha_j^{(\ell)} - \frac{1}{n^2}$ and $\alpha_j \geq \alpha_j^{(\ell+1)} - \frac{1}{n^2}$. For each client j, we define a set of facilities $S_j \subseteq \mathsf{IS}$ to which j contributes, as follows. For all $i \in \mathsf{IS}$, we have $i \in S_j$ if $\alpha_j > d(j,i)^2$. Note that S_j is a subset of j's neighborhood in G and therefore

$$\alpha_j = \min(\alpha_j^{(\ell)}, \alpha_j^{(\ell+1)}) \leq t_i = \begin{cases} t_i^{(\ell)} & \text{if } i \in \mathcal{V}^{(\ell)} \\ t_i^{(\ell+1)} & \text{if } i \in \mathcal{V}^{(\ell+1)} \end{cases} \qquad \text{for all } i \in S_j.$$

Using the fact that $\alpha^{(\ell+1)}$ is a good dual solution, we can bound the total service cost of all clients in the integral solution IS. Let us first proceed separately for those clients with $|S_j| > 0$. Let $\mathcal{D}_0 = \{j \in \mathcal{D} : |S_j| = 0\}$, and $\mathcal{D}_{>0} = \mathcal{D} \setminus \mathcal{D}_0$. We remark that the analysis is now very similar to the proof of Theorem 3.3.4. We define $\beta_{ij} = [\alpha_j - d(i,j)^2]^+$ and similarly $\beta_{ij}^{(\ell)} = [\alpha_j^{(\ell)} - d(i,j)^2]^+$ and $\beta_{ij}^{(\ell+1)} = [\alpha_j^{(\ell+1)} - d(i,j)^2]^+$.

Lemma 3.4.6. *For any $j \in \mathcal{D}_{>0}$, $d(j,\mathsf{IS})^2 \leq \rho \cdot \left(\alpha_j - \sum_{i \in S_j} \beta_{ij}\right)$.*

Proof. Consider some $j \in \mathcal{D}_{>0}$ and first suppose that $|S_j| = 1$. Then, if we let $S_j = \{i\}$, $\alpha_j = \beta_{ij} + d(j,i)^2 \geq \beta_{ij} + d(j,\mathsf{IS})^2$ just as in "Case $s = 1$" of Theorem 3.3.4. Next, suppose that $|S_j| = s > 1$. In other words, j is contributing to multiple facilities in IS. We then have $\alpha_j - \sum_{i \in S_j} \beta_{ij} \geq \frac{1}{\rho} d(j,\mathsf{IS})^2$ by the exact same arguments as in "Case $s > 1$" of Theorem 3.3.4 using that we also in this case have $\alpha_j \leq \min(t_i, t_{i'})$ for any two facilities $i, i' \in S_j$. □

Next, we bound the total service cost of all those clients that do not contribute to any facility in IS. The proof is very similar to "Case $s = 0$" in the proof of Theorem 3.3.4.

Lemma 3.4.7. *For every $j \in \mathcal{D}_0$, $d(j,\mathsf{IS})^2 \leq (1+5\epsilon)\rho \cdot \alpha_j$.*

Proof. Consider some client $j \in \mathcal{D}_0$, and let $i \in \mathcal{V}^{(\ell+1)}$ be a tight facility so that

$$(1+\sqrt{\delta}+\epsilon)\sqrt{\alpha_j^{(\ell+1)}} \geq d(j,i) + \sqrt{\delta t_i^{(\ell+1)}}\,.$$

Such a facility i is guaranteed to exist because $\alpha^{(\ell+1)}$ is a good dual solution. Furthermore, note that i is present in H since H contains all facilities in $\mathcal{V}^{(\ell+1)}$. By definition $t_i = t_i^{(\ell+1)}$ and, as all α-values are at least 1 (by the preprocessing of Lemma 3.4.1), $(1+\frac{1}{n^2})\alpha_j \geq \alpha_j + 1/n^2 \geq \alpha_j^{(\ell+1)}$. Hence, the above inequality implies

$$(1+\tfrac{1}{n^2})(1+\sqrt{\delta}+\epsilon)\sqrt{\alpha_j} \geq d(j,i) + \sqrt{\delta t_i}\,.$$

Note the similarity of this inequality with that of (3.3.5) and the proof is now identical to "Case $s = 0$" of Theorem 3.3.4.

Indeed, since IS is a maximal independent set of H, either $i \in \mathsf{IS}$, in which case $d(j,\mathsf{IS}) \leq d(j,i)$, or there is a $i' \in \mathsf{IS}$ such that the edge (i',i) is in H, in which case

$$d(j,\mathsf{IS}) \leq d(j,i) + d(i,i') \leq d(j,i) + \sqrt{\delta t_i}\,,$$

where the inequality follows from $d(i,i')^2 \leq \delta \min(t_i, t_{i'})$ by the definition of H. In any case, we have (using $n \gg 1/\epsilon$)

$$d(j,\mathsf{IS}) \leq (1+\tfrac{1}{n^2})(1+\sqrt{\delta}+\epsilon)\sqrt{\alpha_j} \leq (1+2\epsilon)(1+\sqrt{\delta})\sqrt{\alpha_j}\,.$$

Squaring both sides and recalling that $\rho \geq \frac{1}{(1+\sqrt{\delta})^2}$ and that ϵ is a small constant so $(1+2\epsilon)^2 \leq (1+5\epsilon)$ completes the proof of the lemma. □

One difference compared to the analysis in Section 3.3.2 is that not all opened facilities are fully paid for. However, they are almost paid for:

Lemma 3.4.8. *For any $i \in \mathsf{IS}$, $\sum_{j\in\mathcal{D}} \beta_{ij} \geq \lambda - \frac{1}{n}$.*

Proof. If $i \in \mathcal{V}^{(\ell+1)}$, then it is a tight facility with respect to $\alpha^{(\ell+1)}$, i.e., $\sum_{j\in\mathcal{D}} \beta_{ij}^{(\ell+1)} = \lambda + \epsilon_z$. Similarly, if $i \in \mathcal{V}^{(\ell)}$ then $\sum_{j\in\mathcal{D}} \beta_{ij}^{(\ell)} = \lambda$. Now since $\alpha_j \geq \max(\alpha_j^{(\ell+1)}, \alpha_j^{(\ell)}) - \frac{1}{n^2}$ for every client j,

$$\sum_{j\in\mathcal{D}} \beta_{ij} \geq \sum_{j\in\mathcal{D}} \left(\max(\beta_{ij}^{(\ell+1)}, \beta_{ij}^{(\ell)}) - \tfrac{1}{n^2}\right) \geq \lambda - \tfrac{1}{n}\,. \qquad \square$$

We now combine the above lemmas to bound the approximation guarantee of the found solution. Recall that OPT_k denotes the optimum value of the standard LP-relaxation (see Section 3.2).

Theorem 3.4.9. $\sum_{j\in\mathcal{D}} d(j,\mathsf{IS})^2 \leq (1+6\epsilon)\rho \cdot \mathrm{OPT}_k$.

Proof. From Lemmas 3.4.6 and 3.4.7 we have:

$$\sum_{j\in\mathcal{D}} d(j,\mathsf{IS})^2 \leq (1+5\epsilon)\rho \sum_{j\in\mathcal{D}} \left(\alpha_j - \sum_{i\in S_j} \beta_{ij}\right)\,.$$

By Lemma 3.4.8 (note that by definition, $\sum_{i\in\mathsf{IS}} \beta_{ij} = \sum_{i\in S_j} \beta_{ij}$),

$$\sum_{j\in\mathcal{D}} \left(\alpha_j - \sum_{i\in S_j} \beta_{ij}\right) \leq \sum_{j\in\mathcal{D}} \alpha_j - |\mathsf{IS}| \left(\lambda - \tfrac{1}{n}\right) = \sum_{j\in\mathcal{D}} \alpha_j - k\cdot\lambda + \tfrac{k}{n} \leq \mathrm{OPT}_k + 1\,,$$

where the last inequality follows from $k \leq n$ and, as α is a feasible solution to $\mathsf{DUAL}(\lambda)$, $\sum_{j\in\mathcal{D}} \alpha_j - k\cdot\lambda \leq \mathrm{OPT}_k$. The statement now follows from $\mathrm{OPT}_k \geq \sum_{j\in\mathcal{D}} \min_{i\in\mathcal{F}} d(i,j)^2 \geq n$ and $n \gg 1/\epsilon$.

□

We have thus proved that our quasi-polynomial algorithm produces a $(\rho + O(\epsilon))$-approximate solution which implies Theorem 3.1.1. The quasi-polynomial algorithms for the other considered problems are the same except for the selection of δ and ρ, and that in the k-median problem the connection costs are the (non-squared) distances.

3.5 A Polynomial Time Approximation Algorithm

We now show how to obtain a polynomial-time algorithm, building on the ideas presented in the previous section. As in Section 3.4, we focus exclusively on the k-means problem, and let $\delta = \delta_{\mathsf{mean}} \approx 2.3146$ and $\rho = \rho_{\mathsf{mean}} = (1+\sqrt{\delta})^2 \approx 6.3574$, and assume that the squared-distances between clients and facilities are in $[1, n^6]$ by Lemma 3.4.1. Additionally, we choose ϵ and γ to be suitably small constants with $0 < \gamma \ll \epsilon \ll 1$, and for notational convenience we assume without loss of generality that $n \gg 1/\gamma$.

Similarly to Section 3.4.1, we give an algorithm for generating a close sequence of feasible solutions to $\mathsf{DUAL}(\lambda)$, and then show how to use this sequence to generate a sequence of integral solutions that must contain some solution of size exactly k. Here, however, we ensure that our sequence of feasible solutions is of polynomial length. In order to accomplish this, we must relax some of the requirements in our definition of a good solution (Definition 3.4.2).

First, rather than requiring that every facility have opening cost λ, we instead allow each facility i to have its own price in $z_i \in \{\lambda, \lambda + \frac{1}{n}\}$ (Condition 1 of Definition 3.5.1). For each $\alpha^{(\ell)}$, our algorithm will produce an associated set of facility prices $z^{(\ell)} = \{z_i^{(\ell)}\}_{i \in \mathcal{F}}$. For any such $(\alpha^{(\ell)}, z^{(\ell)})$, we define $\beta_{ij}^{(\ell)} = [\alpha_j^{(\ell)} - d(j,i)^2]^+$ and $N^{(\ell)}(i) = \{j \ : \ \beta_{ij}^{(\ell)} > 0\}$, as before. However, we now say that a facility i is *tight* in $(\alpha^{(\ell)}, z^{(\ell)})$ if $\sum_{j \in \mathcal{D}} \beta_{ij}^{(\ell)} = z_i^{(\ell)}$. That is, we consider a facility i tight once its (possibly unique) price z_i is paid in the dual. Intuitively, if all the facility prices z_i are *almost* the same, we can still carry out our analysis, and obtain a $(\rho + O(\epsilon))$-approximation.

Second, we shall designate a set of *special* facilities $\mathcal{F}_{\mathsf{S}} \subseteq \mathcal{F}$ that we shall open, *even if they are not tight*. To each special facility $i \in \mathcal{F}_{\mathsf{S}}$ we assign a set of special clients $\mathcal{D}_{\mathsf{S}}(i) \subseteq \mathcal{D}$ that are allowed to pay for i. Then, for each $i \in \mathcal{F}_{\mathsf{S}}$, we define the time $\tau_i = \max_{j \in N(i) \cap \mathcal{D}_{\mathsf{S}}(i)} \alpha_j$, while for each $i \in \mathcal{F} \setminus \mathcal{F}_{\mathsf{S}}$ we set $\tau_i = t_i = \max_{j \in N(i)} \alpha_j$. Again, we adopt the convention that $\tau_i = 0$ if $N(i) \cap \mathcal{D}_{\mathsf{S}}(i) = \emptyset$ for $i \in \mathcal{F}_{\mathsf{S}}$ or $N(i) = \emptyset$ for $i \in \mathcal{F} \setminus \mathcal{F}_{\mathsf{S}}$. Although a facility in $\mathcal{F}_{\mathsf{S}}$ is not necessarily tight, we shall require that the total of all payments to such facilities by special clients is *almost* equal to $\lambda|\mathcal{F}_{\mathsf{S}}|$ (Condition 3 of Definition 3.5.1). That is, on average, each facility of $\mathcal{F}_{\mathsf{S}}$ is almost tight.

Finally, given the times τ_i, we shall not require that *every* client j have some tight or special facility i such that $(1+\sqrt{\delta}+10\epsilon)\sqrt{\alpha_j} \geq d(j,i) + \sqrt{\delta\tau_i}$. Specifically, we shall allow some small set of *bad* clients $\mathcal{D}_{\mathsf{B}}$ to instead satisfy a weaker inequality $6\sqrt{\alpha_j} \geq d(j,i) + \sqrt{\delta\tau_i}$ for some tight or special facility i. Such clients will have a higher service cost, so we require that their total contribution to the cost of an optimal solution is small (Condition 2 of Definition 3.5.1).

Combining the above, we have the following definition.

Definition 3.5.1. *Consider a tuple $(\alpha, z, \mathcal{F}_S, \mathcal{D}_S)$ where $\alpha \in \mathbb{R}^{\mathcal{D}}, z \in \mathbb{R}^{\mathcal{F}}$, $\mathcal{F}_S \subseteq \mathcal{F}$ is a*

set of special facilities, and $\mathcal{D}_S : \mathcal{F}_S \to 2^{\mathcal{D}}$ *is a function assigning each special facility* i *a set of special clients* $\mathcal{D}_S(i)$*. We say that this tuple is roundable for* λ *(or* λ*-roundable) if* α *is a feasible solution of* $\mathsf{DUAL}(\lambda + \frac{1}{n})$*, and:*

1. *For all* $i \in \mathcal{F}$, $\lambda \leq z_i \leq \lambda + \frac{1}{n}$.
2. *There exists a subset* $\mathcal{D}_B$ *of clients so that for all* $j \in \mathcal{D}$ *there is a facility* $w(j)$ *that is either tight or in* $\mathcal{F}_S$ *and:*
   (a) $(1 + \sqrt{\delta} + 10\epsilon)^2 \alpha_j \geq \left(d(j, w(j)) + \sqrt{\delta \cdot \tau_{w(j)}}\right)^2$ *for all* $j \in \mathcal{D} \setminus \mathcal{D}_B$.
   (b) $36\gamma \cdot \mathrm{OPT}_k \geq \sum_{j \in \mathcal{D}_B} \left(d(j, w(j)) + \sqrt{\delta \cdot \tau_{w(j)}}\right)^2$,
3. $\sum_{i \in \mathcal{F}_S} \sum_{j \in \mathcal{D}_S(i)} \beta_{ij} \geq \lambda |\mathcal{F}_S| - \gamma \cdot \mathrm{OPT}_k$ *and* $|\mathcal{F}_S| \leq n$.

Observe that any λ-roundable solution with $\mathcal{F}_\mathsf{S} = \emptyset$, and $\mathcal{D}_\mathsf{B} = \emptyset$ is essentially a good solution for $\mathsf{DUAL}(\lambda + \frac{1}{n})$ (as defined for the quasi-polynomial algorithm in Section 3.4) except that the opening costs of the facilities are allowed to vary slightly. We shall also say that (α, z) is roundable if $(\alpha, z, \emptyset, \mathcal{D}_\mathsf{S})$ is roundable.

An overview of our polynomial time algorithm is shown in Algorithm 1. The algorithm maintains a current base price λ and a current roundable solution $\mathcal{S}^{(0)} = (\alpha^{(0)}, z^{(0)}, \mathcal{F}_\mathsf{S}^{(0)}, \mathcal{D}_\mathsf{S}^{(0)})$ for λ, as well as a corresponding integral solution $\mathsf{IS}^{(0)}$. As in the quasi-polynomial algorithm, we shall enumerate a sequence $0, 1 \cdot \epsilon_z, 2 \cdot \epsilon_z, \ldots, L \cdot \epsilon_z$ of base prices λ, where now $\epsilon_z = n^{-O(1)}$ and, as before we define $L = 4n^7 \cdot \epsilon_z^{-1}$. Here, however we increase facility prices from λ to $\lambda + \epsilon_z$ *one-by-one* using an auxiliary procedure RaisePrice, which takes as input a fractional dual solution $\alpha^{(0)}$, a set of prices $z^{(0)}$, a current integral solution $\mathsf{IS}^{(0)}$, and a facility i. RaisePrice increases the price of facility i, then outputs a close sequence of roundable solutions $\mathcal{S}^{(1)} = (\alpha^{(1)}, z^{(1)}, \mathcal{F}_\mathsf{S}^{(1)}, \mathcal{D}_\mathsf{S}^{(1)}), \ldots, \mathcal{S}^{(q)} = (\alpha^{(q)}, z^{(q)}, \mathcal{F}_\mathsf{S}^{(q)}, \mathcal{D}_\mathsf{S}^{(q)})$, each having $z_i^{(\ell)} = z_i^{(0)} + \epsilon_z$ and $z_{i'}^{(\ell)} = z_{i'}^{(0)}$ for all $i' \neq i$. Note that in addition to increasing the facility prices one-by-one, we now generate a *sequence* of solutions for each individual price increase.

Initially, we set $\lambda \leftarrow 0$ and then initialize $\mathcal{S}^{(0)}$ by setting $z_i^{(0)} \leftarrow 0$ for all $i \in \mathcal{F}$ and $\mathcal{F}_\mathsf{S} = \emptyset$ (observe that $\mathcal{D}_\mathsf{S}$ is then an empty function), and constructing $\alpha^{(0)}$ as follows. We set $\alpha_j = 0$ for all $j \in \mathcal{D}$ and then increase all α_j at a uniform rate. We stop increasing a value α_j whenever j gains a tight edge to some facility $i \in \mathcal{F}$ or $2\sqrt{\alpha_j} \geq d(j, j') + 6\sqrt{\alpha_{j'}}$ for some $j' \in \mathcal{D}$ (the rationale behind this choice will be made clear in Section 3.7). Finally, we initialize our current integral solution $\mathsf{IS}^{(0)} = \mathcal{F}$.

As long as an integral solution of size k has not yet been produced, Algorithm 1 iterates through each facility $i \in \mathcal{F}$, calling RaisePrice to raise z_i by $\epsilon_z < 1/n$. The sequences that are produced are used to obtain a sequence of integral solutions in which the size of each solution decreases by at most 1. This is done by using a second procedure,

GraphUpdate, which is very similar to the procedure QuasiGraphUpdate described in the previous section. Note that raise price always increases a single facility i's price by $\epsilon_z < 1/n$, and does not increase z_i further until all other facility prices have also been increased by ϵ_z. Thus, each in every pair of consecutive solutions $\mathcal{S}^{(\ell)}, \mathcal{S}^{(\ell+1)}$ considered by GraphUpdate in line 7, every price $z_i \in \{\lambda, \lambda + \epsilon_z\}$ and so both solutions are λ-roundable (for the same value λ). We describe our auxiliary procedures GraphUpdate and RaisePrice in the next sections. Note that initially $|\mathsf{IS}^{(0)}| = |\mathcal{F}|$ and, by the same reasoning as in Section 3.4.2, once $\lambda = L \cdot \epsilon_z = 4n^7$ we must have $|\mathsf{IS}^{(0)}| = 1$. Thus, at some intermediate point, we will indeed find some solution IS of size k.

Algorithm 1: Polynomial time $(\rho_{\mathsf{mean}} + O(\epsilon))$-approximation algorithm for k-means

1 Initialize $\mathcal{S}^{(0)} = (\alpha^{(0)}, z^{(0)}, \mathcal{F}_{\mathsf{S}}^{(0)}, \mathcal{D}_{\mathsf{S}}^{(0)})$ as described in our discussion above
2 $\lambda \leftarrow 0, \mathsf{IS}^{(0)} \leftarrow \mathcal{F}$
3 **for** $\lambda = 0,\ 1 \cdot \epsilon_z,\ 2 \cdot \epsilon_z,\ \ldots,\ L \cdot \epsilon_z$ **do**
  `/* Raise the price of each facility` i `to` $z_i^{(0)} + \epsilon_z = \lambda + \epsilon_z$ `*/`
4 **foreach** $i \in \mathcal{F}$ **do**
5 Call RaisePrice$(\alpha^{(0)}, z^{(0)}, \mathsf{IS}^{(0)}, i)$ to produce a sequence $\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(q)}$ of λ-roundable solutions
  `/* Move through this sequence, constructing integral solutions */`
6 **for** $\ell = 0$ **to** $q - 1$ **do**
7 Call GraphUpdate$(\mathcal{S}^{(\ell)}, \mathcal{S}^{(\ell+1)}, \mathsf{IS}^{(\ell)})$ to produce a sequence $\mathsf{IS}^{(\ell,0)}, \ldots, \mathsf{IS}^{(\ell,p_\ell)}$
8 **if** $|\mathsf{IS}^{(\ell,r)}| = k$ for some $\mathsf{IS}^{(\ell,r)}$ in this sequence **then return** $\mathsf{IS}^{(\ell,r)}$
9 **else** $\mathsf{IS}^{(\ell+1)} \leftarrow \mathsf{IS}^{(\ell,p_\ell)}$
  `/* After each price increase, update current solutions */`
10 $\mathcal{S}^{(0)} \leftarrow \mathcal{S}^{(q)}$, $\mathsf{IS}^{(0)} \leftarrow \mathsf{IS}^{(q)}$
  `/* All prices have been increased. Continue to the next base price` λ `*/`

Algorithm 1 executes $L = 4n^7 \cdot \epsilon_z^{-1}$ base price increases, each of which performs $|\mathcal{F}|$ calls to RaisePrice. In order to show that Algorithm 1 runs it polynomial time, it is sufficient to show that each call to RaisePrice and GraphUpdate produces a polynomial length sequence in polynomial time. In the next sections, we describe these procedures in more detail and show that they run in polynomial time. In addition, we show that RaisePrice produces a sequence of roundable solutions (Proposition 3.8.18) that are close (Proposition 3.8.10). In Section 3.6, we show that given these solutions, GraphUpdate finds a $(\rho + 1000\epsilon)$-approximate solution (Theorem 3.6.4).[3] This implies our main theorem:

Theorem 3.5.2. *For any $\epsilon > 0$, there is a $(\rho + \epsilon)$-approximation algorithm for k-means.*

[3]We remark that we have chosen to first describe GraphUpdate as that procedure is very similar to QuasiGraphUpdate in the quasi-polynomial algorithm whereas RaisePrices is more complex.

3.6 Opening a Set of Exactly k Facilities in a Close, Roundable Sequence

In this section, we describe our algorithm GraphUpdate for interpolating between two close roundable solutions $\mathcal{S}^{(\ell)}$ and $\mathcal{S}^{(\ell+1)}$ starting with a maximal independent set $\mathsf{IS}^{(\ell)}$ of the conflict graph[4] $H^{(\ell)}$ of $\mathcal{S}^{(\ell)}$. The goal of this procedure is the same as that of QuasiGraphUpdate explained in Section 3.4.2: we maintain a sequence of maximal independent sets in appropriately constructed conflict graphs so that the size of the independent set never decreases by more than 1, and the last solution is a maximal independent set of the conflict graph $H^{(\ell+1)}$ of $\mathcal{S}^{(\ell+1)}$. Similar to Section 3.4.2, we use a "hybrid" client-facility graph to generate our conflict graph in each step of our procedure. The only difference is that we need to slightly generalize the definition of a client-facility graph to incorporate the concept of roundable solutions.

Client-facility and conflict graphs of roundable solutions. We define the *client-facility* graph G of a roundable solution $\mathcal{S} = (\alpha, z, \mathcal{F}_\mathsf{s}, \mathcal{D}_\mathsf{s})$ as in Section 3.3.1 with the following two changes: First, recall that we now consider a facility i *tight* if and only if $\sum_{j \in N(i)} \beta_{ij} = z_i$. Second, we shall additionally add every facility $i \in \mathcal{F}_\mathsf{s}$ to G, but place an edge between each $i \in \mathcal{F}_\mathsf{s}$ and $j \in \mathcal{D}$ only if $j \in N(i) \cap \mathcal{D}_\mathsf{s}(i)$. Intuitively, we treat special facilities $i \in \mathcal{F}_\mathsf{s}$ essentially the same as tight facilities, except only those clients in $N(i) \cap \mathcal{D}_\mathsf{s}(i)$ are considered to be paying for i.

Formally, let $\mathcal{V}$ denote the set of all tight facilities or special facilities with respect to $\mathcal{S}$. Then, G is a bipartite graph on $\mathcal{D}$ and $\mathcal{V}$ that contains an edge (i, j) if and only if $i \in \mathcal{V} \setminus \mathcal{F}_\mathsf{s}$ and $j \in N(i)$ or $i \in \mathcal{F}_\mathsf{s}$ and $j \in N(i) \cap \mathcal{D}_\mathsf{s}(i)$. As before, we assign an *opening time* τ_i to each $i \in \mathcal{V}$. For $i \in \mathcal{V} \setminus \mathcal{F}_\mathsf{s}$, $\tau_i = t_i = \max_{j \in N(i)} \alpha_j$, and for $i \in \mathcal{F}_\mathsf{s}$, $\tau_i = \max_{j \in N(i) \cap \mathcal{D}_\mathsf{s}(i)} \alpha_j$. In other words, τ_i equals the maximum α_j over all clients j such that (j, i) is an edge in G (in the case that there is no such edge, we adopt the convention that $\tau_i = 0$).

Given a client facility graph G, and a set of opening times τ, we construct the corresponding *conflict graph* H in the same way as in Section 3.3: the vertex set of H is the set of all facilities appearing in G and we place an edge between two facilities i and i' in H if and only if there is some $j \in \mathcal{D}$ such that both (j, i) and (j, i') are present in G and $d(i, i')^2 \leq \delta \cdot \min(\tau_i, \tau_{i'})$. Notice that this coincides with the definition in Section 3.3 when the set of special facilities is empty. In particular, the initial independent set $\mathcal{F}$ is a maximal independent set of the conflict graph associated to the initial solution (which has all facilities and no edges). Then, as in each iteration the last constructed independent set by GraphUpdate is given as input in the next call (see Algorithm 1), we maintain the property that the input independent set $\mathsf{IS}^{(\ell)}$ is a maximal independent

[4] Below, we slightly generalize the definition of client-facility and conflict graphs in Section 3.3.1 to that of roundable solutions.

set of the conflict graph of $\mathcal{S}^{(\ell)}$.

Description of GraphUpdate**.** Our algorithm now proceeds in the exact same way as QuasiGraphUpdate in Section 3.4.2. A short description is repeated here for convenience. Let $G^{(\ell)}, \tau^{(\ell)}$ and $G^{(\ell+1)}, \tau^{(\ell+1)}$ be the client-facility graphs and times associated with $\mathcal{S}^{(\ell)}$ and $\mathcal{S}^{(\ell+1)}$, respectively. Furthermore, let $H^{(\ell)}$ and $H^{(\ell+1)}$ be the conflict graphs generated by $G^{(\ell)}, \tau^{(\ell)}$ and $G^{(\ell+1)}, \tau^{(\ell+1)}$. Recall that the input to GraphUpdate is $\mathcal{S}^{(\ell)}, \mathcal{S}^{(\ell+1)}$ and a maximal independent set $\mathsf{IS}^{(\ell)}$ of $H^{(\ell)}$.

Define the "hybrid" client-facility graph G as the union of $G^{(\ell)}$ and $G^{(\ell+1)}$ where the client vertices are shared and the facilities are duplicated if necessary so as to make sure that the facilities of $G^{(\ell)}$ and $G^{(\ell+1)}$ are disjoint. The opening times are defined by

$$\tau_i = \begin{cases} \tau_i^{(\ell)} & \text{if } i \in \mathcal{V}^{(\ell)} \\ \tau_i^{(\ell+1)} & \text{if } i \in \mathcal{V}^{(\ell+1)} \end{cases},$$

where $\mathcal{V}^{(\ell)}$ and $\mathcal{V}^{(\ell+1)}$ denote the (disjoint) sets of facilities in $G^{(\ell)}$ and $G^{(\ell+1)}$, respectively. We then generate the conflict graph $H^{(\ell,1)}$ from G and τ. As the induced subgraph of $H^{(\ell,1)}$ on vertex set $\mathcal{V}^{\ell}$ equals $H^{(\ell)} = H^{(\ell,0)}$, we have that the given maximal independent set $\mathsf{IS}^{(\ell)}$ of $H^{(\ell)}$ is also an independent set of $H^{(\ell,1)}$. We obtain a maximal independent set $\mathsf{IS}^{(\ell,1)}$ of $H^{(\ell,1)}$ by greedily extending $\mathsf{IS}^{(\ell)}$. We then obtain the remaining conflict graphs and independent sets by removing from G each facility $i \in \mathcal{V}^{(\ell)}$, one by one. After each step we generate the associated conflict graph and we greedily extend the previous independent set (with i potentially removed) so as to obtain a maximal independent set in the updated conflict graph. This results, as in Section 3.4.2, in the sequence $H^{(\ell)} = H^{(\ell,0)}, H^{(\ell,1)}, \ldots, H^{(\ell,p_\ell)} = H^{(\ell+1)}$ of $|\mathcal{V}^{(\ell)}| + 2$ many conflict graphs and a sequence $\mathsf{IS}^{(\ell)} = \mathsf{IS}^{(\ell,0)}, \mathsf{IS}^{(\ell,1)}, \ldots, \mathsf{IS}^{(\ell,p_\ell)} = \mathsf{IS}^{(\ell+1)}$ of associated maximal independent sets so that $|\mathsf{IS}^{(\ell,s)}| \geq |\mathsf{IS}^{(\ell,s-1)}| - 1$ for any $s = 1, \ldots, p_\ell$. The output of GraphUpdate is this sequence of independent sets.

3.6.1 Analysis

GraphUpdate clearly runs in polynomial time since the number of steps is polynomial and each step requires only the construction of a conflict graph and greedily maintaining a maximal independent set.

We proceed to analyze the approximation guarantee. In comparison to Section 3.4.2, our analysis here is slightly more involved because it is with respect to roundable solutions instead of good solutions. In addition, we prove that *all* independent sets constructed in Algorithm 1 (by calls to GraphUpdate) of size at least k have small connection cost. Specifically, we show that any constructed independent set IS with $|\mathsf{IS}| \geq k$ has

$\sum_{j\in\mathcal{D}} d(j,\mathsf{IS})^2 \le (\rho + O(\epsilon))\mathrm{OPT}_k$.

First note that the initial independent set $\mathsf{IS}^{(0)}$ of Algorithm 1 contains all facilities and hence $\sum_{j\in\mathcal{D}} d(j,\mathsf{IS}^{(0)})^2 \le \mathrm{OPT}_k$. All other independent sets are constructed by calls to GraphUpdate. Consider one such independent set IS with $|\mathsf{IS}| \ge k$ and consider the first time this independent set was constructed. Suppose that when this happened, we were moving between two solutions $\mathcal{S}^{(\ell)}$ and $\mathcal{S}^{(\ell+1)}$ that are roundable for the same λ. Then, $\mathsf{IS} = \mathsf{IS}^{(\ell,s)}$ for some step $s \ge 1$ of GraphUpdate. We may assume $s \ge 1$ because $\mathsf{IS}^{(\ell,0)} = \mathsf{IS}^{(\ell)}$ was constructed in the previous call to GraphUpdate (or it equals the initial independent set). Let G and τ be the client-facility graph and the opening times that generated the conflict graph $H = H^{(\ell,s)}$ in which $\mathsf{IS} = \mathsf{IS}^{(\ell,s)}$ is a maximal independent set. Also note that we may assume, without loss of generality, that $|\mathsf{IS}| \le n$. Otherwise, we can reduce the size of IS since the connection cost of IS equals that of $\bigcup_{j\in\mathcal{D}}\{\arg\min_{i\in\mathsf{IS}} d(j,i)\}$.

Similar to Section 3.4.2, we analyze the cost of IS with respect to a hybrid solution α obtained by setting $\alpha_j = \min(\alpha_j^{(\ell)}, \alpha_j^{(\ell+1)})$ for each client $j \in \mathcal{D}$. The following observations and concepts are also very similar to the ones in that section. We remark that α is a feasible solution of $\mathsf{DUAL}(\lambda + \frac{1}{n})$ and, since $\alpha^{(\ell)}$ and $\alpha^{(\ell+1)}$ are close, $\alpha_j \ge \alpha_j^{(\ell)} - \frac{1}{n^2}$ and $\alpha_j \ge \alpha_j^{(\ell+1)} - \frac{1}{n^2}$. For each client j, we define a set of facilities $S_j \subseteq \mathsf{IS}$ to which j contributes, as follows. For all $i \in \mathsf{IS}$, we have $i \in S_j$ if $\alpha_j > d(j,i)^2$ and (j,i) is an edge in G. Note that S_j is a subset of j's neighborhood in G and therefore

$$\alpha_j = \min(\alpha_j^{(\ell)}, \alpha_j^{(\ell+1)}) \le \tau_i \qquad \text{for all } i \in S_j. \tag{3.6.1}$$

Using the fact that $\mathcal{S}^{(\ell+1)}$ is roundable, we can bound the total service cost of all clients in the integral solution IS. Let us first proceed separately for those clients with $|S_j| > 0$. Let $\mathcal{D}_0 = \{j \in \mathcal{D} : |S_j| = 0\}$, and $\mathcal{D}_{>0} = \mathcal{D} \setminus \mathcal{D}_0$. The following lemma is identical to Lemma 3.4.6 and its proof is therefore omitted.

Lemma 3.6.1. *For any $j \in \mathcal{D}_{>0}$, $d(j,\mathsf{IS})^2 \le \rho \cdot \left(\alpha_j - \sum_{i\in S_j} \beta_{ij}\right)$.*

Next, we bound the total service cost of all those clients that do not contribute to any facility in IS. The proof is very similar to that of Lemma 3.4.7 except that we also need to handle the bad clients in $\mathcal{D}_{\mathsf{B}}$.

Lemma 3.6.2. $\sum_{j\in\mathcal{D}_0} d(j,\mathsf{IS})^2 \le (\rho + 200\epsilon)\sum_{j\in\mathcal{D}_0}\alpha_j + 36\gamma\cdot\mathrm{OPT}_k$.

Proof. Consider some client $j \in \mathcal{D}_0$, and let $w(j) \in \mathcal{V}^{(\ell+1)}$ be the tight or special facility for j corresponding to the roundable solution $\mathcal{S}^{(\ell+1)}$. Note that $w(j)$ is present in H (since $H = H^{(\ell,s)}$ with $s \ge 1$ contains all facilities in $\mathcal{V}^{(\ell+1)}$) and $\tau_{w(j)} = \tau_{w(j)}^{(\ell+1)}$ by definition. Thus, since IS is a maximal independent set of H, either $w(j) \in \mathsf{IS}$, in which

case $d(j,\mathsf{IS}) \le d(j,w(j))$, or there must be some other facility $i \in \mathsf{IS}$ such that H contains the edge $(i,w(j))$, in which case

$$d(j,\mathsf{IS}) \le d(j,w(j)) + d(w(j),i) \le d(j,w(j)) + \sqrt{\delta\tau_i}\,,$$

where the inequality follows from the fact that i and $w(j)$ are adjacent in H and thus $d(i,w(j))^2 \le \delta\min(\tau_i,\tau_{i'})$ by the definition of H. In any case, we have $d(j,\mathsf{IS}) \le d(j,w(j)) + \sqrt{\delta\tau_i}$ with $\tau_i = \tau_i^{(\ell+1)}$, and so:

$$\begin{aligned}\sum_{j\in\mathcal{D}_0} d(j,\mathsf{IS})^2 &= \sum_{j\in\mathcal{D}_0\setminus\mathcal{D}_\mathsf{B}} d(j,\mathsf{IS})^2 + \sum_{j\in\mathcal{D}_0\cap\mathcal{D}_\mathsf{B}} d(j,\mathsf{IS})^2 \\ &\le \sum_{j\in\mathcal{D}_0\setminus\mathcal{D}_\mathsf{B}} \left(d(j,w(j)) + \sqrt{\delta\cdot\tau_{w(j)}^{(\ell+1)}}\right)^2 + \sum_{j\in\mathcal{D}_\mathsf{B}} \left(d(j,w(j)) + \sqrt{\delta\cdot\tau_{w(j)}^{(\ell+1)}}\right)^2 \\ &\le \sum_{j\in\mathcal{D}_0\setminus\mathcal{D}_\mathsf{B}} (1+\sqrt{\delta}+10\epsilon)^2\cdot\alpha_j^{(\ell+1)} + 36\gamma\cdot\mathrm{OPT}_k\,,\end{aligned}$$

where the final inequality follows from the fact that $\mathcal{S}^{(\ell+1)}$ is roundable. The statement now follows since $\alpha_j^{(\ell+1)} \le \alpha_j + 1/n^2 \le (1+1/n^2)\alpha_j$ for all $j\in\mathcal{D}$, $\epsilon \le 1$, $\sqrt{\delta}\le 2$, and $\rho \ge (1+\sqrt{\delta})^2$. □

We now bound the contributions to the opened facilities as in Lemma 3.4.8 except that we also need to handle the special facilities.

Lemma 3.6.3. *For any $i \in \mathsf{IS}\setminus(\mathcal{F}_\mathsf{S}^{(\ell)}\cup\mathcal{F}_\mathsf{S}^{(\ell+1)})$, we have $\sum_{j\in\mathcal{D}}\beta_{ij} \ge \lambda - \frac{1}{n}$ and for any $i\in\mathcal{F}_\mathsf{S}^{(x)}$ for some $x\in\{\ell,\ell+1\}$, we have $\sum_{j\in\mathcal{D}_\mathsf{S}^{(x)}(i)}\beta_{ij} \ge \left(\sum_{j\in\mathcal{D}_\mathsf{S}^{(x)}(i)}\beta_{ij}^{(x)}\right) - \frac{1}{n}$.*

Proof. For the first bound, consider a facility $i\in\mathsf{IS}\setminus(\mathcal{F}_\mathsf{S}^{(\ell)}\cup\mathcal{F}_\mathsf{S}^{(\ell+1)})$ and let $x\in\{\ell,\ell+1\}$ be such that $i\in\mathcal{V}^{(x)}$. Then i is a tight facility with respect to $(\alpha^{(x)},z^{(x)})$, i.e., $\sum_{j\in\mathcal{D}}\beta_{ij}^{(x)} = z_i^{(x)}$. As $\mathcal{S}^{(x)}$ is roundable for λ, we have $z_i^{(x)}\ge\lambda$. Moreover, $\alpha_j \ge \alpha_j^{(x)} - \frac{1}{n^2}$ for every client j, and so

$$\sum_{j\in\mathcal{D}}\beta_{ij} \ge \sum_{j\in\mathcal{D}}\left(\beta_{ij}^{(x)} - \tfrac{1}{n^2}\right) \ge \lambda - \tfrac{1}{n}\,.$$

Now consider a special facility $i\in\mathcal{F}_\mathsf{S}^{(x)}$ for some $x\in\{\ell,\ell+1\}$. Then, by again using that $\alpha_j \ge \alpha_j^{(x)} - \frac{1}{n^2}$ for every client j,

$$\sum_{j\in\mathcal{D}_\mathsf{S}^{(x)}(i)}\beta_{ij} \ge \sum_{j\in\mathcal{D}_\mathsf{S}^{(x)}(i)}\left(\beta_{ij}^{(x)} - \tfrac{1}{n^2}\right)\,,$$

and the lemma follows since $|\mathcal{D}_\mathsf{S}^{(x)}(i)| \le |\mathcal{D}| = n$. □

We are now ready to prove our main result, which bounds the connection cost of IS in terms of OPT_k as desired. The proof is very similar to the proof of Theorem 3.4.9.

Theorem 3.6.4. *For any* IS *produced by* GraphUpdate *with* $|\mathsf{IS}| \geq k$,

$$\sum_{j \in \mathcal{D}} d(j, \mathsf{IS})^2 \leq (\rho + 1000\epsilon) \cdot \mathrm{OPT}_k.$$

Proof. From Lemmas 3.6.1 and 3.6.2 we have:

$$\sum_{j \in \mathcal{D}} d(j, \mathsf{IS})^2 \leq (\rho + 200\epsilon) \left(\sum_{j \in \mathcal{D}} \alpha_j - \sum_{i \in S_j} \beta_{ij} \right) + 36\gamma \cdot \mathrm{OPT}_k \,. \tag{3.6.2}$$

Note that by definition, if $i \notin \mathcal{F}_{\mathsf{S}}^{(\ell)} \cup \mathcal{F}_{\mathsf{S}}^{(\ell+1)}$ then $\sum_{j \in \mathcal{D}} \beta_{ij} = \sum_{j : i \in S_j} \beta_{ij}$ and if $i \in \mathcal{F}_{\mathsf{S}}^{(x)}$ then $\sum_{j \in \mathcal{D}_{\mathsf{S}}^{(x)}(i)} \beta_{ij} = \sum_{j : i \in S_j} \beta_{ij}$. Also, recall that by our construction of H, $\mathcal{F}_{\mathsf{S}}^{(\ell)}$ and $\mathcal{F}_{\mathsf{S}}^{(\ell+1)}$ are distinct. Thus, by Lemma 3.6.3,

$$\begin{aligned}
\sum_{j \in \mathcal{D}} \left(\alpha_j - \sum_{i \in S_j} \beta_{ij} \right) &\leq \sum_{j \in \mathcal{D}} \alpha_j - |\mathsf{IS} \setminus (\mathcal{F}_{\mathsf{S}}^{(\ell)} \cup \mathcal{F}_{\mathsf{S}}^{(\ell+1)})| \left(\lambda - \tfrac{1}{n} \right) \\
&\quad - \sum_{x \in \{\ell, \ell+1\}} \sum_{i \in \mathcal{F}_{\mathsf{S}}^{(x)} \cap \mathsf{IS}} \left(\sum_{j \in \mathcal{D}_{\mathsf{S}}^{(x)}(i)} \beta_{ij}^{(x)} - \tfrac{1}{n} \right) \\
&\leq \sum_{j \in \mathcal{D}} \alpha_j - |\mathsf{IS} \setminus (\mathcal{F}_{\mathsf{S}}^{(\ell)} \cup \mathcal{F}_{\mathsf{S}}^{(\ell+1)})| \lambda \\
&\quad - \sum_{x \in \{\ell, \ell+1\}} \sum_{i \in \mathcal{F}_{\mathsf{S}}^{(x)} \cap \mathsf{IS}} \sum_{j \in \mathcal{D}_{\mathsf{S}}^{(x)}(i)} \beta_{ij}^{(x)} + \frac{|\mathsf{IS}|}{n} \,.
\end{aligned}$$

Since $\mathcal{S}^{(x)}$ is roundable for $x \in \{\ell, \ell+1\}$, we have $\sum_{i \in \mathcal{F}_{\mathsf{S}}^{(x)}} \sum_{j \in \mathcal{D}_{\mathsf{S}}^{(x)}(i)} \beta_{ij}^{(x)} \geq \lambda |\mathcal{F}_{\mathsf{S}}^{(x)}| - \gamma \cdot \mathrm{OPT}_k$. Moreover, as $\alpha^{(x)}$ is a feasible solution of $\mathsf{DUAL}(\lambda + \frac{1}{n})$, we have that $\sum_{j \in \mathcal{D}_{\mathsf{S}}^{(x)}(i)} \beta_{ij}^{(x)} \leq \lambda + \frac{1}{n}$ for any $i \in \mathcal{F}_{\mathsf{S}}^{(x)}$. Therefore,

$$\begin{aligned}
\sum_{i \in \mathcal{F}_{\mathsf{S}}^{(x)} \cap \mathsf{IS}} \sum_{j \in \mathcal{D}_{\mathsf{S}}^{(x)}(i)} \beta_{ij}^{(x)} &\geq \lambda |\mathcal{F}_{\mathsf{S}}^{(x)} \cap \mathsf{IS}| - \tfrac{|\mathcal{F}_{\mathsf{S}}^{(x)} \setminus \mathsf{IS}|}{n} - \gamma \cdot \mathrm{OPT}_k \\
&\geq \lambda |\mathcal{F}_{\mathsf{S}}^{(x)} \cap \mathsf{IS}| - 2\gamma \cdot \mathrm{OPT}_k \,.
\end{aligned}$$

where for the final inequality we use that $|\mathcal{F}_{\mathsf{S}}^{(x)}| \leq n \leq \mathrm{OPT}_k$, which follows from Definition 3.5.1, the fact that any client has distance at least 1 to its closest facility, and

$1/n \ll \gamma$. Combining this with the above inequalities yields

$$\begin{aligned}\sum_{j\in\mathcal{D}}\left(\alpha_j - \sum_{i\in S_j}\beta_{ij}\right) &\le \sum_{j\in\mathcal{D}}\alpha_j - |\mathsf{IS}|\lambda + 4\gamma\cdot\mathrm{OPT}_k + \frac{|\mathsf{IS}|}{n}\\ &= \sum_{j\in\mathcal{D}}\alpha_j - |\mathsf{IS}|(\lambda+\tfrac{1}{n}) + 4\gamma\cdot\mathrm{OPT}_k + \frac{2|\mathsf{IS}|}{n}\\ &\le \mathrm{OPT}_k + 4\gamma\cdot\mathrm{OPT}_k + \frac{2|\mathsf{IS}|}{n} \le (1+5\gamma)\mathrm{OPT}_k\,,\end{aligned}$$

where we in the penultimate inequality used that α is a feasible solution to $\mathsf{DUAL}(\lambda+\frac{1}{n})$ and $|\mathsf{IS}| \ge k$, therefore $\sum_{j\in\mathcal{D}}\alpha_j - |\mathsf{IS}|(\lambda+\frac{1}{n}) \le \sum_{j\in\mathcal{D}}\alpha_j - k(\lambda+\frac{1}{n}) \le \mathrm{OPT}_k$; and, in the last inequality, we used that $\gamma\cdot\mathrm{OPT}_k \ge \gamma n \ge 2$ and the assumption that $|\mathsf{IS}| \le n$.

We conclude the proof by substituting this bound in (3.6.2):

$$\sum_{j\in\mathcal{D}} d(j,\mathsf{IS})^2 \le (\rho+200\epsilon)(1+5\gamma)\mathrm{OPT}_k + 36\gamma\cdot\mathrm{OPT}_k \le (\rho+1000\epsilon)\mathrm{OPT}_k\,. \qquad \square$$

3.7 The Algorithm RaisePrice

In this section, we give the details of the algorithm RaisePrice, which is responsible for raising facility prices and generating sequences of roundable solutions in Algorithm 1. It is based on similar insights as used in the quasi-polynomial algorithm described in Section 3.4. Let us first provide a high-level overview of our approach. Recall that in our analysis of that procedure, changing the values α_j in some bucket b by ϵ_z roughly required changing the values in bucket $b+1$ by up to $n\epsilon_z$. Because there were $\Omega(\log(n))$ buckets, the total change in the last bucket was potentially $\epsilon_z n^{\Omega(\log n)}$, and so to obtain a close sequence of α values, we required $\epsilon_z = n^{-\Omega(\log n)}$ in that section. Here, we reduce the dependence on n by changing the way in which we increase the opening price z. As in the quasi-polynomial procedure, our algorithm repeatedly increases the opening cost of every facility from λ to $\lambda+\epsilon_z$, for some appropriate small increment $\epsilon_z = n^{-O(1)} < \epsilon$. However, instead of performing each such increase for every facility at once, we instead increase only a single facility's price at a time. Each such increase will still cause some clients to become unsatisfied (or undecided as we shall call them), and so we must repair the solution. In contrast to the quasi-polynomial procedure, RaisePrice repairs the solution over a series of *stages*. We show this will result in a *polynomial length* sequence of close, roundable solutions.

Notation: Throughout this section, we let z_i denote the current price for a facility $i \in \mathcal{F}$, where always $z_i \in \{\lambda, \lambda+\epsilon_z\}$. We shall now say that i is *tight* if $\sum_{j\in\mathcal{D}}\beta_{ij} = z_i$, where as before for a solution α, we use β_{ij} as a shorthand for $[\alpha_j - d(j,i)^2]^+$. It will also be convenient to denote $\sqrt{\alpha_j}$ by $\bar{\alpha}_j$. Note that $\beta_{ij} > 0$, if and only if $\bar{\alpha}_j > d(j,i)$.

As in the quasi-polynomial procedure, we shall divide the range of possible values for α_j into buckets: we define $B(v) = 1 + \lfloor \log_{1+\epsilon} v \rfloor$ for any $v \geq 1$ and $B(v) = 0$, for all $v \leq 1$.

To control the number of undecided (unsatisfied) clients, it will be important to control the way clients may be increased and decreased throughout our algorithm. To accomplish this, we shall not insist that every client has some tight witness in every vector α that we produce (in contrast to Invariant 1 in the quasi-polynomial algorithm). Rather, we shall consider several different types of clients:

- *witnessed clients* j have a tight edge to some tight facility i with $B(\alpha_j) \geq B(t_i)$. In this case, we say that i is a *witness* for j. Note that if i is a witness for j we necessarily have $(1+\epsilon)\alpha_j \geq t_i$.[5]
- *stopped clients* j have

$$2\bar{\alpha}_j \geq d(j, j') + 6\bar{\alpha}_{j'} \tag{3.7.1}$$

  for some other client j'. In this case, we say that j' *stops* j. Note that if j' stops j, we necessarily have $\bar{\alpha}_j \geq 3\bar{\alpha}_{j'}$ and so $\alpha_j \geq 9\alpha_{j'}$.
- *undecided* clients j are neither witnessed nor stopped.

Let us additionally call any client that is witnessed or stopped *decided*. Note that the sets of witnessed and stopped clients are not necessarily disjoint. However, we have the following lemma, which follows directly from the triangle inequality and our definitions:

Lemma 3.7.1. *Suppose that j is stopped. Then j must be stopped by some j' that is* not *stopped.*

Proof. We proceed by induction over clients j in non-decreasing order of α_j. First, note that the client j with smallest value α_j cannot be stopped. For the general case, suppose that j is stopped by some j_1. Then, $\alpha_{j_1} < \alpha_j$. If j_1 is stopped, then by the induction hypothesis it must be stopped by some j_2 that is not stopped. Then, we have $2\bar{\alpha}_j \geq d(j, j_1) + 6\bar{\alpha}_{j_1}$, and $2\bar{\alpha}_{j_1} \geq d(j_1, j_2) + 6\bar{\alpha}_{j_2}$. It follows that

$$2\bar{\alpha}_j \geq d(j, j_1) + (6-2)\bar{\alpha}_{j_1} + d(j_1, j_2) + 6\bar{\alpha}_{j_2} \geq d(j, j_2) + 6\bar{\alpha}_{j_2}.$$

Thus j is stopped by j_2, as well. □

Intuitively, the stopping criterion will ensure that no α_j grows too large compared to the α-values of nearby clients. At the same time it is designed so that all decided clients will have a good approximation guarantee.

[5] Here, we use that all α-values will be at least one and two values in the same bucket differs thus by at most a factor $1+\epsilon$. We also remark that this is the same concept as in Invariant 1 of the quasi-polynomial algorithm.

Finally, we shall require that the following invariants hold throughout the execution of Algorithm 1.

Invariant 2 (Feasibility)**.** *For all $j \in \mathcal{D}$, $\alpha_j \geq 1$ and for all $i \in \mathcal{F}$, $\sum_{j \in \mathcal{D}} \beta_{ij} \leq z_i$.*

We remark that for dual feasibility $\alpha_j \geq 0$ is sufficient but the stronger assumption $\alpha_j \geq 1$ which is implied by Lemma 3.4.1 will be convenient.

Invariant 3 (No strict containment)**.** *For any two clients $j, j' \in \mathcal{D}$, $\bar{\alpha}_j \leq d(j, j') + \bar{\alpha}_{j'}$*

Note that the above invariant says that the ball centered at j of radius $\bar{\alpha}_j$ does *not* strictly contain the ball centered at j' of radius $\bar{\alpha}_{j'}$. For future reference, we refer to the ball centered at client j of radius $\bar{\alpha}_j$ as the α-ball of that client.

Invariant 4 ($(\alpha^{(0)}, z^{(0)})$ Completely Decided)**.** *Every client is decided in $(\alpha^{(0)}, z^{(0)})$.*

Invariant 4 will maintained as follows (as we show formally in Lemma 3.8.1): The initial solution satisfies the invariant. Then, given an initial solution $(\alpha^{(0)}, z^{(0)})$ in which all clients are decided, RaisePrice will output a close, roundable sequence $\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(q)}$, where $\mathcal{S}^{(q)} = (\alpha^{(q)}, z^{(q)}, \emptyset, \mathcal{D}_{\mathsf{S}}^{(q)})$ is a roundable solution in which all clients are decided. As the next call to RaisePrice will use $(\alpha^{(q)}, z^{(q)})$ as the initial solution, the invariant is maintained.

3.7.1 The RaisePrice procedure

RaisePrice is described in detail in Algorithm 2. Initially, we suppose that we are given a λ-roundable and a completely decided dual solution $(\alpha^{(0)}, z^{(0)})$ (i.e., satisfying Invariant 4) where $z_i \in \{\lambda, \lambda + \epsilon_z\}$ for all $i \in \mathcal{F}$. Additionally, let $\mathsf{IS}^{(0)}$ be the independent set $(\alpha^{(0)}, z^{(0)})$ of the conflict graph $H^{(0)}$ associated to the roundable solution $(\alpha^{(0)}, z^{(0)})$, produced at the end of the previous call to GraphUpdate as described in Algorithm 1. We shall assume that $|\mathsf{IS}^{(0)}| \geq k$, as otherwise, Algorithm 1 would have already terminated. For a specified facility i^+, RaisePrice sets $z_{i^+} \leftarrow z_{i^+} + \epsilon_z$. This may result in some clients using i^+ as a witness becoming undecided; specifically, those clients that are not stopped and have no witness except i^+ in $(\alpha^{(0)}, z^{(0)})$. We let $U^{(0)}$ to be the set of all these initially undecided clients. Throughout RaisePrice, we maintain a set U of currently undecided clients, and repair the solution over a series of multiple stages, by calling an auxiliary procedure, Sweep. Each repair stage s will be associated with a threshold θ_s, and will make multiple calls to the procedure Sweep, each producing a new solution α. The algorithm RaisePrice constructs a roundable solution $\mathcal{S} = (\alpha, z, \mathcal{F}_{\mathsf{S}}, \mathcal{D}_{\mathsf{S}})$ from each such α, and returns the sequence $\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(q)}$ of all such roundable solutions, in the order they were constructed. RaisePrice terminates once it constructs some solution in which all clients are decided. In Section 3.8, we shall show that this must happen after

Algorithm 2: RaisePrice($\alpha^{(0)}, z^{(0)}, \mathsf{IS}^{(0)}, i^+$)

Input: $(\alpha^{(0)}, z^{(0)})$: a λ-roundable solution satisfying Invariant 4 and each $z_i \in \{\lambda, \lambda + \epsilon_z\}$.
$\mathsf{IS}^{(0)}$: the independent set of conflict graph $H^{(0)}$ produced by GraphUpdate.
i^+: a facility whose price z_{i^+} is being increased from λ to $\lambda + \epsilon_z$

Output: Sequence $\mathcal{S}^{(1)} = (\alpha^{(1)}, z^{(1)}, \mathcal{F}_\mathsf{s}^{(1)}, \mathcal{D}_\mathsf{s}^{(1)}), \ldots, \mathcal{S}^{(q)} = (\alpha^{(q)}, z^{(q)}, \mathcal{F}_\mathsf{s}^{(q)}, \mathcal{D}_\mathsf{s}^{(q)})$ of close λ-roundable solutions, where all clients are decided in $\mathcal{S}^{(q)}$.

$(\alpha, z) \leftarrow (\alpha^{(0)}, z^{(0)})$
$z_{i^+} \leftarrow z_{i^+} + \epsilon_z$
Let $U^{(0)}$ be the set of clients now undecided.
Set $K = \Theta(\epsilon^{-1}\gamma^{-4})$ and select a shift parameter $0 \le \sigma < K/2$.
Set $\theta_1 = (\max_{j \in U^{(0)}} \alpha_j^{(0)} + 2\epsilon_z)(1+\epsilon)^\sigma$ and $\theta_s = (1+\epsilon)^K \theta_{s-1}$ for all $s > 1$.
$U \leftarrow U^{(0)}$
$\ell \leftarrow 1, s \leftarrow 1$
while $U \neq \emptyset$ **do**
  `/* Execute repair stage` s `*/`
  while *there is some* $j \in U$ *with* $\alpha_j < \theta_s$ **do**
    $\alpha \leftarrow$ Sweep(θ_s, α) (this procedure is described in Section 3.7.2)
    $U \leftarrow$ set of clients now undecided.
    Form $\mathcal{F}_\mathsf{s}$ and $\mathcal{D}_\mathsf{s}$ using α, z, $\alpha^{(0)}$, and $\mathsf{IS}^{(0)}$.
    $\mathcal{S}^{(\ell)} \leftarrow (\alpha, z, \mathcal{F}_\mathsf{s}, \mathcal{D}_\mathsf{s})$.
    $\ell \leftarrow \ell + 1$
  $s \leftarrow s + 1$

at most $O(\log n)$ stages, and that each stage requires only a polynomial number of calls to Sweep. In addition, we show that the produced sequence is close and roundable.

Before describing Sweep in detail, let us first provide some intuition for the selection of the thresholds θ_s and describe the construction of each roundable solution $\mathcal{S}^{(\ell)} = (\alpha^{(\ell)}, z^{(\ell)}, \mathcal{F}_\mathsf{s}^{(\ell)}, \mathcal{D}_\mathsf{s}^{(\ell)})$ in RaisePrice. Our procedure Sweep will adjust client values α_j similarly to the procedure QuasiSweep described in Section 3.4.1. However, in each stage s, we ensure that Sweep never increases any α_j above the threshold θ_s beyond its initial value $\alpha_j^{(0)}$, i.e., we ensure that $\alpha_j \le \alpha_j^{(0)}$ for any $\alpha_j \ge \theta_s$. We set

$$\theta_1 = (\max_{j \in U^{(0)}} \alpha_j^{(0)} + 2\epsilon_z)(1+\epsilon)^\sigma \quad \text{and} \quad \theta_s = (1+\epsilon)^K \theta_{s-1}\,,$$

where $K = \Theta(\epsilon^{-1}\gamma^{-4})$ is an integer parameter and σ is a integer "shift" parameter chosen uniformly at random[6] from $[0, K/2)$. Our selection of thresholds ensures that each stage updates only those α_j in a constant K number of buckets. Thus, the total change in any α-value will be at most $n^{O(K)}$, which will allow us to obtain a polynomial running time.

[6] We shall show that it is in fact easy to select an appropriate σ deterministically (see Remark 3.8.16).

This comes at the price of some clients remaining undecided after each stage, and some such clients j may have service cost much higher than $\rho \cdot \alpha_j$. We let $\mathcal{B}$ denote the set of all such "bad" clients. Using that the α-values are relatively well-behaved throughout RaisePrice, we show that only those clients j with $\alpha_j^{(0)}$ relatively near to the threshold θ_s can be added to $\mathcal{B}$ in stage s. Then, the random shift σ in choosing our definition of thresholds will allow us to show that only an $O(K^{-1})$ fraction of clients can be bad throughout RaisePrice. Moreover, we can bound the cost of each client $j \in \mathcal{B}$ by $36\alpha_j^{(0)}$. Intuitively, then, if at least a constant fraction of each $\alpha_j^{(0)}$ is contributing to the service cost $c(j, \mathsf{IS}^{(0)})$, then we can bound the effect of these bad clients by setting K to be a sufficiently large constant, then using Theorem 3.6.4 to conclude that:

$$\sum_{j \in \mathcal{B}} 36\alpha_j^{(0)} \leq \epsilon \cdot \sum_{j \in \mathcal{D}} c(j, \mathsf{IS}^{(0)}) \leq O(\epsilon) \cdot \mathrm{OPT}_k \,.$$

Unfortunately, it may happen that many clients $j \in \mathcal{B}$ have $\alpha_j^{(0)} - c(j, \mathsf{IS}^{(0)}) \approx \alpha_j^{(0)}$. That is, some clients may be using almost all of their $\alpha^{(0)}$-values to pay for the opening costs of facilities. In this case, we could have $\sum_{j \in \mathcal{B}} \alpha_j^{(0)}$ arbitrarily larger than $\sum_{j \in \mathcal{D}} (j, \mathsf{IS}^{(0)})$. In order to cope with this situation, we introduce a notion of *dense* clients and facilities in Section 3.8.5. These troublesome clients and facilities are handled by carefully constructing the remaining components $\mathcal{F}_\mathsf{s}$ and $\mathcal{D}_\mathsf{s}$ of the roundable solution in line 12. We defer the formal details to Section 3.8.5, but the intuition is if enough bad clients are paying mostly for the opening cost of a facility, then we can afford to open this facility *even if it is not tight*. This is precisely the role of special facilities in Definition 3.5.1.

3.7.2 The Sweep procedure

It remains to describe our last procedure, Sweep in more detail. Sweep operates in some stage s, with corresponding threshold value θ_s, takes as input the previous α produced by the algorithm, and produces a new α. Note that in every call to Sweep, we let $\alpha^{(0)}$ denote the roundable solution passed to RaisePrice, and U is the set of undecided clients immediately before Sweep was called. Just like QuasiSweep, the procedure Sweep, maintains a current set of active clients A and a current threshold θ, where initially, $A = \emptyset$, and $\theta = 0$. We slowly increase θ and whenever $\theta = \alpha_j$ for some client j, we add j to A. While $j \in A$, we increase α_j with θ. However, in contrast to QuasiSweep, Sweep removes a client j from A, whenever one of the following five events occurs:

Rule 1. j has some witness i.

Rule 2. j is stopped by some client j'.

Rule 3. $j \in U$ and α_j is ϵ_z larger than its value at the start of Sweep.

Rule 4. $\alpha_j \geq \theta_s$ and $\alpha_j \geq \alpha_j^{(0)}$.

Rule 5. There is a client j' that has already been removed from A such that $\bar{\alpha}_j \geq d(j, j') + \bar{\alpha}_{j'}$.

We remark that Rule 5 says that j is removed from A as soon as its α-ball contains the α-ball of another client j' that is not currently in A. This rule is designed so that the algorithm maintains Invariant 3. Also note that if a client j satisfies one of these conditions when it is added to A, then we remove j from A immediately after it is added. In this case, α_j is not increased.

As in QuasiSweep, increasing the values α_j for clients in A may cause $\sum_{j \in \mathcal{D}} \beta_{ij}$ to exceed z_i for some facility i. We again handle this by decreasing some other values $\alpha_{j'}$. However, here we are more careful in our choice of clients to decrease. Let us call a facility i *potentially tight* if one of the following conditions hold:

- There is some $j \in N(i)$ with $\alpha_j > \alpha_j^{(0)}$.
- For all $j \in N^{(0)}(i)$, $\alpha_j \geq \alpha_j^{(0)}$.

We now decrease $\alpha_{j'}$ if and only if $B(\alpha_{j'}) > B(\theta)$ and additionally: for some potentially tight facility i with $|N(i) \cap A| \geq 1$, we have $\alpha_{j'} = t_i$. We decrease each such $\alpha_{j'}$ at a rate of $|A|$ times the rate that θ is increasing. To see that this maintains feasibility we observe that at any time there are $|A \cap N(i)|$ clients whose contribution to facility i is increasing, and these contributions are increasing at the same rate as θ. Suppose that i is tight at some moment with some $j \in N(i) \cap A$. Then, since $z_i \geq z_i^{(0)}$, there must be either at least one client j' with $\alpha_{j'} > \alpha_{j'}^{(0)}$ or $\alpha_{j'} = \alpha_{j'}^{(0)}$ for all $j' \in N^{(0)}(i)$, and so i must be potentially tight. Consider some client j_0 with $\alpha_{j_0} = t_i$ and note that $B(\alpha_{j_0}) = B(t_i) > B(\alpha_j) = B(\theta)$, since otherwise we would remove j from A by Rule 1. The value of α_{j_0} is currently decreasing at a rate of $|A| \geq |N(i) \cap A|$ times the rate that θ is increasing. Thus, the total contribution to any tight facility i is never increased.

As in QuasiSweep, we stop increasing θ once every client j has been added and removed from A, and then output the resulting α. Note that Sweep never changes any $\alpha_j < \theta$. In particular, once some j has been removed from A it is not subsequently changed. Additionally, observe that once $B(\theta) \geq B(\alpha_j)$, Sweep will not decrease α_j.

3.8 Analysis of the Polynomial-Time Algorithm

Unfortunately, in contrast to the quasi-polynomial time procedure, here our analysis is quite involved. Let us first provide a high-level overview of our overall approach. Note that any solution that does not contain any undecided clients must necessarily be roundable with $\mathcal{F}_{\mathsf{S}} = \emptyset$, and $\mathcal{D}_{\mathsf{B}} = \emptyset$. Indeed, for any witnessed client j there is a tight

facility i with $(1+\epsilon)\bar{\alpha}_j \geq \sqrt{t_i}$ and $\bar{\alpha}_j \geq d(j,i)$ and so

$$(1+(1+\epsilon)\sqrt{\delta})\bar{\alpha}_j \geq d(j,i) + \sqrt{\delta t_i}\,.$$

Similarly, any stopped client j in such a solution must be stopped by some witnessed j' (using Lemma 3.7.1). Let i be the witness of j'. Then,

$$d(j,i)+\sqrt{\delta t_i} \leq d(j,j')+d(j',i)+\sqrt{\delta t_i} \leq 2\bar{\alpha}_j - 6\bar{\alpha}_{j'} + (1+(1+\epsilon)\sqrt{\delta})\bar{\alpha}_{j'} < (1+\sqrt{\delta})\bar{\alpha}_j,$$

since $\sqrt{\delta} \geq 1$. As $\tau_i \leq t_i$ for any facility $i \in \mathcal{F}$, the required inequalities from Definition 3.5.1 hold for any decided client j. In the following our main goal will then be to bound the cost of the remaining, undecided clients in any solution produced by RaisePrice.

Our first task is to characterize which clients may *currently* be undecided. To this end, we first prove some basic properties about the way Sweep alters α-values together with Invariants 2, 3, and 4 (Section 3.8.1). Then, we show that only clients above threshold θ_s in each stage s can become undecided (Section 3.8.2). In Section 3.8.3, we bound the cost of all decided and undecided clients, showing that we can indeed obtain a $(\rho + O(\epsilon))$-approximation for all decided clients and a slightly worse guarantee for undecided ones. Next, we would like to argue that most clients have good connection cost. Specifically, we would like to choose a set of thresholds that ensure that only a constant fraction of clients become undecided throughout the *entirety* of RaisePrice. In order to accomplish this, we show that our α-values remain relatively stable throughout RaisePrice (Section 3.8.4). This allows us to prove that RaisePrice outputs close solutions and to characterize those clients that may become undecided in RaisePrice by their values $\alpha_j^{(0)}$ at the *beginning* of RaisePrice. This, together with our selection of thresholds, ensures that only an arbitrarily small, constant fraction of clients do not have the desired guarantee. However, we must also show that these clients do not contribute more than a constant to OPT_k. As discussed above, this will follow immediately from our analysis for those clients whose service cost is at least a constant fraction of $\alpha_j^{(0)}$. For other (i.e. *dense*) clients, we must use a different argument, involving the sets of special facilities and clients $\mathcal{F}_{\mathsf{S}}$ and $\mathcal{D}_{\mathsf{S}}(i)$ (Section 3.8.5). Finally, we put all of these pieces together and show that RaisePrice produces a close sequence of polynomially many roundable solutions and runs in polynomial time (Section 3.8.6).

3.8.1 Basic properties of Sweep and Invariants 2, 3, and 4

We start by showing that Invariants 2, 3, and 4 hold.

Lemma 3.8.1. *Invariants 2, 3, and 4 hold throughout Algorithm 1.*

Proof. We begin by proving Invariant 2, i.e., that the algorithm maintains a feasible dual solution α with the additional property that $\alpha_j \geq 1$ for all $j \in \mathcal{D}$. Recall our construction

of the initial solution $\alpha^{(0)}$ for Algorithm 1: we set $\alpha_j = 0$ for all $j \in \mathcal{D}$ and then increase all α_j at a uniform rate. We stop increasing a value α_j whenever j gains a tight edge to some facility $i \in \mathcal{F}$ or $2\bar{\alpha}_j \geq d(j,j') + 6\bar{\alpha}_{j'}$ for some $j' \in \mathcal{D}$. Note that no α_j is increased after $\alpha_j = d(j,i)^2$ for some facility i. Thus, we have $\beta_{ij}^{(0)} = 0$ for all $j \in \mathcal{D}$ and $i \in \mathcal{F}$, and so $\alpha^{(0)}$ is feasible. Now, we show that $\min_{j\in\mathcal{D}} \alpha_j^{(0)} \geq 1$. Consider the client j_0 that first stops increasing in our greedy initialization process. At the time α_{j_0} stops increasing, we have $\alpha_j = \alpha_{j_0}$ for all $j \in \mathcal{D}$ and so $2\bar{\alpha}_j \geq d(j,j') + 6\bar{\alpha}_{j'}$ cannot hold for any pair j, j' of clients. Thus, j_0 must have stopped increasing because $\alpha_{j_0} = d(j_0,i)^2$ for some facility i. By our preprocessing (Lemma 3.4.1) we have $d(j_0,i)^2 \geq 1$, and so $\alpha_{j_0}^{(0)} \geq 1$. Moreover, $\alpha_{j_0}^{(0)} = \min_{j\in\mathcal{D}} \alpha_j^{(0)}$, and so indeed $\alpha_j^{(0)} \geq 1$ for all $j \in \mathcal{D}$. Now, we show that Algorithm 1 preserves Invariant 2. Note that α is altered only by subroutine Sweep, and by construction, Sweep ensures that always $\sum_j \beta_{ij} \leq z_i$. Moreover, Sweep decreases any α_j only while it is increasing some other $\alpha_{j'} < \alpha_j$ such that j' has a tight edge to some facility i. By our preprocessing (Lemma 3.4.1) $\alpha_{j'} \geq d(j',i)^2 \geq 1$ for any such j'. Thus, no α_j is ever decreased below 1.

Next, we prove Invariant 3, i.e., that no client's α-ball is strictly contained in the α-ball of another client. First, let us show that the initially constructed solution $(\alpha^{(0)}, z^{(0)})$ satisfies Invariant 3. Note that $\alpha_j^{(0)}$ is equal to the value of α_j at the time that our initialization procedure stopped increasing α_j. Consider any pair of clients j and j'. If $\alpha_j^{(0)} \leq \alpha_{j'}^{(0)}$ then clearly $\bar{\alpha}_j^{(0)} \leq d(j,j') + \bar{\alpha}_{j'}^{(0)}$. Thus, suppose that $\alpha_j^{(0)} > \alpha_{j'}^{(0)}$, so $\alpha_{j'}$ stopped increasing before α_j in our initialization procedure. If $\alpha_{j'}$ stopped increasing because j' gained a tight edge to a facility i, then once $\bar{\alpha}_j = d(j,j') + \bar{\alpha}_{j'}$, j will have a tight edge to i and stop increasing. If $\alpha_{j'}$ stopped increasing because $2\bar{\alpha}_{j'} = d(j',j'') + 6\bar{\alpha}_{j''}$ for some client j'', then when $\bar{\alpha}_j = d(j,j') + \bar{\alpha}_{j'}$ we will have

$$2\bar{\alpha}_j = 2d(j,j') + 2\bar{\alpha}_{j'} = 2d(j,j') + d(j',j'') + 6\bar{\alpha}_{j''} \geq d(j,j'') + 6\bar{\alpha}_{j''}$$

and so α_j must stop increasing. In any case, we must have $\bar{\alpha}_j^{(0)} \leq d(j,j') + \bar{\alpha}_{j'}^{(0)}$. Having shown that the invariant is true for the first $\alpha^{(0)}$ constructed in Algorithm 1, let us now prove that it is maintained. First, we show that the inequality $\bar{\alpha}_j \leq d(j,j') + \bar{\alpha}_{j'}$ will not be violated by increasing $\bar{\alpha}_j$. Suppose that $j \in A$ and so α_j is increasing. As long as $j' \in A$, as well, we have $\alpha_j = \alpha_{j'} = \theta$, and so $\bar{\alpha}_j \leq d(j,j') + \bar{\alpha}_{j'}$. On the other hand, if $j' \notin A$, then as soon as $\bar{\alpha}_j = d(j,j') + \bar{\alpha}_{j'}$, j will be removed from A by Rule 5 and $\bar{\alpha}_j$ will no longer increase. Now we show that also $\bar{\alpha}_j \leq d(j,j') + \bar{\alpha}_{j'}$ will not be violated by decreasing $\alpha_{j'}$. Suppose that $\alpha_{j'}$ is decreasing. Then, there must be some potentially tight facility i with $j' \in N(i)$ and $t_i = \alpha_{j'}$. Let i be *any* such facility. If at some point we have $\bar{\alpha}_j = d(j,j') + \bar{\alpha}_{j'}$, then we must also have $j \in N(i)$ at this moment and $\alpha_j \geq \alpha_{j'} = t_i$. Thus, α_j is also decreasing and in fact $\alpha_j = \alpha_{j'}$ (since also $\alpha_j \leq t_i$). Then, $\bar{\alpha}_j$ and $\bar{\alpha}_{j'}$ are decreasing at same rate and so $\bar{\alpha}_j = d(j,j') + \bar{\alpha}_{j'}$ as long as $\bar{\alpha}_{j'}$ continues to decrease.

Finally, we prove Invariant 4, i.e., that the input solution $(\alpha^{(0)}, z^{(0)})$ to RaisePrice is always completely decided. Every client j is either stopped by some client j' or has a tight edge to some facility i in our initially constructed solution $(\alpha^{(0)}, z^{(0)})$. Moreover, the initialization process ensures that $N(i) = \emptyset$ for all i (since $\beta_{ij} = 0$ for all $i \in \mathcal{F}$ and $j \in \mathcal{D}$). Thus, in the latter case $t_i = 0$ and so i is in fact a witness for j, and so every client j is indeed either stopped or witnessed in this initial solution $(\alpha^{(0)}, z^{(0)})$. To show that Invariant 4 holds throughout the rest of the Algorithm 1, we note that $(\alpha^{(0)}, z^{(0)})$ is always updated (in line 10 of Algorithm 1 where $\mathcal{S}^{(0)} \leftarrow \mathcal{S}^{(q)}$) with the α-values corresponding to the last solution produced in a call to RaisePrice. Due to the condition in the main loop of RaisePrice, every client is decided in this solution. □

The next lemma makes some basic observations about the way in which Sweep alters the α-values.

Lemma 3.8.2. *The procedure* Sweep *satisfies the following properties:*

Property 1. *Any client j that becomes decided after being added to A remains decided until the end of* Sweep.

Property 2. *If the α-ball of a client j contains the α-ball of a decided client, then j is decided.*

Property 3. *Consider the solution α at the beginning of* Sweep, *and let $\mu = \min_{j' \in U} \alpha_{j'}$. Then, no $\alpha_j < \mu$ is increased by* Sweep, *and no α_j with $B(\alpha_j) \leq B(\mu)$ is decreased by* Sweep.

Proof. For Property 1, suppose first that j had a witness i at some point after being added to A. Consider any $j' \in N(i)$ at this moment. At this moment, we must have $B(\alpha_{j'}) \leq B(\alpha_j) \leq B(\theta)$ and so $\alpha_{j'}$ cannot be decreased for the remainder of Sweep. In particular, j retains a tight edge to i until the end of Sweep and i remains tight until the end of Sweep. Additionally, any client j' with $\alpha_{j'} > t_i$ will be removed from A as soon as it gains a tight edge to i (by Rule 1 since i would then be a witness for j'). Thus, t_i cannot increase and so i remains a witness for j until the end of Sweep. Next, suppose that j was stopped by some j' after being added to A. Then, at this moment, $\alpha_{j'} < \alpha_j \leq \theta$. Hence, for the remainder of Sweep, neither $\alpha_{j'}$ or α_j are changed and so j remains stopped by j'.

For Property 2 suppose that the α-ball of client j contains the α-ball of a decided client j'. Then if j' has a witness i, then i is also a witness for j, since $\bar{\alpha}_j \geq d(j, j') + \bar{\alpha}_{j'} \geq d(j, j') + d(j', i) \geq d(j, i)$ and $B(\alpha_j) \geq B(\alpha_{j'}) \geq B(t_i)$. Similarly if j' is stopped by some client j'' then

$$2\bar{\alpha}_j \geq 2(d(j, j') + \bar{\alpha}_{j'}) \geq 2d(j, j') + 6\bar{\alpha}_{j''} + d(j', j'') \geq 6\bar{\alpha}_{j''} + d(j, j'')\,,$$

and so j is also stopped by j''.

Finally, for Property 3, consider the first client j whose value α_j is increased by Sweep. Note that j must not be decided before calling Sweep: otherwise, since no other α-value has yet been changed, this would hold at the moment j was added to A, as well, and so j would immediately be removed by Rule 1 or 2. Thus, the first α_j that is increased by Sweep must correspond to some $j \in U$, and at the moment this occurs, $\theta = \alpha_j \geq \mu$. Furthermore, by the definition of Sweep, no α_j can then be decreased unless $B(\alpha_j) \geq B(\mu) + 1$. □

3.8.2 Characterizing currently undecided clients

The next observations follow rather directly from the properties given in Lemma 3.8.2 and the invariants. These facts will help us bound the number of clients that can become bad throughout the algorithm, and also the total number of calls to Sweep that must be executed in each call to RaisePrice. Throughout this section, we consider a single call to RaisePrice and let $(\alpha^{(0)}, z^{(0)}, \mathsf{IS}^{(0)}, i^+)$ be its input.

Lemma 3.8.3. *In stage 1,* Sweep *is executed only a single time. After this call, for every $j \in U^{(0)}$, we have $\alpha_j \leq \alpha_j^{(0)} + \epsilon_z < \theta_1$ and j is decided.*

Proof. Consider any client $j_0 \in U^{(0)}$. Then i^+ was j_0's witness in $(\alpha^{(0)}, z^{(0)})$, and j_0 must not have been stopped or have had any other witness $i \neq i^+$. Observe that our choice of θ_1 ensures that $\alpha_{j_0}^{(0)} + \epsilon_z < \theta_1$, so any $j_0 \in U^{(0)}$ will be removed from A by Rule 3 once $\alpha_j = \alpha_j^{(0)} + \epsilon_z$. Thus we must $\alpha_j \leq \alpha_{j_0}^{(0)} + \epsilon_z < \theta_1$ at the end of Sweep for every $j_0 \in U^{(0)}$.

This also implies that no such j_0 is removed from A by Rule 4. We now show that when j_0 is removed from A by any other rule, it must be decided. By Property 1, j_0 is then decided at the end of Sweep, as well. First, we observe that if j_0 is removed from A by Rules 1 or 2, then it is decided by definition. Next, suppose that j_0 was removed by Rule 3, and let $\mu = \min_{j \in U^{(0)}} \alpha_j^{(0)}$. Since i^+ was a witness for every $j \in U^{(0)}$, we must have $B(\alpha_j^{(0)}) \leq B(\mu)$ for all $j \in N^{(0)}(i^+)$. Thus, by Property 3 of Sweep, $\alpha_j \geq \alpha_j^{(0)}$ for every $j \in N^{(0)}(i^+)$. Then, since $\alpha_{j_0} = \alpha_{j_0}^{(0)} + \epsilon_z$, at the time j_0 was removed from A, i^+ must have been tight and also a witness for j_0. By Property 1, j_0 then remains decided until the end of Sweep. Finally, we consider the case in which j_0 was removed by Rule 5. We show the following:

Claim. *Suppose that some client j is removed from A by Rule 5 and that j is undecided at this time. Then, $\alpha_j \geq \theta_1$.*

Proof. Consider the first time that any client j that is undecided is removed from A

by Rule 5. By Property 2, the α-ball of this client j must contain the α-ball of some undecided client j' that was previously removed from A. By Property 1 and our choice of time, j' must have been removed from A by Rule 3 or 4. However, if j' was removed by Rule 3, we must have $j' \in U^{(0)}$ and so, as we have previously shown, j' must be decided. Thus, j' was removed by Rule 4, and so presently $\alpha_j = \theta \geq \alpha_{j'} \geq \theta_1$. To complete the proof, we observe that any client that is removed from A after j must have an α-value at least $\alpha_j \geq \theta_1$. □

It follows by the above Claim that no $j_0 \in U^{(0)}$ can be undecided when it is removed by Rule 5, since, as we have shown, $\alpha_{j_0} < \theta_1$ for all such j_0. By the above cases, every client $j_0 \in U^{(0)}$ is decided with $\alpha_j \leq \alpha_{j_0}^{(0)} + \epsilon_z < \theta_1$ at the end of Sweep.

It remains to show that RaisePrice continues to stage 2 after one call to Sweep. Consider some client j that is undecided at the end of Sweep. By Property 1 j must not have been removed from A by Rule 1 or Rule 2. Moreover, we must have $j \notin U^{(0)}$ and so j was not removed from A by Rule 3. Thus, j was removed from A by Rule 4 or 5. In either case, $\alpha_j \geq \theta_1$ at this moment (and so also at the end of Sweep, since no α_j is changed after j is removed from A). Thus, after the first call to Sweep in stage 1, every undecided client j has $\alpha_j \geq \theta_1$ and so RaisePrice immediately continues to stage 2. □

Lemma 3.8.4. *Consider any solution (α, z) produced by* RaisePrice. *If j is undecided in (α, z), then $\alpha_j \geq \alpha_j^{(0)}$.*

Proof. Suppose toward contradiction that the statement is false. Consider the first call to Sweep that produces a solution violating it and for this call let j be the first client (in the order of removal from A) such that $\alpha_j < \alpha_j^{(0)}$ when j is removed from A but j is undecided[7]. Then since, j is undecided it was removed by Rule 3, 4, or 5. If j was removed by Rule 4, then at this moment $\alpha_j \geq \alpha_j^{(0)}$. Suppose then that j was removed by Rule 3. Then, $j \in U$. By Lemma 3.8.3, no client $j \in U$ before the first call to Sweep is undecided after this call, so j must have been undecided at the end of some preceding call to Sweep. By assumption, we must have had $\alpha_j \geq \alpha_j^{(0)}$ at the moment j was removed from A in this preceding call (and so also immediately before the present call). But, α_j has increased by ϵ_z, so still $\alpha_j \geq \alpha_j^{(0)}$. Finally, suppose j was removed by Rule 5. Then, the α-ball of j must contain the α-ball of a client j' that has already been removed A. If j' is decided, then by Property 2 j is decided as well. Suppose that j' is undecided. Then, since we picked the first client that violated the condition of the lemma, and j' was already removed from A, we have that $\alpha_{j'} \geq \alpha_{j'}^{(0)}$. But then, if $\alpha_j < \alpha_j^{(0)}$, we have $\bar{\alpha}_j^{(0)} > \bar{\alpha}_j \geq d(j, j') + \bar{\alpha}_{j'} \geq d(j, j') + \bar{\alpha}_{j'}^{(0)}$, which contradicts Invariant 3. In all cases we showed that we must have $\alpha_j \geq \alpha_j^{(0)}$ at the moment that j was removed from A,

[7]By Property 1, any client j violating the statement must be undecided when removed from A and have $\alpha_j < \alpha_j^{(0)}$ at the time of its removal from A since Sweep does not change j's α-value thereafter.

and so also at the end of Sweep, contradicting our assumption that $\alpha_j < \alpha_j^{(0)}$ for some undecided client j. □

Lemma 3.8.5. *In every stage $s > 1$, no α_j is changed by* Sweep *until $\theta \geq \theta_{s-1}$. In particular, every client j with $\alpha_j < \theta_{s-1}$ is decided for every solution produced by* RaisePrice *in stage $s > 1$.*

Proof. By Property 3 of Sweep, no α_j is changed until $\theta = \min_{j \in U} \alpha_j$. Thus to prove the first part of the claim, it suffices to show that in every stage $s > 1$, if $U \neq \emptyset$ then $\min_{j \in U} \alpha_j \geq \theta_{s-1}$. Note that the second part of the claim then follows as well for every solution except the one produced by the final call to Sweep in RaisePrice, and this last solution has no undecided clients by Invariant 4.

Let us now prove that $\min_{j \in U} \alpha_j \geq \theta_{s-1}$ in every stage $s > 1$. We proceed by induction on the number of calls to Sweep made in stage s. Before the first call to Sweep in stage s, we must have $\alpha_j \geq \theta_{s-1}$ for every $j \in U$, since otherwise stage $s-1$ would have continued. So, consider some later call to Sweep in stage s, and consider any $j \in U$ before this call. Then, we must have had j undecided after the preceding call to Sweep in stage s. Moreover, by Property 1, j must have been undecided when it was removed from A in this preceding call. Consider the *first* client j that was undecided upon removal from A in this preceding call. Then, j cannot have been removed by Rules 1 or 2. Moreover, since every client that has been removed from A before j is decided, Property 2 implies that j must not have been removed by Rule 5. If j was removed by Rule 3, then we must have had $j \in U$ already in this preceding call to Sweep, and so by the induction hypothesis, $\alpha_j \geq \theta_{s-1}$. Then, since j was removed from A by Rule 3, we had $\alpha_j \geq \theta_{s-1} + \epsilon_z$. Finally, if j was removed by Rule 4, then we must have $\alpha_j \geq \theta_s > \theta_{s-1}$ by definition. Thus, throughout every stage $s > 1$, if $U \neq \emptyset$, then $\min_{j \in U} \alpha_j \geq \theta_{s-1}$, as desired. □

Corollary 3.8.6. *Suppose that in $(\alpha^{(0)}, z^{(0)})$, j is not stopped and has only i^+ as a witness, i.e., $j \in U^{(0)}$. Then, we have that j is decided with $\alpha_j \leq \alpha_j^{(0)} + \epsilon_z$ in every solution (α, z) produced by* RaisePrice$(\alpha^{(0)}, z^{(0)}, \mathsf{IS}^{(0)}, i^+)$.

Proof. We have $j \in U^{(0)}$ and so by Lemma 3.8.3, j is decided with $\alpha_j \leq \alpha_j^{(0)} + \epsilon_z < \theta_1$ in the first solution produced by RaisePrice. Moreover, by Lemma 3.8.5, α_j remains unchanged and j remains decided in all later stages. □

3.8.3 Bounding the cost of clients

In this section we derive inequalities that are used to bound the service cost of each (α, z) produced during the algorithm. Consider some solution α produced by the algorithm,

and define

$$\mathcal{B} = \{j \in \mathcal{D} : \ j \text{ is undecided and } 2\bar{\alpha}_j < d(j,j') + 6\bar{\alpha}_{j'}^{(0)} \text{ for all clients } j'\}\,. \quad (3.8.1)$$

The set $\mathcal{B}$ is defined to contain those clients that are (potentially) bad, i.e., have worse connection cost than our target guarantee. Specifically, we now show that all clients $j \in \mathcal{D} \setminus \mathcal{B}$, satisfy the first inequality of Property 2 in Definition 3.5.1 (with τ_i replaced by t_i), while all clients (in particular those in $\mathcal{B}$) satisfy a slightly weaker inequality.

Lemma 3.8.7. *Consider any (a, z) produced by* RaisePrice. *For every client j the following holds:*

- *If $j \in \mathcal{D}\setminus\mathcal{B}$, then there exists a tight facility i such that $(1+\sqrt{\delta}+\epsilon)\bar{\alpha}_j \geq d(j,i)+\sqrt{\delta t_i}$.*
- *There exists a tight facility i such that $6\bar{\alpha}_j^{(0)} \geq d(j,i) + \sqrt{\delta t_i}$.*

Proof. The proof is by induction on the well-ordered set (with respect to the natural order $\leq$)

$$R = \{0\} \cup \{\alpha_j\}_{j\in\mathcal{D}\setminus\mathcal{B}} \cup \{(1+\epsilon)\alpha_j^{(0)}\}_{j\in\mathcal{D}}\,.$$

Specifically, we prove the following induction hypothesis: for $r \in R$,

(a) each client $j \in \mathcal{D} \setminus \mathcal{B}$ with $\alpha_j \leq r$ has a tight facility i such that $(1 + \sqrt{\delta} + \epsilon)\bar{\alpha}_j \geq d(j,i) + \sqrt{\delta t_i}$;

(b) each client $j \in \mathcal{D}$ with $(1 + \epsilon)\alpha_j^{(0)} \leq r$ has a tight facility i such that $6\bar{\alpha}_j^{(0)} \geq d(j,i) + \sqrt{\delta t_i}$.

The statement then follows from the above with $r = \arg\max_{r\in R} r$.

For the base case (when $r = 0$), the claim is vacuous since there is no client j such that $\alpha_j \leq 0$ or $(1 + \epsilon)\alpha_j^{(0)} \leq 0$ (because every α-value is at least 1 by Invariant 2). For the induction step, we assume that each client $j \in \mathcal{D} \setminus \mathcal{B}$ with $\alpha_j < r$ satisfies (a) and each client $j \in \mathcal{D}$ with $(1+\epsilon)\alpha_j^{(0)} < r$ satisfies (b). We need to prove that any client $j_0 \in \mathcal{D} \setminus \mathcal{B}$ with $\alpha_{j_0} = r$ (respectively, $j_0 \in \mathcal{D}$ with $(1 + \epsilon)\alpha_{j_0}^{(0)} = r$) satisfies (a) (respectively, (b)). We divide the proof into two cases.

Case 1: $j_0 \in \mathcal{D} \setminus \mathcal{B}$ with $\alpha_{j_0} = r$. We prove that in this case j_0 satisfies (a). Since $j_0 \notin \mathcal{B}$, either j_0 has a witness, j_0 is currently stopped, or there is another client j such that $2\bar{\alpha}_{j_0} \geq d(j_0, j) + 6\bar{\alpha}_j^{(0)}$.

Suppose first that j_0 has a witness i. Then, i is a tight facility and, since j_0 has a tight edge to i, $d(j_0, i) \leq \bar{\alpha}_{j_0}$. Moreover, $B(\alpha_{j_0}) \geq B(t_i)$ which implies that $(1 + \frac{\epsilon}{2})\bar{\alpha}_{j_0} \geq \sqrt{(1+\epsilon)}\bar{\alpha}_{j_0} \geq \sqrt{t_i}$. Therefore, using that $\sqrt{\delta} \leq 2$,

$$d(j_0, i) + \sqrt{\delta t_i} \leq (1 + \sqrt{\delta} + \epsilon)\bar{\alpha}_{j_0} .$$

Now suppose that j_0 is stopped by another client j. Then $\alpha_j \leq \alpha_{j_0}/3^2 = r/9$. On the one hand, if $j \in \mathcal{D} \setminus \mathcal{B}$, we have $d(j_0, i) + \sqrt{\delta t_i} \leq (1 + \sqrt{\delta} + \epsilon)\bar{\alpha}_j \leq 6\bar{\alpha}_j$ for some tight facility i by the induction hypothesis (a). On the other hand, if $j \in \mathcal{B}$ then j is undecided so by Lemma 3.8.4, $\alpha_j^{(0)} \leq \alpha_j$. This in turn implies that $\alpha_j^{(0)} \leq \alpha_j \leq r/9 < r/(1+\epsilon)$. We can thus apply the induction hypothesis (b) to j, to conclude that there is a tight facility i such that $d(j, i) + \sqrt{\delta t_i} \leq 6\bar{\alpha}_j^{(0)} \leq 6\bar{\alpha}_j$. From above we have that, whether j is in $\mathcal{B}$ or not, there is a tight facility i such that

$$\begin{aligned} d(j_0, i) + \sqrt{\delta t_i} &\leq d(j_0, j) + d(j, i) + \sqrt{\delta t_i} \\ &\leq d(j_0, j) + 6\bar{\alpha}_j \\ &\leq 2\bar{\alpha}_{j_0} \leq (1 + \sqrt{\delta} + \epsilon)\bar{\alpha}_{j_0} , \end{aligned}$$

where the penultimate inequality uses the fact that j_0 is stopped by j and thus $2\bar{\alpha}_{j_0} \geq d(j, j_0) + 6\bar{\alpha}_j$.

Finally, suppose that j_0 is not stopped or witnessed. Then, j_0 is currently undecided and, as $j_0 \notin \mathcal{B}$, there is a client j such that $2\bar{\alpha}_{j_0} \geq d(j_0, j) + 6\bar{\alpha}_j^{(0)}$. This implies that $\alpha_j^{(0)} \leq \alpha_{j_0}/9 = r/9 < r/(1+\epsilon)$. We can thus apply the induction hypothesis (b) to j to conclude, that there is a tight facility i such that $d(j, i) + \sqrt{\delta t_i} \leq 6\bar{\alpha}_j^{(0)}$. Now, we have:

$$\begin{aligned} d(j_0, i) + \sqrt{\delta t_i} &\leq d(j_0, j) + d(j, i) + \sqrt{\delta t_i} \\ &\leq d(j_0, j) + 6\bar{\alpha}_j^{(0)} \\ &\leq 2\bar{\alpha}_{j_0} \leq (1 + \sqrt{\delta} + \epsilon)\bar{\alpha}_{j_0} . \end{aligned}$$

Case 2: $j_0 \in \mathcal{D}$ with $(1+\epsilon)\alpha_{j_0}^{(0)} = r$. We prove that in this case j_0 satisfies (b). Suppose first that $\alpha_{j_0} < \alpha_{j_0}^{(0)}$. Then j_0 is decided by Lemma 3.8.4. Therefore $j_0 \in \mathcal{D} \setminus \mathcal{B}$ with $\alpha_{j_0} < r$ and so by the induction hypothesis (a) there is a tight facility i satisfying $d(j_0, i) + \sqrt{\delta t_i} \leq (1 + \sqrt{\delta} + \epsilon)\bar{\alpha}_{j_0} < 6\bar{\alpha}_{j_0}^{(0)}$, as required. Similarly, if $j_0 \in U^{(0)}$ then by Corollary 3.8.6, j_0 is decided and so

$$\alpha_{j_0} \leq \alpha_{j_0}^{(0)} + \epsilon_z < (1+\epsilon)\alpha_{j_0}^{(0)} = r ,$$

where the second inequality follows from $\epsilon_z < \epsilon$ and $\alpha_{j_0}^{(0)} \geq 1$ by Invariant 2. We can thus again apply the induction hypothesis (a) to conclude that there is a tight facility

i satisfying $d(j_0, i) + \sqrt{\delta t_i} \leq (1 + \sqrt{\delta} + \epsilon)\bar{\alpha}_{j_0} \leq 6\bar{\alpha}_{j_0}^{(0)}$. Thus, from now on, we assume that $\alpha_{j_0} \geq \alpha_{j_0}^{(0)}$ and that $j_0 \notin U^{(0)}$. We divide the remaining part of the analysis into two sub-cases depending on whether j_0 was stopped in $\alpha^{(0)}$.

First, suppose that j_0 was stopped in $\alpha^{(0)}$ by another client j. Then $\alpha_j^{(0)} \leq \alpha_{j_0}^{(0)}/9 < r/(1+\epsilon)$ and so by the induction hypothesis (b), there is a tight facility i satisfying $d(j, i) + \sqrt{\delta t_i} \leq 6\bar{\alpha}_j^{(0)}$. Hence,

$$\begin{aligned} d(j_0, i) + \sqrt{\delta t_i} &\leq d(j_0, j) + d(j, i) + \sqrt{\delta t_i} \\ &\leq d(j_0, j) + 6\bar{\alpha}_j^{(0)} \\ &\leq 2\bar{\alpha}_{j_0}^{(0)} < 6\bar{\alpha}_{j_0}^{(0)}. \end{aligned}$$

Finally, suppose that j_0 was not stopped in $\alpha^{(0)}$. Then since every client is decided in $\alpha^{(0)}$ (Invariant 4) j_0 had a witness i in $\alpha^{(0)}$. Moreover, as $j_0 \notin U^{(0)}$, we may assume that $i \neq i^+$ and so $z_i = z_i^{(0)}$. By the definition of a witness, $\alpha_{j_1}^{(0)} \leq (1+\epsilon)\alpha_{j_0}^{(0)}$ for all $j_1 \in N^{(0)}(i)$. If $\alpha_{j_1} \geq \alpha_{j_1}^{(0)}$ for all $j_1 \in N^{(0)}(i)$, then, since $z_i = z_i^{(0)}$, our feasibility invariant (Invariant 2) implies that in fact $\alpha_{j_1} = \alpha_{j_1}^{(0)}$ for all $j_1 \in N^{(0)}(i)$ and so $N(i) = N^{(0)}(i)$. Therefore, in this case i is still a witness for j_0 and $d(j_0, i) + \sqrt{\delta t_i} \leq (1 + \sqrt{\delta} + \epsilon)\bar{\alpha}_{j_0}^{(0)} \leq 6\bar{\alpha}_{j_0}^{(0)}$. It remains to consider the case when $\alpha_{j_1} < \alpha_{j_1}^{(0)}$ for some $j_1 \in N^{(0)}(i)$ (note that $j_1 \neq j_0$, since $\alpha_{j_0} \geq \alpha_{j_0}^{(0)}$ by assumption). Since $\alpha_{j_1} < \alpha_{j_1}^{(0)}$, j_1 must be decided (by Lemma 3.8.4) and so $j_1 \in \mathcal{D} \setminus \mathcal{B}$. Moreover, $\alpha_{j_1} < \alpha_{j_1}^{(0)} \leq (1+\epsilon)\alpha_{j_0}^{(0)} = r$, and so we can apply the induction hypothesis (a) to conclude that there is a tight facility i_1 satisfying $d(j_1, i_1) + \sqrt{\delta t_{i_1}} \leq (1 + \sqrt{\delta} + \epsilon)\bar{\alpha}_{j_1} < (1 + \sqrt{\delta} + \epsilon)\bar{\alpha}_{j_1}^{(0)}$. Then,

$$\begin{aligned} d(j_0, i_1) + \sqrt{\delta t_{i_1}} &\leq d(j_0, i) + d(i, j_1) + d(j_1, i_1) + \sqrt{\delta t_{i_1}} \\ &< \bar{\alpha}_{j_0}^{(0)} + \bar{\alpha}_{j_1}^{(0)} + (1 + \sqrt{\delta} + \epsilon)\bar{\alpha}_{j_1}^{(0)} \\ &\leq \bar{\alpha}_{j_0}^{(0)} + (1+\epsilon)\bar{\alpha}_{j_0}^{(0)} + (1+\epsilon)(1 + \sqrt{\delta} + \epsilon)\bar{\alpha}_{j_0}^{(0)} \\ &\leq 6\bar{\alpha}_{j_0}^{(0)}, \end{aligned}$$

as required. □

Lemma 3.8.7 shows that the clients in $\mathcal{D} \setminus \mathcal{B}$ satisfy the first inequality of Property 2 in Definition 3.5.1 while the potentially bad clients $j \in \mathcal{B}$ satisfy a slightly weaker inequality. It remains to prove that the potentially bad clients will have a small contribution towards the total cost of our solution.

3.8.4 Showing that α-values are stable

The key to our remaining analysis is showing that the α-values are relatively well-behaved throughout the algorithm. The following lemma implies that Sweep decreases an $\alpha_{j'}$ only because it is increasing an α_j which is at most a constant factor smaller. This will imply the required stability properties.

Lemma 3.8.8. *At any time during Algorithm 1: if a client j has a tight edge to some facility, then $\alpha_{j'} \leq 19^2 \alpha_j$ for every other client j' with a tight edge to this facility.*

Proof. We prove the following stronger statement: at any time during Algorithm 1, we have

$$2\bar{\alpha}_{j'} \leq d(j', j) + 18\bar{\alpha}_j \tag{3.8.2}$$

for any pair j, j' of clients. To see that this implies the lemma consider two clients j and j' that both have tight edges to i^*. Then

$$2\bar{\alpha}_{j'} \leq d(j', j) + 18\bar{\alpha}_j \leq d(j', i^*) + d(i^*, j) + 18\bar{\alpha}_j \leq \bar{\alpha}_{j'} + \bar{\alpha}_j + 18\bar{\alpha}_j,$$

which implies that $\alpha_{j'} \leq 19^2 \alpha_j$.

Inequality (3.8.2) is clearly satisfied by the initial solution $\alpha^{(0)}$ constructed at the beginning of Algorithm 1, since we stop increasing any α_j as soon as $2\bar{\alpha}_j \geq d(j', j) + 6\bar{\alpha}_{j'}$ for any client j', and neither α_j nor $\alpha_{j'}$ are later changed. We now show that (3.8.2) continues to hold throughout the execution of Algorithm 1. The only procedure that updates the dual solution is Sweep, so let us analyze its behavior.

First note that the inequality cannot become violated by increasing j', because as soon as $2\bar{\alpha}_{j'} \geq d(j', j) + 6\bar{\alpha}_j$, j' will be removed from A by Rule 2 of Sweep. It remains to prove that the inequality does not become violated because j is decreasing. To this end, consider a time when j is decreasing. Then, by the definition of Sweep, there must be some potentially tight facility i, such that $j \in N(i)$ with $\alpha_j = t_i$. Since j has the largest α-value in $N(i)$ and i is potentially tight, there is some client $j_1 \in N(i)$ (note that possibly $j_1 = j$) such that $\alpha_{j_1}^{(0)} \leq \alpha_{j_1} \leq \alpha_j$. We show the following:

Claim. *There exists some facility $i^\star$ such that $i^\star$ was tight in $(\alpha^{(0)}, z^{(0)})$ and also:*

$$d(j_1, i^\star) \leq 2\bar{\alpha}_{j_1}^{(0)} \leq 2\bar{\alpha}_j \qquad \text{and}$$
$$\alpha_{j''}^{(0)} \leq (1+\epsilon)\alpha_{j_1}^{(0)} \leq (1+\epsilon)\alpha_j \text{ for all } j'' \in N^{(0)}(i^\star)\,.$$

Proof. By Invariant 4, every client must be decided in $(\alpha^{(0)}, z^{(0)})$. Consider client j_1. If j_1 was witnessed in $(\alpha^{(0)}, z^{(0)})$, then there was a tight facility $i^\star$ such that $d(j_1, i^\star) \leq \bar{\alpha}_{j_1}^{(0)} \leq \bar{\alpha}_j$ and $\alpha_{j''}^{(0)} \leq (1+\epsilon)\alpha_{j_1}^{(0)}$ for every $j'' \in N^{(0)}(i^\star)$. If j_1 was stopped by

a client j_2 in $(\alpha^{(0)}, z^{(0)})$ (i.e., $2\bar{\alpha}_{j_1}^{(0)} \geq d(j_1, j_2) + 6\bar{\alpha}_{j_2}^{(0)}$), then we may assume that j_2 is witnessed by Lemma 3.7.1. In this case, let $i^\star$ be the witness of j_2. Then,

$$d(j_1, i^\star) \leq d(j_1, j_2) + d(j_2, i^\star) \leq d(j_1, j_2) + \bar{\alpha}_{j_2}^{(0)} \leq 2\bar{\alpha}_{j_1}^{(0)} \leq 2\bar{\alpha}_j \,,$$

and also

$$\alpha_{j''}^{(0)} \leq (1+\epsilon)\alpha_{j_2}^{(0)} \leq \alpha_{j_1}^{(0)} \leq \alpha_j,$$

for all $j'' \in N^{(0)}(i^\star)$. In either case, the claim holds. □

Now, let $i^\star$ be the facility guaranteed to exist by the Claim. Consider the dual solution $\alpha^{(p)}$ at the last time that j' was previously increased. Then, we must have $\alpha_{j'}^{(p)} \geq \alpha_{j'}$. Additionally, since Algorithm 1 never decreases any facility's price, and the current call to RaisePrice has increased any facility's price by at most ϵ_z, we have $z_{i^\star}^{(p)} \leq z_{i^\star} \leq z_{i^\star}^{(0)} + \epsilon_z$. Let $j^\star = \arg\min_{j'' \in N^{(0)}(i^\star)} \alpha_{j''}^{(p)}$. We claim that:

$$\alpha_{j^\star}^{(p)} = \min_{j'' \in N^{(0)}(i^\star)} \alpha_{j''}^{(p)} \leq (1+\epsilon)\alpha_j + \epsilon_z \,. \tag{3.8.3}$$

Indeed, otherwise by the Claim, we would have $\alpha_{j''}^{(p)} > (1+\epsilon)\alpha_j + \epsilon_z \geq \alpha_{j''}^{(0)} + \epsilon_z$ for every $j'' \in N^{(0)}(i^\star)$. Then, since $i^\star$ is tight in $(\alpha^{(0)}, z^{(0)})$ we would have:

$$\begin{aligned}\sum_{j'' \in \mathcal{D}} [\alpha_{j''}^{(p)} - d(j'', i^\star)]^+ &\geq \sum_{j'' \in N^{(0)}(i^\star)} [\alpha_{j''}^{(p)} - d(j'', i^\star)]^+ \\ &> \sum_{j'' \in N^{(0)}(i^\star)} [\alpha_{j''}^{(0)} + \epsilon_z - d(j'', i^\star)]^+ \geq z_{i^\star}^{(0)} + \epsilon_z \geq z^{(p)} \,,\end{aligned}$$

contradicting Invariant 2.

We shall now show that (3.8.3) and the claim imply (3.8.2). Since j' was increasing when

$\alpha^{(p)}$ was maintained, Rule 2 of SWEEP implies that:

$$\begin{aligned}
2\bar{\alpha}_{j'} &\leq 2\bar{\alpha}_{j'}^{(p)} \leq d(j', j^\star) + 6\bar{\alpha}_{j^\star}^{(p)} && (j' \text{ was last increased in } \alpha^{(p)}) \\
&\leq d(j', j) + d(j, j_1) + d(j_1, i^\star) + d(i^\star, j^\star) + 6\bar{\alpha}_{j^\star}^{(p)} && (\text{triangle inequality}) \\
&\leq d(j', j) + d(j, j_1) + d(j_1, i^\star) + \bar{\alpha}_{j^\star}^{(0)} + 6\bar{\alpha}_{j^\star}^{(p)} && (j^\star \in N^{(0)}(i^\star)) \\
&\leq d(j', j) + d(j, j_1) + d(j_1, i^\star) + \bar{\alpha}_{j^\star}^{(0)} + 12\bar{\alpha}_j && (\text{inequality (3.8.3)}) \\
&\leq d(j', j) + d(j, j_1) + 2\bar{\alpha}_j + (1+\epsilon)^{1/2}\bar{\alpha}_j + 12\bar{\alpha}_j && (j^\star \in N^{(0)}(i^\star) \text{ and Claim above}) \\
&\leq d(j', j) + d(j, j_1) + 16\bar{\alpha}_j && (\text{arithmetic}) \\
&\leq d(j', j) + d(j, i) + d(i, j_1) + 16\bar{\alpha}_j && (\text{triangle inequality}) \\
&\leq d(j', j) + \bar{\alpha}_j + \bar{\alpha}_{j_1} + 16\bar{\alpha}_j && (j, j_1 \in N(i)) \\
&\leq d(j', j) + 18\bar{\alpha}_j\,. && (\alpha_j \geq \alpha_{j_1} \text{ since } j \text{ decreasing})
\end{aligned}$$

and thus (3.8.2) remains satisfied when j is decreasing. □

Using Lemma 3.8.8, we can now prove that RAISEPRICE produces a close sequence of solutions, and also bound the total number of clients in $\mathcal{B}$ for any solution produced by RAISEPRICE. For both of these tasks, we make use of the following auxiliary lemma, which is a consequence of Lemma 3.8.8.

Lemma 3.8.9. *Throughout stage s, $\alpha_j \leq \alpha_j^{(0)}$ for all j with $\alpha_j > \theta_s$ and if $\alpha_j^{(0)} \geq 20^2\theta_s$ or $\alpha_j \geq 20^2\theta_s$ then $\alpha_j = \alpha_j^{(0)}$ for all j.*

Proof. For the first claim, we show that any client α_j with $\alpha_j \geq \theta_s$ can continue to increase in stage s only while $\alpha_j < \alpha_j^{(0)}$. Indeed, if $\alpha_j \geq \theta_s$ then once $\alpha_j = \alpha_j^{(0)}$, j will immediately be removed from A by Rule 4.

For the remaining claim, suppose first that $\alpha_j^{(0)} \geq 20^2\theta_s$. Suppose further, towards contradiction, that $\alpha_j < \alpha_j^{(0)}$ at some moment in stage s or earlier, and let $\alpha^{(-)}$ be the value of α at this time. Then, at some moment in stage s or earlier, we must have had $\alpha_j^{(-)} < \alpha_j < \alpha_j^{(0)}$, and $\alpha_j \geq 19^2\theta_s$ but j decreasing. Since j is being decreased by SWEEP at this moment, we must have $j \in N(i)$ for a potentially tight facility i. Since $\alpha_j^{(0)} > \alpha_j$ we must also have $j \in N^{(0)}(i)$. However, Lemma 3.8.8 implies that for every other $j' \in N(i)$ at this moment we have $\alpha_{j'} \geq 19^{-2}\alpha_j \geq \theta_s$. Thus, by the first claim, $\alpha_{j'} \leq \alpha_{j'}^{(0)}$ for all $j' \in N(i)$. This contradicts the fact that i is potentially tight, since $j \in N^{(0)}(i)$ with $\alpha_j < \alpha_j^{(0)}$.

Finally, suppose that $\alpha_j \geq 20^2\theta_s$. Then, by the first claim, we must have $\alpha_j \leq \alpha_j^{(0)}$ and so also $\alpha_j^{(0)} \geq 20^2\theta_s$. Then, as we have just shown, $\alpha_j = \alpha_j^{(0)}$. □

RaisePrice produces a close sequence of α-values in polynomial time

In the preceding section, we showed that all of the α-values are relatively stable throughout the algorithm. Using those observations, we can now prove that RaisePrice indeed produces a close sequence of α-values. To that end, let us select the remaining parameters K, σ, and ϵ_z used in RaisePrice.

Recall that the thresholds used by RaisePrice are defined by:

$$\theta_1 = (\max_{j \in U^{(0)}} \alpha_j^{(0)} + 2\epsilon_z)(1+\epsilon)^\sigma \quad \text{and} \quad \theta_s = (1+\epsilon)^K \theta_{s-1} \text{ for all } s > 1.$$

Therefore, the ratio of two consecutive thresholds is $\theta_s/\theta_{s-1} = (1+\epsilon)^K$. We select K to be the smallest integer satisfying

$$(1+\epsilon)^K \geq C_0^{2/\gamma^4}, \quad \text{where } C_0 = 81 \cdot 25 \cdot 20^8.$$

Note that $K = \Theta(\epsilon^{-1}\gamma^{-4})$. Given K, we select an integer "shift" σ uniformly at random from the interval $(0, K/2]$. This completes the definition of our thresholds. Finally, we set the price increment ϵ_z to:

$$\epsilon_z = n^{-6(K+C_1+2)-3} \qquad \text{where } C_1 = \lceil \log_{1+\epsilon}(20^4) \rceil + 1 = O(\epsilon^{-1}). \tag{3.8.4}$$

Using these parameters, we can show that the sequence of solutions (α, z) produced by RaisePrice is indeed close. Because each successive α-value in this sequence is produced by calling Sweep on the previous value, it suffices to show the following.

Proposition 3.8.10. *Each call to* Sweep *changes every α_j by at most n^{-2}.*

Proof. Consider a call to Sweep performed in stage s. By the definition of Sweep, it suffices to bound how much α_j has changed at the moment it is removed from A, since it is not subsequently changed. Let us begin by bounding how much any α_j may be increased. As in our analysis of QuasiSweep, it will then be possible to bound how much any α-value is decreased. Let $\alpha^{(1)}$ be the value of α before this call to Sweep, and let $\mu = \min_{j \in U} \alpha_j^{(1)}$. We first show the following:

Claim. *Any α_j can increase by at most $\epsilon_z n^{6(b+1)}$ while $B(\theta) \leq B(\mu) + b$.*

Proof. The proof is by induction on $b = -1, 0, 1, \ldots$.

Base case $b = -1$: Let us first show that this base case indeed occurs in any call to Sweep. Initially we have $\theta = 0$ and, by Invariant 2, $\mu \geq 1$. Thus, at the start of any call to Sweep, we must have $B(\theta) = 0$ and $B(\mu) \geq 1$. Now, note that while $B(\theta) \leq B(\mu) - 1$

we must have $\theta < \mu$. Then, by Property 3 of SWEEP no α-value has yet been altered, and so the claim holds trivially.

Inductive step ($b \geq 0$): Now suppose that some α_j is increased by at least ϵ_z while $B(\theta) \leq B(\mu) + b$. Otherwise, the claim is immediate since $\epsilon_z < \epsilon_z n^{6(b+1)}$. Note that while this α_j is increasing we must also have $\alpha_j = \theta$ and so $B(\alpha_j) \leq B(\mu) + b$.

First, suppose that $\alpha_j < \alpha_j^{(1)}$. Then, α_j was previously decreased. Moreover, since α_j was increased by at least ϵ_z while $B(\theta) \leq B(\mu) + b$, we must have previously decreased α_j while $B(\alpha_j) \leq B(\mu) + b$. In particular, at the last moment α_j was decreased, we must have had $B(\alpha_j) \leq\leq B(\mu) + b$, and since α_j was decreasing at this moment, we also had $B(\theta) < B(\alpha_j)$. Therefore, α_j was decreased only while $B(\theta) < B(\mu) + b$. Moreover, during this time, j's α-value was decreased at most $|A| \leq n$ times the amount that any other client's α-value was increased. By the induction hypothesis, any client's α-value can increase at most $\epsilon_z n^{6b}$ while $B(\theta) < B(\mu) + b$. Thus, α_j has decreased at most $\epsilon_z \cdot n^{6b+1}$, and after increasing α_j by at most this amount, we will again have $\alpha_j = \alpha_j^{(1)}$.

Next, let us bound how much j's α-value may increase while $\alpha_j \geq \alpha_j^{(1)}$ (and still $B(\alpha_j) \leq B(\mu) + b$). We now consider three cases, based on the initial status of j in $\alpha^{(1)}$.

If j is undecided initially, then $j \in U$ and α_j can increase by at most $\epsilon_z \leq \epsilon_z n^{6b}$ (since $b \geq 0$) before it is removed by Rule 3.

Next, suppose that j had some witness i in $\alpha^{(1)}$, and let $N^{(1)}(i)$ be the set of clients paying for i in $\alpha^{(1)}$. For each $j' \in N^{(1)}(i)$ we must have $B(\alpha_{j'}^{(1)}) \leq B(\alpha_j^{(1)}) \leq B(\mu) + b$, and so $\alpha_{j'}$ is decreased by SWEEP only while $B(\theta) \leq B(\mu) + b - 1$. By the same argument given above (when considering the case that $\alpha_j < \alpha_j^{(1)}$), the α-value of any such $j' \in N^{(1)}(i)$ can decrease at most $\epsilon_z n^{6b+1}$ during SWEEP. Thus, the total contribution to i can decrease at most $n \cdot \epsilon_z n^{6b+1} = \epsilon_z n^{6b+2}$ during SWEEP. After increasing α_j by at most this amount, i will again be tight. Moreover, at this moment any client j' contributing to i was either already added to A (and potentially also removed), in which case $B(\alpha_{j'}) \leq B(\theta) = B(\alpha_j)$, or it was not already added to A, in which case $B(\alpha_{j'}) \leq B(\alpha_{j'}^{(1)}) \leq B(\alpha_j^{(1)}) \leq B(\alpha_j)$. Thus, at this moment i is a witness for j, and so j will be removed from A by Rule 1.

Finally, suppose that j was initially stopped by some client j'. Then, by Lemma 3.7.1, we may assume that j' was *not* stopped. Let $\Delta = \bar{\alpha}_{j'} - \bar{\alpha}_{j'}^{(1)}$ be the amount that $\bar{\alpha}_{j'}$ has been increased by SWEEP. Then, once $\bar{\alpha}_j - \bar{\alpha}_j^{(1)} \geq 3\Delta$, we will have:

$$2\bar{\alpha}_j \geq 2\bar{\alpha}_j^{(1)} + 6(\bar{\alpha}_{j'} - \bar{\alpha}_{j'}^{(1)}) \geq d(j, j') + 6\bar{\alpha}_{j'},$$

where in the last inequality we have used the fact that j' stopped j in $\alpha^{(1)}$. Thus, $\bar{\alpha}_j$ can increase by at most 3Δ, before j will again be stopped by j' and removed from A by

Rule 2. It remains to bound the corresponding increases in α_j and $\alpha_{j'}$. We have:

$$\alpha_j - \alpha_j^{(1)} \leq \left(\bar{\alpha}_j^{(1)} + 3\Delta\right)^2 - \alpha_j^{(1)} = 6\Delta \cdot \bar{\alpha}_j^{(1)} + 9\Delta^2 \,.$$

Now, let us bound the right hand side. Since j' is not stopped, the previous cases show that $\alpha_{j'} - \alpha_{j'}^{(1)} \leq \epsilon_z n^{6b+2}$. Then, we have:

$$\Delta^2 = \left(\sqrt{\alpha_{j'}} - \sqrt{\alpha_{j'}^{(1)}}\right)^2 \leq \left(\sqrt{\alpha_{j'}^{(1)} + \epsilon_z n^{6b+2}} - \sqrt{\alpha_{j'}^{(1)}}\right)^2 \leq \epsilon_z n^{6b+2},$$

where the last inequality follows from $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for all $a, b \in \mathbb{R}_+$. On the other hand, since $g(x) = \sqrt{x}$ is a concave function of x for all $x > 0$, we have:

$$\Delta = \sqrt{\alpha_{j'}} - \sqrt{\alpha_{j'}^{(1)}} \leq g\left(\alpha_{j'}^{(1)} + \epsilon_z n^{6b+2}\right) - g\left(\alpha_{j'}^{(1)}\right) \leq \epsilon_z n^{6b+2} \cdot g'\left(\alpha_{j'}^{(1)}\right) = \frac{\epsilon_z n^{6b+2}}{2\sqrt{\alpha_{j'}^{(1)}}} \leq \frac{\epsilon_z n^{6b+2}}{2},$$

where the last inequality follows from Invariant 2, which implies $\alpha_{j'}^{(1)} \geq 1$. Combining the above bounds, in this case we have

$$\begin{aligned}\alpha_j - \alpha_j^{(1)} \leq 6\Delta \cdot \bar{\alpha}_j^{(1)} + 9\Delta^2 &\leq 6\bar{\alpha}_j^{(1)} \cdot \frac{\epsilon_z n^{6b+2}}{2} + 9\epsilon_z n^{6b+2} \\ &\leq 3\sqrt{5n^7} \cdot \epsilon_z n^{6b+2} + 9\epsilon_z n^{6b+2} \leq 9\epsilon_z n^{6b+11/2}\,,\end{aligned}$$

where the penultimate inequality follows from the feasibility invariant (Invariant 2) and the preprocessing of Lemma 3.4.1 (that all squared-distances are at most n^6) which together imply that $\alpha_j \leq \min_i(z_i + d(i,j)^2) \leq 4n^7 + n^6 \leq 5n^7$ for all $j \in \mathcal{D}$.

Combining all of the above cases, α_j can increase at most $\epsilon_z n^{6b+1}$, until $\alpha_j = \alpha_j^{(1)}$ and then at most an additional $9\epsilon_z n^{6b+11/2}$. Thus, the total increase in α_j while $B(\theta) \leq B(\mu) + b$ is at most $9\epsilon_z n^{6b+11/2} + \epsilon_z n^{6b+1} \leq \epsilon_z n^{6b+6}$, as required. □

We now complete the proof of Proposition 3.8.10. By Lemma 3.8.9, no $\alpha_j \geq 20^2\theta_s$ is changed by Sweep in any stage s, and so once $B(\theta) \geq B(20^2\theta_s)$ no α-values are changed. By the Claim, we then have that in any call to Sweep in stage s, each client's α-value is increased at most $\epsilon_z n^{6(b+1)}$ where

$$b = B(20^2\theta_s) - B(\mu) \leq \lfloor \log_{1+\epsilon}(20^2\theta_s/\mu) \rfloor + 1\,.$$

We now bound the above value b for every stage s.

In stage 1, we execute only a single call to Sweep (as shown in Lemma 3.8.3) and in this call, $\mu = \min_{j \in U^{(0)}} \alpha_j^{(0)}$. Since every $j \in U^{(0)}$ must have a tight edge to the facility

i^+ in $\alpha^{(0)}$, Lemma 3.8.8 implies that $\nu \triangleq \max_{j \in U^{(0)}} \alpha_j^{(0)} \leq 20^2\mu$. Then, recall that

$$\theta_1 = (\nu + 2\epsilon_z)(1+\epsilon)^\sigma \leq \nu(1+\epsilon)^K \leq 20^2\mu(1+\epsilon)^K,$$

where we have used that $\nu \geq 1$ (by Invariant 2), $\epsilon_z < \epsilon$ and $\sigma \leq K/2 < K$. Finally, recalling that $C_1 = \lceil \log_{1+\epsilon}(20^4) \rceil$, we have:

$$b \leq \log_{1+\epsilon}(20^2\theta_1/\mu) + 1 \leq \log_{1+\epsilon}(20^2 \cdot 20^2 \cdot (1+\epsilon)^K) + 1 \leq K + C_1 + 1\,.$$

In stage $s > 1$, we have $\mu \geq \theta_{s-1}$ by Lemma 3.8.5. Then, recall that $\theta_s = (1+\epsilon)^K\theta_{s-1}$ Then, we have:

$$b \leq \log_{1+\epsilon}(20^2\theta_s/\mu) + 1 \leq \log_{1+\epsilon}(20^2(1+\epsilon)^K) + 1 < K + C_1 + 1\,.$$

In any case, the maximum increase in any client's α-value is at most $\epsilon_z n^{6(K+C_1+2)} = n^{-3}$ (recalling that by definition $\epsilon_z = n^{-6(K+C_1+2)-3}$). As we have already observed above in the proof of the Claim, each α-value can decrease at most n times this amount. Thus, no α-value can decrease more than n^{-2}. □

Bounding the number of clients in $\mathcal{B}$

We bound the number of clients in $\mathcal{B}$ by showing that such clients need to have an $\alpha^{(0)}$-value close to a threshold θ_s. We then select the thresholds so that only a tiny fraction of the clients can be in $\mathcal{B}$.

Lemma 3.8.11. *Suppose that $j \in \mathcal{B}$ for some (α, z) produced by* RaisePrice. *Then, we must have $\frac{1}{81}\theta_s \leq \alpha_j^{(0)} < 25 \cdot 20^4\theta_s$ for some s.*

Proof. Consider a call to RaisePrice and let $(\alpha^{(0)}, z^{(0)})$ be the input solution. We denote by $(\alpha^{(\ell)}, z^{(\ell)})$ the solution produced by the ℓth call to Sweep in the execution of RaisePrice. We also use $\mathcal{B}^{(\ell)}$ to refer to the set $\mathcal{B}$ of potentially bad clients associated to the solution $(\alpha^{(\ell)}, z^{(\ell)})$. That is,

$$\mathcal{B}^{(\ell)} = \{j \in \mathcal{D} :\ j \text{ is undecided in } (\alpha^{(\ell)}, z^{(\ell)}) \text{ and } 2\bar{\alpha}_j^{(\ell)} < d(j,j') + 6\bar{\alpha}_{j'}^{(0)} \text{ for all clients } j' \in \mathcal{D}\}.$$

Note that $\mathcal{B}^{(0)} = \emptyset$ since every client is decided in $(\alpha^{(0)}, z^{(0)})$ by Invariant 4. We prove the lemma by showing the following claim by induction on ℓ:

Claim. *For each client $j \in \mathcal{B}^{(\ell)}$ there is a client j' such that $\bar{\alpha}_j^{(\ell)} \geq d(j,j') + \bar{\alpha}_{j'}^{(0)}$ and $\frac{1}{9}\theta_s \leq \alpha_{j'}^{(0)} \leq 20^4\theta_s$, for some s.*

Before proving the claim, let us show that it indeed implies the Lemma. Suppose that

$j \in \mathcal{B}^{(\ell)}$ for some ℓ. Then, selecting $j' = j$ in the definition of $\mathcal{B}^{(\ell)}$, we must have $2\bar{\alpha}_j^{(\ell)} < 6\bar{\alpha}_j^{(0)}$ and so $\alpha_j^{(\ell)} < 9\alpha_j^{(0)}$. Now, consider the client j' and stage s guaranteed by the claim. Then, we must have:

$$\alpha_j^{(0)} \geq \tfrac{1}{9}\alpha_j^{(\ell)} \geq \tfrac{1}{9}\alpha_{j'}^{(0)} \geq \tfrac{1}{81}\theta_s.$$

Moreover, because $j \in \mathcal{B}(\ell)$, j is undecided in $(\alpha^{(\ell)}, z^{(\ell)})$ and so by Lemma 3.8.4, $\alpha_j^{(0)} \leq \alpha_j^{(\ell)}$. Also, we must have:

$$2\bar{\alpha}_j^{(\ell)} < d(j,j') + 6\bar{\alpha}_{j'}^{(0)} \leq d(j,j') + \bar{\alpha}_{j'}^{(0)} + \sqrt{25 \cdot 20^4 \cdot \theta_s} \leq \bar{\alpha}_j^{(\ell)} + \sqrt{25 \cdot 20^4 \cdot \theta_s}.$$

Thus, $\alpha_j^{(0)} \leq \alpha_j^{(\ell)} \leq 25 \cdot 20^4 \theta_s$, as well. □

Proof of the Claim. The base case when $\ell = 0$ is trivially true since $\mathcal{B}^{(0)} = \emptyset$. For the inductive step, we assume the induction hypothesis for $0, 1, \ldots, \ell - 1$ and prove it for ℓ. Any client $j \in \mathcal{B}^{(\ell)}$ is undecided and (by Property 1 of Sweep) must have been removed from A by one of the Rules 3, 4, or 5. We divide the analysis based on these three cases.

Case 1: $j \in \mathcal{B}^{(\ell)}$ was removed by Rule 3. In this case, we must have $j \in U$, by the definition of Rule 3. Moreover, since all clients in $U^{(0)}$ are decided in every solution produced by RaisePrice (Corollary 3.8.6), we must have that $\ell \geq 2$. Thus, j must have been undecided in the previously produced solution $(\alpha^{(\ell-1)}, z^{(\ell-1)})$, and $\alpha_j^{(\ell)} = \alpha_j^{(\ell-1)} + \epsilon_z$. Then, $j \in \mathcal{B}^{(\ell-1)}$, as well, since

$$2\bar{\alpha}_j^{(\ell-1)} < 2\bar{\alpha}_j^{(\ell)} < d(j,j') + 6\bar{\alpha}_{j'}^{(0)} \qquad \text{for all } j' \in \mathcal{D}.$$

The statement then follows from the induction hypothesis and from that $\alpha_j^{(\ell)} \geq \alpha_j^{(\ell-1)}$.

Case 2: $j \in \mathcal{B}^{(\ell)}$ was removed by Rule 4. By the definition of Rule 4, we must have $\alpha_j^{(\ell)} \geq \alpha_j^{(0)}$ and $\alpha_j^{(\ell)} \geq \theta_s$, where s is the stage in which $\alpha^{(\ell)}$ was produced. In this case, we prove the claim for $j' = j$. Clearly, we have $\bar{\alpha}_j^{(\ell)} \geq \bar{\alpha}_j^{(0)} = d(j,j) + \bar{\alpha}_j^{(0)}$. Next, we prove that $\frac{1}{9}\theta_s \leq \alpha_j^{(0)} \leq 20^4\theta_s$. For the lower bound, we observe that since $j \in \mathcal{B}^{(\ell)}$ we must have $2\bar{\alpha}_j^{(\ell)} < 6\bar{\alpha}_j^{(0)}$ and so $\frac{1}{9}\theta_s \leq \frac{1}{9}\alpha_j^{(\ell)} \leq \alpha_j^{(0)}$. We now prove the upper bound. First, note that j was not stopped by any client j' in $(\alpha^{(0)}, z^{(0)})$ since then we would have $2\bar{\alpha}_j^{(\ell)} \geq 2\bar{\alpha}_j^{(0)} \geq d(j,j') + 6\bar{\alpha}_{j'}^{(0)}$ which would contradict that $j \in \mathcal{B}^{(\ell)}$. Then, since every client in $(\alpha^{(0)}, z^{(0)})$ is decided (Invariant 4), j must have been witnessed by some facility i in $(\alpha^{(0)}, z^{(0)})$. Since j is undecided, by Corollary 3.8.6 we may further assume $i \neq i^+$ and thus $z_i^{(\ell)} = z_i^{(0)}$. Now suppose toward contradiction that $\alpha_j^{(0)} > 20^4\theta_s$. Lemma 3.8.8 then implies $\alpha_{j'}^{(0)} \geq 20^{-2}\alpha_j^{(0)} > 20^2\theta_s$ for every $j' \in N^{(0)}(i)$. Then, by Lemma 3.8.9, $\alpha_{j'} = \alpha_{j'}^{(0)}$

for all $j' \in N^{(0)}(i)$. Thus, as $z_i = z_i^{(0)}$, i is still tight and a witness for i, which contradicts the assumption that $j \in \mathcal{B}^{(\ell)}$, since j is then decided.

Case 3: $j \in \mathcal{B}^{(\ell)}$ was removed by Rule 5. Let $j_1, j_2, \ldots, j_p$ be the clients in $\mathcal{B}^{(\ell)}$ that were removed from A by Rule 5 in the ℓth call to SWEEP. We index these clients according to the order in which they were removed from A. The previous cases already imply that the clients in $\mathcal{B}^{(\ell)} \setminus \{j_1, \ldots, j_p\}$ satisfy the induction hypothesis. We now assume that it is true for the clients in $\mathcal{B}^{(\ell)}_{<a} := \mathcal{B}^{(\ell)} \setminus \{j_a, \ldots, j_p\}$ and show that it is also true for client j_a, i.e., for all clients in $\mathcal{B}^{(\ell)} \setminus \{j_{a+1}, \ldots, j_p\}$. Consider client j_a. Then, we must have $\bar{\alpha}_{j_a}^{(\ell)} \geq d(j_a, j') + \bar{\alpha}_{j'}^{(\ell)}$ for some j' that was previously removed from A. Moreover, since j_a is undecided, Property 2 implies that j' must also be undecided. We now show that $j' \in \mathcal{B}^{(\ell)}$. Indeed, since j' is undecided, if $j' \notin \mathcal{B}^{(\ell)}$, there must be a j'' such that $2\bar{\alpha}_{j'}^{(\ell)} \geq d(j', j'') + 6\bar{\alpha}_{j''}^{(0)}$. But then

$$2\bar{\alpha}_{j_a}^{(\ell)} \geq 2d(j_a, j') + 2\bar{\alpha}_{j'}^{(\ell)} \geq 2d(j_a, j') + d(j', j'') + 6\bar{\alpha}_{j''}^{(0)} \geq d(j_a, j'') + 6\bar{\alpha}_{j''}^{(0)},$$

which contradicts the fact that $j_a \in \mathcal{B}^{(\ell)}$. Now, since $j' \in \mathcal{B}^{(\ell)}$ was removed from A before j_a, we in fact have $j' \in \mathcal{B}^{\ell}_{<a}$, and so by assumption there exists some j'' and s such that $\frac{1}{9}\theta_s \leq \alpha_{j''}^{(0)} \leq 20^4\theta_s$ and $\bar{\alpha}_{j'}^{(\ell)} \geq d(j', j'') + \bar{\alpha}_{j''}^{(0)}$. Thus

$$\bar{\alpha}_{j_a}^{(\ell)} \geq d(j_a, j') + \bar{\alpha}_{j'}^{(\ell)} \geq d(j_a, j') + d(j', j'') + \bar{\alpha}_{j''}^{(0)} \geq d(j_a, j'') + \bar{\alpha}_{j''}^{(0)},$$

and so the induction hypothesis holds for j_a as well (using j'' and s).

□

The above lemma says that any client j that becomes potentially bad in any solution produced by RAISEPRICE (i.e., in $j \in \mathcal{B}$ for one produced solution) must have $\alpha_j^{(0)}$ close to a threshold. Our selection of the shift-parameter σ then ensures that this can only happen for a tiny fraction of the clients. This allows us to bound the connection cost of clients in $\mathcal{B}$ by an arbitrarily small constant fraction (depending on the parameter K) of $\sum_{j \in \mathcal{D}} \alpha_j$. However, as stated in the second inequality of Property 2 in Definition 3.5.1, we need to bound the total connection cost of these clients as a tiny fraction of OPT_k instead of $\sum_{j \in \mathcal{D}} \alpha_j$. As all we know is that $\mathrm{OPT}_k \geq \sum_{j \in \mathcal{D}} \alpha_j - \lambda k$, this requires further arguments that we present in the next section.

We complete this section by formally showing that a client is unlikely to become potentially bad over the randomness of the shift-parameter σ. For any given integer $\sigma \in [0, K/2)$, let

$$\mathcal{W}(\sigma) = \{j \in \mathcal{D} : \ 81^{-1} \cdot 20^{-2} \cdot \theta_s \leq \alpha_j^{(0)} \leq 25 \cdot 20^6 \cdot \theta_s \text{ for some } \theta_s\}.$$

Note that by the above lemma, any client that is in $\mathcal{B}$ in any solution produced during

the considered call to RaisePrice, is in $\mathcal{W}(\sigma)$.[8] Note that each value $\alpha_j^{(0)}$ is fixed at the beginning of RaisePrice, and there are only a (relatively) small number of choices for σ such that any given j is in $\mathcal{W}(\sigma)$. Thus, if we choose σ uniformly at random, the probability that any given $j \in \mathcal{W}(\sigma)$ is small. The following corollary formalizes this intuition.

Corollary 3.8.12. *If we select the shift-parameter σ uniformly at random from $[0, K/2)$,*

$$\Pr[j \in \mathcal{W}(\sigma)] \leq \gamma^4 \qquad \textit{for any client } j.$$

Proof. Suppose that we select an integer σ uniformly at random from $[0, K/2)$. Then, note that by definition $\theta_s = (\max_{j \in U^{(0)}} \alpha_j + 2\epsilon_z)(1+\epsilon)^{K\cdot(s-1)+\sigma}$ and so $j \in \mathcal{W}(\sigma)$ if and only if:

$$K_1 + K(s-1) + \sigma \leq \log_{1+\epsilon} \alpha_j^{(0)} \leq K_2 + K(s-1) + \sigma\,,$$

for some s, where $K_1 = \log_{1+\epsilon}(81^{-1} \cdot 20^{-2}) + \log_{1+\epsilon}(\max_{j \in U^{(0)}} \alpha_j^{(0)} + 2\epsilon_z)$ and $K_2 = \log_{1+\epsilon}(25 \cdot 20^6) + \log_{1+\epsilon}(\max_{j \in U^{(0)}} \alpha_j^{(0)} + 2\epsilon_z)$. In other words, σ needs to satisfy

$$K_1 + K(s-1) - \log_{1+\epsilon} \alpha_j^{(0)} \leq -\sigma \leq K_2 + K(s-1) - \log_{1+\epsilon} \alpha_j^{(0)} \text{ for some integer } s.$$

Notice that the difference between the upper bound and the lower bound is $K_2 - K_1 = \log_{1+\epsilon}(81 \cdot 25 \cdot 20^8)$ which by selection of K and C_0 is at most $\frac{\gamma^4}{2}K$. Moreover, as $\sigma \in [0, K/2)$ there is at most one value of s that can satisfy the above inequalities. It follows that there are at most $\frac{\gamma^4}{2}K$ distinct values of σ so that $j \in \mathcal{W}(\sigma)$. Thus, $j \in \mathcal{W}(\sigma)$ with probability at most γ^4. □

3.8.5 Handling dense clients

Corollary 3.8.12 implies that by carefully selecting our thresholds, we can ensure that only an arbitrarily small fraction γ^4 of clients j can ever appear in $\mathcal{B}$ throughout the execution of RaisePrice. As briefly discussed previously, this is unfortunately insufficient for our purposes. Specifically, in order to charge the extra service cost incurred by this small fraction of clients to $\mathrm{OPT}_k \geq \sum_{j \in \mathcal{D}} \alpha_j - \lambda k$, we need to handle carefully those clients j for which most of α_j is contributing toward the opening cost λk.

To cope with this difficulty, we introduce the notion *dense* facilities and clients, as follows. Recall that $\gamma \ll \epsilon$ is a small constant. We define the *γ-close neighborhood* of a facility i

[8] To argue $\mathcal{B} \subseteq \mathcal{W}(\sigma)$, the bounds $81^{-1}\theta_s \leq \alpha_j^{(0)} \leq 25 \cdot 20^4 \cdot \theta_s$ for some θ_s would be sufficient in the definition of $\mathcal{W}(\sigma)$. However, the more relaxed bounds will be useful when analyzing dense clients in the next section.

as

$$N_\gamma^{(0)}(i) = \{j \in \mathcal{D} : d(j,i)^2 \leq \gamma \alpha_j^{(0)}\}.$$

Then, we say that a facility $i \in \mathsf{IS}^{(0)}$ is *dense* if

$$\sum_{j \in N_\gamma^{(0)}(i)} \alpha_j^{(0)} \geq (1-\gamma) z_i^{(0)}.$$

We let $\mathcal{F}_\mathsf{D} \subseteq \mathsf{IS}^{(0)}$ be the set of all dense facilities, and then define the set of *dense clients* as $\mathcal{D}_\mathsf{D} = \bigcup_{i \in \mathcal{F}_\mathsf{D}} N_\gamma^{(0)}(i)$. Note that the γ-close neighborhoods, dense facilities, and dense clients are all determined only by the input solution $(\alpha^{(0)}, z^{(0)})$ and the integral solution $\mathsf{IS}^{(0)}$ passed to RaisePrice.

Intuitively, the dense clients are precisely those troublesome clients for which $\alpha_j^{(0)}$ is much larger then the service cost of j in $\mathsf{IS}^{(0)}$. In order to avoid paying $36\alpha_j^{(0)}$ for any such clients, we construct a set of *special facilities* $\mathcal{F}_\mathsf{S}^{(\ell)}$ and *special clients* $\mathcal{D}_\mathsf{S}^{(\ell)}$ for each $\alpha^{(\ell)}$ produced by our RaisePrice, as follows. Let

$$\begin{aligned} \mathcal{F}_\mathsf{S}^{(\ell)} &= \{i \in \mathcal{F}_\mathsf{D} : |N_\gamma^{(0)}(i) \cap \mathcal{B}| \neq \emptyset \text{ and } \alpha_j^{(\ell)} \geq \alpha_j^{(0)} \ \forall j \in N_\gamma^{(0)}(i)\}, \quad \text{and} \\ \mathcal{D}_\mathsf{S}^{(\ell)}(i) &= N_\gamma^{(0)}(i) \quad \text{for every } i \in \mathcal{F}_\mathsf{S}^{(\ell)}, \end{aligned} \tag{3.8.5}$$

We will show that each solution $\mathcal{S}^{(\ell)} = (\alpha^{(\ell)}, z^{(\ell)}, \mathcal{F}_\mathsf{S}^{(\ell)}, \mathcal{D}_\mathsf{S}^{(\ell)})$ is roundable, with the set of remaining bad clients given by $\mathcal{D}_\mathsf{B} = \mathcal{B} \setminus \mathcal{D}_\mathsf{D}$.

Recall that in Definition 3.5.1, we have $\tau_i = \max_{j \in N(i) \cap \mathcal{D}_\mathsf{S}(i)} \alpha_j$ for any facility $i \in \mathcal{F}_\mathsf{S}$ and $\tau_i = t_i$ for all other facilities. Note that as $(\alpha^{(0)}, z^{(0)})$ by Invariant 4 is a completely decided solution, by our choice of $\mathcal{F}_\mathsf{S}$ and $\mathcal{D}_\mathsf{S}$, we have $\mathcal{F}_\mathsf{S}^{(0)} = \emptyset$ in the roundable solution $(\alpha^{(0)}, z^{(0)}, \mathcal{F}_\mathsf{S}^{(0)}, \mathcal{D}_\mathsf{S}^{(0)})$. Therefore, the conflict graph $H^{(0)}$ of $(\alpha^{(0)}, z^{(0)}, \mathcal{F}_\mathsf{S}^{(0)}, \mathcal{D}_\mathsf{S}^{(0)})$ does not contain any special facilities, and so $\tau_i = t_i$ for each facility $i \in H^{(0)}$. Moreover, recall that $\mathsf{IS}^{(0)}$ was the maximal independent set of $H^{(0)}$ computed in the previous call to GraphUpdate. In particular, $|\mathsf{IS}^{(0)}| > k$ and $\mathsf{IS}^{(0)}$ does not contain any special facilities.

The following simple lemma is now a direct consequence of our definitions.

Lemma 3.8.13. *Suppose that $j \in N_\gamma^{(0)}(i)$ for some $i \in \mathsf{IS}^{(0)}$. Then, $\beta_{i'j}^{(0)} = 0$ for all other $i' \in \mathsf{IS}^{(0)}$. Moreover, for every client $j \in \mathcal{D}$, $\alpha_j^{(0)} \geq \sum_{i \in \mathsf{IS}^{(0)}} \beta_{ij}^{(0)}$.*

Proof. We start by proving that if $j \in N_\gamma^{(0)}(i)$ for some $i \in \mathsf{IS}^{(0)}$, then $\beta_{i'j}^{(0)} = 0$ for all other $i' \in \mathsf{IS}^{(0)}$. Consider some facility $i \in \mathsf{IS}^{(0)}$, and suppose that $j \in N_\gamma^{(0)}(i)$. Further, suppose that for some other facility $i' \in H^{(0)}$ we have $\beta_{i'j} > 0$. Note that this implies (since no facility is special) that j is adjacent to both i and i' in the client-facility graph

that generated $H^{(0)}$. We shall show that $i' \notin \mathsf{IS}^{(0)}$. Indeed, we must have:

$$d(i,i') \le d(i,j) + d(j,i') < \sqrt{\gamma} \cdot \bar{\alpha}_j^{(0)} + \bar{\alpha}_j^{(0)} < \sqrt{\delta} \cdot \bar{\alpha}_j^{(0)} \le \sqrt{\delta t_i} = \sqrt{\delta \tau_i}\,.$$

Thus, there is an edge between i and i' in the conflict graph $H^{(0)}$, and since $i \in \mathsf{IS}^{(0)}$, we have $i' \notin \mathsf{IS}^{(0)}$.

We shall now prove that $\alpha_j^{(0)} \ge \sum_{i \in \mathsf{IS}^{(0)}} \beta_{ij}^{(0)}$ for any client $j \in \mathcal{D}$. Again using that no facility is special, we have that j's neighborhood in the client-facility graph that generated $H^{(0)}$ is equal to the set of tight facilities that j is paying for. Therefore, we have that $\alpha_j^{(0)} - \sum_{i \in \mathsf{IS}^{(0)}} \beta_{ij}^{(0)} \ge d(j, \mathsf{IS}^{(0)})^2/\rho$ (which implies $\alpha_j^{(0)} \ge \sum_{i \in \mathsf{IS}^{(0)}} \beta_{ij}^{(0)}$) by the exact same arguments as "Case $s = 1$" and "Case $s > 1$" in the proof of Theorem 3.3.4. □

The next lemma formalizes our intuition, showing that we can indeed charge the total $\alpha^{(0)}$-value of all non-dense clients to OPT_k.

Lemma 3.8.14. $\sum_{j \in \mathcal{D} \setminus \mathcal{D}_D} \alpha_j^{(0)} \le \gamma^{-3} \cdot \mathrm{OPT}_k$.

Proof. We partition the clients in $\mathcal{D} \setminus \mathcal{D}_{\mathsf{D}}$ into two sets:

$$\mathcal{D}_{>\gamma} = \{j \in \mathcal{D} \setminus \mathcal{D}_{\mathsf{D}} : d(j, \mathsf{IS}^{(0)})^2 > \gamma \cdot \alpha_j^{(0)}\} \text{ and}$$
$$\mathcal{D}_{\le\gamma} = \{j \in \mathcal{D} \setminus \mathcal{D}_{\mathsf{D}} : d(j, \mathsf{IS}^{(0)})^2 \le \gamma \cdot \alpha_j^{(0)}\}.$$

By definition,

$$\sum_{j \in \mathcal{D}_{>\gamma}} d(j, \mathsf{IS}^{(0)})^2 > \gamma \cdot \sum_{j \in \mathcal{D}_{>\gamma}} \alpha_j^{(0)}\,. \tag{3.8.6}$$

To bound the remaining clients consider the following fractional token argument: each client $j \in \mathcal{D}_{>\gamma}$ distributes $\beta_{ij}^{(0)} = [\alpha_j^{(0)} - d(j,i)^2]^+$ tokens to each facility $i \in \mathsf{IS}^{(0)}$. Lemma 3.8.13 says that $\sum_{i \in \mathsf{IS}^{(0)}} \beta_{ij}^{(0)} \le \alpha_j^{(0)}$ for every client j, and so the total number of tokens distributed is at most $\sum_{j \in \mathcal{D}_{>\gamma}} \alpha_j^{(0)}$.

Now, we observe that for each client $j \in \mathcal{D}_{\le\gamma}$, the closest facility $i \in \mathsf{IS}^{(0)}$ to j must *not* be in $\mathcal{F}_{\mathsf{D}}$, since otherwise j would be in $\mathcal{D}_{\mathsf{D}}$. Hence, we have

$$\sum_{j \in N_\gamma^{(0)}(i)} (1-\gamma)\alpha_j^{(0)} \le \sum_{j \in N_\gamma^{(0)}(i)} [\alpha_j^{(0)} - d(j,i)^2]^+ \le (1-\gamma) z_i^{(0)}\,. \tag{3.8.7}$$

Moreover, there must be at least $\gamma z_i^{(0)}$ tokens assigned to i, because it is a tight facility with respect to $\alpha^{(0)}$ and by Lemma 3.8.13, every client $j \notin \mathcal{D}_{>\gamma} \cup N_\gamma^{(0)}(i)$ must have

$\beta_{ij} = 0$. Therefore,

$$\sum_{j \in \mathcal{D}_{>\gamma}} \beta_{ij}^{(0)} = \sum_{j \in \mathcal{D} \setminus N_\gamma^{(0)}(i)} \beta_{ij}^{(0)} \geq \gamma z_i^{(0)} .$$

Thus,

$$\begin{aligned} \sum_{j \in \mathcal{D}_{\leq \gamma}} \alpha_j^{(0)} &= \sum_{i \in \mathsf{IS}^{(0)} \setminus \mathcal{F}_\mathsf{D}} \sum_{j \in N_\gamma^{(0)}(i)} \alpha_j^{(0)} \leq \sum_{i \in \mathsf{IS}^{(0)} \setminus \mathcal{F}_\mathsf{D}} z_i^{(0)} \\ &= \frac{1}{\gamma} \sum_{i \in \mathsf{IS}^{(0)} \setminus \mathcal{F}_\mathsf{D}} \gamma \cdot z_i^{(0)} \leq \frac{1}{\gamma} \sum_{j \in \mathcal{D}_{>\gamma}} \alpha_j^{(0)} , \end{aligned}$$

where the first equality follows from Lemma 3.8.13, the first inequality from (3.8.7), and the last inequality from the fact that each facility $i \in \mathsf{IS}^{(0)} \setminus \mathcal{F}_\mathsf{D}$ received at least $\gamma z_i^{(0)}$ tokens and the total amount of distributed tokes was at most $\sum_{j \in \mathcal{D}_{>\gamma}} \alpha_j^{(0)}$.

Hence,

$$\begin{aligned} \sum_{j \in \mathcal{D}_{>\gamma}} \alpha_j^{(0)} + \sum_{j \in \mathcal{D}_{\leq \gamma}} \alpha_j^{(0)} &\leq \left(1 + \frac{1}{\gamma}\right) \sum_{j \in \mathcal{D}_{>\gamma}} \alpha_j^{(0)} \\ &< \frac{1}{\gamma} \cdot \left(1 + \frac{1}{\gamma}\right) \sum_{j \in \mathcal{D}_{>\gamma}} d(j, \mathsf{IS}^{(0)})^2 \\ &\leq \frac{1}{\gamma} \cdot \left(1 + \frac{1}{\gamma}\right) (\rho + 1000\epsilon) \mathrm{OPT}_k , \end{aligned}$$

where the penultimate inequality follows from (3.8.6) and the last inequality from Theorem 3.6.4. □

Lemma 3.8.14 shows that we can relate the total α-value of all non-dense clients to OPT_k, as desired. Moreover, we have the following corollary.

Corollary 3.8.15. *If we select the shift-parameter σ uniformly at random from $[0, K/2)$,*

$$\mathbb{E}\left[\sum_{j \in \mathcal{W}(\sigma) \setminus \mathcal{D}_D} \alpha_j^{(0)}\right] \leq \gamma \cdot \mathrm{OPT}_k .$$

In particular, if we set $\mathcal{W} = \mathcal{W}(\sigma)$ for the value σ that minimizes $\sum_{j \in \mathcal{W}(\sigma) \setminus \mathcal{D}_D} \alpha_j^{(0)}$ then, we have

$$\sum_{j \in \mathcal{W} \setminus \mathcal{D}_D} \alpha_j^{(0)} \leq \gamma \cdot \mathrm{OPT}_k .$$

Proof. For the first claim, we note that

$$\mathbb{E}\left[\sum_{j\in\mathcal{W}(\sigma)\setminus\mathcal{D}_{\mathsf{D}}}\alpha_j^{(0)}\right] = \sum_{j\in\mathcal{D}\setminus\mathcal{D}_{\mathsf{D}}}\Pr[j\in\mathcal{W}(\sigma)]\cdot\alpha_j^{(0)} \le \gamma^4\sum_{j\in\mathcal{D}\setminus\mathcal{D}_{\mathsf{D}}}\alpha_j^{(0)} \le \gamma\cdot\mathrm{OPT}_k\,,$$

where the first inequality follows from by Corollary 3.8.12 and the last inequality follows from Lemma 3.8.14. The second claim now follows since the minimum of left-hand side over all $\sigma\in[0,K/2)$ is at most its expected value over a randomly chosen $\sigma\in[0,K/2)$. □

Remark 3.8.16. *The only property of the selection of σ that we use is that $\sum_{j\in\mathcal{W}\setminus\mathcal{D}_D}\alpha_j^{(0)} \le \gamma\cdot\mathrm{OPT}_k$. It easy to find the σ that minimizes $\sum_{j\in\mathcal{W}(\sigma)\setminus\mathcal{D}_D}\alpha_j^{(0)}$ (since the number of options is constant) at the start of a call to* RaisePrice *and thus the selection of the shift-parameter can be derandomized.*

Now, we show how to obtain a better bound than that given by Lemma 3.8.7 for dense clients $\mathcal{D}_{\mathsf{D}}\cap\mathcal{B}$. This will allow us to eventually obtain a roundable solution. For this purpose, recall the definition of *special* facilities and clients $\mathcal{F}_{\mathsf{S}}$ and $\mathcal{D}_{\mathsf{S}}$:

$$\mathcal{F}_{\mathsf{S}} = \{i\in\mathcal{F}_{\mathsf{D}} : |N_\gamma^{(0)}(i)\cap\mathcal{B}|\neq\emptyset \text{ and } \alpha_j\ge\alpha_j^{(0)}\ \forall j\in N_\gamma^{(0)}(i)\} \quad \text{and}$$
$$\mathcal{D}_{\mathsf{S}}(i) = N_\gamma^{(0)}(i) \quad \text{for every } i\in\mathcal{F}_{\mathsf{S}}.$$

Also, recall that for each special facility $i\in\mathcal{F}_{\mathsf{S}}$ we define $\tau_i = \max_{j\in N(i)\cap\mathcal{D}_{\mathsf{S}}(i)}\alpha_j$ and for all other facilities i we let $\tau_i = t_i = \max_{j\in N(i)}\alpha_j$. Note that $\tau_i\le t_i$ for all $i\in\mathcal{F}$. We now show how to bound the cost of all clients in $\mathcal{D}_{\mathsf{D}}\cap\mathcal{B}$ using the facilities of $\mathcal{F}_{\mathsf{S}}$.

Lemma 3.8.17. *For any $j\in\mathcal{D}_D\cap\mathcal{B}$, either:*

- *There exists a tight facility $i\in\mathcal{F}$ such that $(1+\sqrt{\delta}+10\epsilon)\bar{\alpha}_j \ge d(j,i)+\sqrt{\delta t_i}$.*
- *There exists a special facility $i\in\mathcal{F}_S$ such that $(1+\sqrt{\delta}+10\epsilon)\bar{\alpha}_j \ge d(j,i)+\sqrt{\delta\tau_i}$.*

Proof. Consider a client $j_0\in\mathcal{D}_{\mathsf{D}}\cap\mathcal{B}$. Since $j_0\in\mathcal{D}_{\mathsf{D}}$ there must be some $i^\star\in\mathcal{F}_{\mathsf{D}}$ such that $j_0\in N_\gamma^{(0)}(i^\star)$. Moreover, since $j_0\in\mathcal{B}$, j_0 is undecided and so by Lemma 3.8.4 we must have $\alpha_{j_0}\ge\alpha_{j_0}^{(0)}$.

Suppose first that $i^\star\in\mathcal{F}_{\mathsf{S}}$. Then $\tau_{i^\star} = \max_{j\in N(i^\star)\cap\mathcal{D}_{\mathsf{S}}(i^\star)}\alpha_j$. Since $i^\star\in\mathcal{F}_{\mathsf{S}}$ we have $\alpha_j\ge\alpha_j^{(0)}$ for all $j\in N_\gamma^{(0)}(i^\star)$. We claim that $\tau_{i^\star}\le(1+\epsilon)^2\alpha_{j_0}$. Indeed, otherwise there

is a client $j \in N(i^\star) \cap \mathcal{D}_\mathsf{S}(i^\star) = N(i^\star) \cap N_\gamma^{(0)}(i^\star)$ such that $\bar{\alpha}_j > (1+\epsilon)\bar{\alpha}_{j_0}$ and so

$$\begin{aligned}
(1+\epsilon)\bar{\alpha}_j &> (1+\epsilon)\bar{\alpha}_{j_0} + \tfrac{\epsilon}{2}\cdot\bar{\alpha}_{j_0} + \tfrac{\epsilon}{2}\cdot\bar{\alpha}_j \\
&\geq (1+\epsilon)\bar{\alpha}_{j_0} + \tfrac{\epsilon}{2}\cdot\bar{\alpha}_{j_0}^{(0)} + \tfrac{\epsilon}{2}\cdot\bar{\alpha}_j^{(0)} && (\alpha_{j_0} \geq \alpha_{j_0}^{(0)} \text{ and } \alpha_j \geq \alpha_j^{(0)}) \\
&\geq (1+\epsilon)\bar{\alpha}_{j_0} + \tfrac{\epsilon}{2\sqrt{\gamma}}\cdot d(j_0, i^\star) + \tfrac{\epsilon}{2\sqrt{\gamma}}\cdot d(j, i^\star) && (j, j_0 \in N_\gamma^{(0)}(i^\star)) \\
&\geq (1+\epsilon)\bar{\alpha}_{j_0} + (1+\epsilon)d(j_0, i^\star) + (1+\epsilon)d(j, i^\star) && (\gamma \ll \epsilon \text{ and so } \tfrac{\epsilon}{2\sqrt{\gamma}} \geq (1+\epsilon)) \\
&\geq (1+\epsilon)\Big(\bar{\alpha}_{j_0} + d(j_0, j)\Big)\,,
\end{aligned}$$

contradicting Invariant 3, since the α-ball of j would then strictly contain the α-ball of j_0. Hence, $\sqrt{\tau_{i^\star}} \leq (1+\epsilon)\bar{\alpha}_{j_0}$. Furthermore, $d(j_0, i^\star) \leq \gamma\bar{\alpha}_{j_0}^{(0)} \leq \gamma\bar{\alpha}_{j_0}$ and therefore

$$(1 + \sqrt{\delta} + 3\epsilon)\bar{\alpha}_{j_0} \geq d(j_0, i^\star) + \sqrt{\delta\tau_{i^\star}}\,.$$

On the other hand, if $i^\star \notin \mathcal{F}_\mathsf{S}$ then (by the definition of $\mathcal{F}_\mathsf{S}$) there must be some $j \in N_\gamma^{(0)}(i^\star)$ with $\alpha_j < \alpha_j^{(0)}$. By Lemma 3.8.4, j must be decided. Then, $j \notin \mathcal{B}$ and so by Lemma 3.8.7 there exists some tight facility i such that $(1 + \sqrt{\delta} + \epsilon)\bar{\alpha}_j \geq d(j, i) + \sqrt{\delta t_i}$. Moreover, applying the same argument as above, we must have $\bar{\alpha}_j^{(0)} \leq (1+\epsilon)\bar{\alpha}_{j_0}^{(0)}$, since otherwise in $\alpha^{(0)}$, the α-ball of j would strictly contain the α-ball of j_0, contradicting Invariant 3. Then, we have:

$$\begin{aligned}
d(j_0, i) + \sqrt{\delta t_i} &\leq d(j_0, i^\star) + d(i^\star, j) + d(j, i) + \sqrt{\delta t_i} \\
&\leq \gamma\bar{\alpha}_{j_0}^{(0)} + \gamma\bar{\alpha}_j^{(0)} + (1 + \sqrt{\delta} + \epsilon)\bar{\alpha}_j \\
&\leq \epsilon\bar{\alpha}_{j_0}^{(0)} + \epsilon(1+\epsilon)\bar{\alpha}_{j_0}^{(0)} + (1 + \sqrt{\delta} + \epsilon)(1+\epsilon)\bar{\alpha}_{j_0}^{(0)} \\
&< (1 + \sqrt{\delta} + 10\epsilon)\bar{\alpha}_{j_0}\,,
\end{aligned}$$

where for the final inequality we used that j_0 is undecided and so by Lemma 3.8.4 we have $\alpha_{j_0} \geq \alpha_{j_0}^{(0)}$. □

3.8.6 Showing that each solution is roundable and completing the analysis

We start by showing that each solution (α, z) produced by Algorithm 1 satisfies the properties of Definition 3.5.1.

Proposition 3.8.18. *Every solution (α, z) produced by Algorithm 1 is roundable.*

Proof. By construction, each solution (α, z) produced by Algorithm 1 is feasible with respect to $\mathsf{DUAL}(\lambda + \epsilon_z)$ and $\epsilon_z < \frac{1}{n}$. In addition, we have $\lambda \leq z_i \leq \lambda + \epsilon_z \leq \lambda + 1/n$ for all $i \in \mathcal{F}$. It remains to show that Properties 2 and 3 of Definition 3.5.1 are satisfied. Recall the definitions of $\mathcal{F}_\mathsf{D}$ and $\mathcal{D}_\mathsf{D}$, and define $\mathcal{F}_\mathsf{S}$ and $\mathcal{D}_\mathsf{S}$ as in (3.8.5). Further let

$\mathcal{D}_{\mathsf{B}} = \mathcal{W} \setminus \mathcal{D}_{\mathsf{D}}$.

Now we show that Property 2 holds for $\mathcal{S} = (\alpha, z, \mathcal{F}_{\mathsf{S}}, \mathcal{D}_{\mathsf{D}})$ with respect to the set $\mathcal{D}_{\mathsf{B}}$. By Lemma 3.8.11 (which shows that a client $j \in \mathcal{B}$ only if $\frac{1}{81}\theta_s \leq \alpha_j^{(0)} \leq 25 \cdot 20^4 \theta_s$ for some stage s) we have $\mathcal{B} \subseteq \mathcal{W}$. Thus $\mathcal{B} \setminus \mathcal{D}_{\mathsf{D}} \subseteq \mathcal{D}_{\mathsf{B}}$. Now, by Lemma 3.8.7 for all $j \in \mathcal{D} \setminus \mathcal{B}$ there exists some tight facility i such that

$$(1 + \sqrt{\delta} + 10\epsilon)^2 \alpha_j \geq \left(d(j,i) + \sqrt{\delta t_i}\right)^2 \geq \left(d(j,i) + \sqrt{\delta \tau_i}\right)^2 .$$

Similarly, by Lemma 3.8.17, for all $j \in \mathcal{B} \cap \mathcal{D}_{\mathsf{D}}$, there exists either some tight facility i or some special facility $i \in \mathcal{F}_{\mathsf{S}}$ such that

$$(1 + \sqrt{\delta} + 10\epsilon)^2 \alpha_j \geq \left(d(j,i) + \sqrt{\delta \tau_i}\right)^2 .$$

Finally, for each remaining client $j \in \mathcal{B} \setminus \mathcal{D}_{\mathsf{D}} \subseteq \mathcal{D}_{\mathsf{B}}$, by Lemma 3.8.7, there is some tight facility i such that

$$36\alpha_j^{(0)} \geq \left(d(j,i) + \sqrt{\delta t_i}\right)^2 \geq \left(d(j,i) + \sqrt{\delta \tau_i}\right)^2 .$$

For each such client j, let $w(j)$ be this specified tight facility i. Then,

$$\sum_{j \in \mathcal{D}_{\mathsf{B}}} \left(d(j,i) + \sqrt{\delta \tau_{w(j)}}\right)^2 \leq 36 \sum_{j \in \mathcal{D}_{\mathsf{B}}} \alpha_j^{(0)} \leq 36\gamma \cdot \mathrm{OPT}_k,$$

where the last inequality follows from Corollary 3.8.15.

Finally, we show that Property 3 must hold. Consider some $i \in \mathcal{F}_{\mathsf{S}}$. By definition of $\mathcal{F}_{\mathsf{S}}$, we must have $j \in \mathcal{B}$ for some $j \in N_\gamma^{(0)}(i)$. Then, by Lemma 3.8.11 we must have $\frac{1}{81}\theta_s \leq \alpha_j^{(0)} \leq 25 \cdot 20^4 \theta_s$ for some s. By Lemma 3.8.8 (which bounds the ratio to be at most $19^2 \leq 20^2$ between α_j and $\alpha_{j'}$ for any pair of clients j, j' that share a tight edge to some common facility i), we must have $\alpha_{j'}^{(0)} \in \mathcal{W}$ for any $j' \in N^{(0)}(i)$. Moreover, by Lemma 3.8.13 (which shows that each dense client pays for at most one dense facility in $\mathsf{IS}^{(0)}$), we have $\beta_{ij}^{(0)} = 0$ for all $j \in \mathcal{D}_{\mathsf{D}} \setminus N_\gamma^{(0)}(i)$. Altogether, then we have $N^{(0)}(i) \subseteq \mathcal{W}$ and $N^{(0)}(i) \cap \mathcal{D}_{\mathsf{D}} = N_\gamma^{(0)}(i)$ and so

$$N^{(0)}(i) \setminus N_\gamma^{(0)}(i) = N^{(0)}(i) \setminus \mathcal{D}_{\mathsf{D}} \subseteq \mathcal{W} \setminus \mathcal{D}_{\mathsf{D}} ,$$

for every $i \in \mathcal{F}_{\mathsf{S}}$. Notice that Lemma 3.8.13 also implies that for each dense facility $i \in \mathcal{F}_{\mathsf{D}}$ there is some dense client that only pays for that facility in $\mathsf{IS}^{(0)}$, so indeed $|\mathcal{F}_{\mathsf{S}}| \leq |\mathcal{F}_{\mathsf{D}}| \leq n$. It remains to prove that $\sum_{i \in \mathcal{F}_{\mathsf{S}}} \sum_{j \in \mathcal{D}_{\mathsf{S}}} \beta_{ij} \geq \lambda |\mathcal{F}_{\mathsf{S}}| - \gamma \cdot \mathrm{OPT}_k$. To that end, recall that Invariant 4 implies that $\mathcal{F}_{\mathsf{S}}^{(0)} = \emptyset$. Thus, every $i \in \mathsf{IS}^{(0)}$ must have been tight in $\alpha^{(0)}$, and hence $\sum_{j \in N^{(0)}(i)} \beta_{ij}^{(0)} \geq z_i^{(0)} \geq \lambda$. Combining these observations, for

every $i \in \mathcal{F}_{\mathsf{S}}$ we have:

$$\sum_{j \in \mathcal{D}_{\mathsf{S}}(i)} \beta_{ij} = \sum_{j \in N_\gamma^{(0)}(i)} \beta_{ij} \geq \sum_{j \in N_\gamma^{(0)}(i)} \beta_{ij}^{(0)} \geq \lambda - \sum_{j \in N^{(0)}(i) \setminus N_\gamma^{(0)}(i)} \beta_{ij}^{(0)} \geq \lambda - \sum_{j \in \mathcal{W} \setminus \mathcal{D}_{\mathsf{D}}} \beta_{ij}^{(0)}\,, \tag{3.8.8}$$

where the first inequality follows from the definition of $\mathcal{F}_{\mathsf{S}}$, which requires that for any $i \in \mathcal{F}_{\mathsf{S}}$, $\alpha_j \geq \alpha_j^{(0)}$ for all $j \in N_\gamma^{(0)}(i)$. By Invariant 4, every client is decided in $\alpha^{(0)}$ and so, in particular, $\mathcal{F}_{\mathsf{S}}^{(0)} = \emptyset$ and $\mathsf{IS}^{(0)}$ contains no special facilities. Then, by Lemma 3.8.13, $\sum_{i \in \mathcal{F}_{\mathsf{S}}} \beta_{ij}^{(0)} \leq \sum_{i \in \mathsf{IS}^{(0)}} \beta_{ij}^{(0)} \leq \alpha_j^{(0)}$ for all j. Summing (3.8.8) over all $i \in \mathcal{F}_{\mathsf{S}}$ we thus have

$$\sum_{i \in \mathcal{F}_{\mathsf{S}}} \sum_{j \in \mathcal{D}_{\mathsf{S}}(i)} \beta_{ij} \geq |\mathcal{F}_{\mathsf{S}}|\lambda - \sum_{j \in \mathcal{W} \setminus \mathcal{D}_{\mathsf{D}}} \sum_{i \in \mathcal{F}_{\mathsf{S}}} \beta_{ij}^{(0)} \geq |\mathcal{F}_{\mathsf{S}}|\lambda - \sum_{j \in \mathcal{W} \setminus \mathcal{D}_{\mathsf{D}}} \alpha_j^{(0)} \geq |\mathcal{F}_{\mathsf{S}}|\lambda - \gamma \cdot \mathrm{OPT}_k,$$

where the final inequality follows from Corollary 3.8.15. □

The following theorem completes the analysis.

Theorem 3.8.19. RaisePrice *runs in polynomial time and produces a polynomial number of close roundable solutions.*

Proof. That the produced solutions are close follows from Proposition 3.8.10 and the produced solutions are roundable follows from Proposition 3.8.18. We continue to bound the running time and the number of produced solutions. RaisePrice produces one solution for each call to Sweep. In Appendix 3.9 we argue (similarly as we did for QuasiSweep) that Sweep can be implemented in polynomial time, and it is clear that the remaining operations in RaisePrice can be implemented in polynomial time. Thus, to prove both claims, it suffices bound the number of calls to Sweep in RaisePrice. For that purpose, define:

$$M = \lambda + \max_{j \in \mathcal{D}, i \in \mathcal{F}} d(j,i) \leq 4n^7 + n^6 < n^8\,.$$

Using our preprocessing (Lemma 3.4.1), we note that at all times during any call to RaisePrice, we have $\alpha_j \leq M$, since otherwise α would be infeasible (contradicting Invariant 2).

Let us now bound the number of calls to Sweep in each stage. In stage 1, we make only 1 call to Sweep, as shown in Lemma 3.8.3. In each stage $s > 1$, RaisePrice calls Sweep only until $\alpha_j \geq \theta_s$ and $\alpha_j \geq \alpha_j^{(0)}$ for every undecided client j. Consider a call to Sweep in stage $s > 1$ and let (α, z) be the produced solution. Let j be the undecided client with the smallest α-value in (α, z) (breaking ties in the order of removal from the set A). If j was removed by Rule 4, we have $\alpha_j \geq \theta_s$ and $\alpha_j \geq \alpha_j^{(0)}$ and so every undecided client has an α-value of at least θ_s which implies the termination of stage s. Otherwise,

as j has the smallest α-value of undecided clients it cannot be removed by Rule 5 (by Property 2) and so it must have been removed by Rule 3. Therefore, by the definition of that rule, j was undecided in the previous iteration and its α-value has increased by ϵ_z in the considered call to Sweep. By the above, we have that either stage s terminates or the smallest α-value of the undecided clients increases by at least ϵ_z. Therefore, the stage must terminate after at most $\epsilon_z^{-1}M = n^{O(\epsilon^{-1}\gamma^{-4})}$ calls to Sweep since no α-value is larger than M.

Finally, let us bound the number of stages executed in RaisePrice. By Lemma 3.8.5 after stage s, all clients with $\alpha_j < \theta_s$ are decided. Then, for $s = (K\epsilon)^{-1}\Theta(\log n) = \gamma^4 O(\log n)$ we have $\theta_s > M$ and so all clients must be decided. □

3.9 Running Time Analysis of Sweep

In this section, we present a polynomial time implementation of the Sweep procedure. The algorithm is exactly the same as QuasiSweep, besides the set of events. Recall that the polynomial time algorithm for QuasiSweep is as follows: repeatedly find the next event that happens, then update the α-values. We increase θ at a rate 1, so that θ corresponds naturally to our notion of time. Let $\theta^{(0)}$ denote the value of θ at the time that the last event has happened.

We now focus on the events, explain them in detail, show the number of times that each can occur, and discuss the way that we can find each one of them. We consider (1) the tight edges, (2) the set A, (3) the set of potentially tight facilities, denoted by P, (4) the set of clients with maximum α-value in $N(i)$ for each facility in $i \in P$, and (5) the buckets of clients and θ. While all the above quantities remain the same, the rate at which any client's α-value is changing remains constant. We now consider the set of events that may cause these quantities to change. We show that we can easily compute the next time that each event would occur if no other event happened before it, and that the number of such events must be polynomial in $n = |\mathcal{D}|$ and $m = |\mathcal{F}|$. Then, the minimum such time is the next time that the behavior of Sweep might change. We compute the minimum such time after $\theta^{(0)}$, then update all of the α-values (using their current rates of change and the time of this event) to obtain the α-values at this time. Having these values, we then recompute the quantities (1)-(5) in the following order:

- Update the α-values: For each $j \in \mathcal{D}$, we compute the new value of α_j by multiplying the previous rate of change in α_j by $\theta - \theta^{(0)}$ and adding this to the previous value α_j.
- The buckets of the α-values: For each α_j, we update the current bucket according to the bucketing described in Sweep. In the case that some α_j was decreasing and now α_j is on the lower border of the current bucket, we place α_j in the next lower

bucket.

- Update A: Now, for each $j \notin A$, if $\alpha_j = \theta$ we add j to A. Then, for each $j \in A$, we remove j if one of the conditions of Rules 1-5 in SWEEP is satisfied by the newly updated α-values. Notice that a client j might be added and then immediately removed in this step.

- Update the set of tight edges: For each $j \in \mathcal{D}$ and $i \in \mathcal{F}$, if $\alpha_j > d(j,i)^2$ then add j to $N(i)$. If $\alpha_j \leq d(j,i)^2$ then we remove j from $N(i)$ unless $j \in A$. If $\alpha_j = d(j,i)^2$ and $j \in A$, then we add j to $N(i)$.

- Update P: For each $i \in \mathcal{F}$, we compute whether i is potentially tight as follows. If $\alpha_j > \alpha_j^{(0)}$ for any $j \in N(i)$, then place $i \in P$. If $\alpha_j = \alpha_j^{(0)}$ for some $j \in N(i) \cap A$ then we also place $i \in P$, since as soon as α_j increases by an infinitesimal amount, we will have $\alpha_j > \alpha_j^{(0)}$. No other facilities are placed in P.

  Notice that the above cases capture all the potentially tight facilities as described in the analysis of SWEEP except for the following. We could have some facility i with $\alpha_j \leq \alpha_j^{(0)}$ for all $j \in N(i)$ and $\alpha_j = \alpha_j^{(0)}$ for all $j \in N^{(0)}(i)$ but there is no $j \in N(i) \cap A$ with $\alpha_j = \alpha_j^{(0)}$. Then, our definition of SWEEP would consider i potentially tight, but here we do not. However, observe that in this case $N(i) \subseteq N^{(0)}(i)$ and so we must have that $N(i) \cap A = \emptyset$. Therefore, i cannot cause any α-value to decrease in SWEEP, so there is no harm in having $i \notin P$.

- Update rates: For each $j \in A$, we set the rate of change for α_j to be 1, and for each j with $\alpha_j = t_i$ for some $i \in P$, we set the rate of change of α_j to be $-|A|$ if and only if $N(i) \cap A \neq \emptyset$. For all other clients j, we set the rate of change for α_j to 0.

Now, we focus on the events that may require updating the above values. Throughout our discussion we use the fact that once a client's α-value has been increased by SWEEP, it is not subsequently decreased.

First we focus on (1), i.e., on the tight edges. The following two events take into account when an edge becomes tight and when an edge becomes untight.

- Event 1: The edge between client j and facility i becomes tight. Since a client's α-value is never decreased after it has been increased, this event can happen at most once for each client-facility pair. Notice that this event can occur only when the edge (i,j) is not tight and $j \in A$, so α_j is increasing. Then, since $j \in A$, this event happens when $d(i,j)^2 = \alpha_j = \theta$.

- Event 2: The edge between client j and facility i becomes untight. Similar to the previous event, this can also happen at most once for each j and i. Notice that this event can occur only when the edge (i,j) is tight and the α_j value is

decreasing. Then, α_j is decreasing at a rate of $|A|$, and so the event happens at time θ satisfying $|A|(\theta - \theta^{(0)}) = \alpha_j - d(i,j)^2$.

Second, we focus on (2), i.e., the changes that can happen to set A. Notice that, A changes only if a client joins it or leaves it. Therefore, we have the following events.

- Event 3: A client j joins A. This event happens exactly once for each client. For each client $j \notin A$, it happens when $\theta = \alpha_j$.
- Event 4: A client $j \in A$ is removed from A. By the description for the algorithm it happens in one of following five situations, and they can happen at most once for each client (so in total n times):
  - j gets a witness i: This can happen if j gets an edge to an already tight facility i, or i becomes tight with $B(t_i) \leq B(\alpha_j)$. In Event 1 we have discussed the situation in which a client gets a tight edge to a facility, therefore here we only focus on the time that facility i becoming tight. Note that the facility i might remove j from A only if $B(t_i) \leq B(\alpha_j)$, so we only consider such facilities in this case. Notice that these facilities are easy to identify because, when considering this event, we assume that no client or θ change bucket (those events are considered separately) and therefore $B(t_i)$ and $B(\alpha_j)$ stay constant. A facility i can only become tight if there is no client in its neighborhood that is decreasing, so we also restrict ourselves to such facilities (which are again easy to identify).

    The time θ, that i becomes tight can now be calculated by solving the following equation:

    $$z_i = \sum_{j \in \mathcal{D} \setminus A} [\alpha_j - d(i,j)^2]^+ + \sum_{j \in A} [\theta - d(i,j)^2]^+ .$$

  - j is stopped by some client j'. This can only happen if $j \in A$ and $j' \notin A$. The time θ that this event happens can be computed by solving the equation $2\sqrt{\theta} = d(j,j') + 6\bar{\alpha}_{j'}$.
  - For a client $j \in U$, α_j increases by ϵ_z. The time θ that this event happens is $\theta = \alpha_j^{(1)} + \epsilon_z$ where $\alpha_j^{(1)}$ denotes the α-value of j at the beginning of the present call to Sweep.
  - $\alpha_j \geq \alpha_j^{(0)}$ and $\alpha_j \geq \theta_s$. This happens at the maximum time that each of the inequalities becomes tight. That is, at the time $\theta = \max(\alpha_j^{(0)}, \theta_s)$.
  - There is a client j' that has already been removed from A such that $\bar{\alpha}_j \geq d(j,j') + \bar{\alpha}_{j'}$. This case is similar to the case that j is stopped by j'. That is, the time it happens can be calculated by solving the equation $\sqrt{\theta} = d(j,j') + \bar{\alpha}_{j'}$.

Now we focus on (3), i.e., the set of potentially tight facilities. In the following we explore the events that make facilities become potentially tight.

- Event 5: A facility i becomes potentially tight. This can only happen if one of the following situations occur:
  - A new client j joins $N(i)$: This is equivalent to saying that the edge (i, j) becomes tight. We have already considered this case.
  - α_j becomes more than $\alpha_j^{(0)}$: This can happen only if α_j is increasing. Moreover, since $\alpha_j^{(0)}$ is fixed throughout the call to Sweep, this might happen at most once for each client (using that if an α-value increases, it will not decrease later on). For any such j, time θ that this event happens is easy to compute: $\theta = \alpha_j^{(0)}$.
  - For all $j \in N^{(0)}(i)$, $\alpha_j \geq \alpha_j^{(0)}$: We can compute for each $j \in N^{(0)}(i)$ the time that this inequality becomes tight, i.e., $\theta = \alpha_j^{(0)}$. This event can occur only if α_j is increasing for some $j \in N^{(0)}(i)$ with $\alpha_j < \alpha_j^{(0)}$, and in this case it may happen at time $\theta = \alpha_j^{(0)}$. Again, this event might happen at most n times, since no α-value is decreased after being increased.
- Event 6: A facility is no longer potentially tight. This event is similar to the previous one, except we consider the clients that are decreasing.

Finally we focus on (4) and (5) in Event 7 and Event 8, respectively.

- Event 7: A client j becomes the maximum client connected to a facility $i \in P$. We are interested in this case since it might result in decreasing α_j. We decrease α_j only if $B(\theta) < B(\alpha_j)$, so we do not need to consider this event if $j \in A$. For the remaining clients, this event can only happen if either the previous maximum client j' loses its tight edge to i or $\alpha_j = \alpha_{j'}$. We have already considered the case that tight edges change, so we focus on the time that $\alpha_j = \alpha_{j'}$. The time θ that the equality happens can be computed as

  $$\alpha_j = -|A|(\theta^{(0)} - \theta) + \alpha_{j'} .$$

  Notice that above we assume that α_j is not already decreasing. This is without loss of generality, since if α_j is already decreasing then it cannot become equal to $\alpha_{j'}$, as $\alpha_{j'}$ is decreasing by at most the same rate as α_j (namely, $-|A|$).

  This event can happen at most once for each client-facility pair assuming that the set P of potentially tight facilities and the set A of active clients do not change. The reason is as follows. If $\alpha_j > \theta$ *becomes* the maximum α-value of all clients in $N(i)$ for some potentially tight facility i, it was not previously decreasing. Moreover, while

α_j does not decrease, it remains the maximum α-value of all clients in $N(i)$. On the other hand, if it starts decreasing then it will not stop decreasing until at least one of the sets A or P has changed. In this case j will not become the maximum α-value for any facility (for which it is not already the maximum). Therefore, as P and A both change polynomially many times, this event can also happen at most polynomially many times.

- Event 8: A client enters the bucket of θ or θ changes bucket. We can compute the times of these events exactly as discussed in the analysis of the running time of QuasiSweep. The number of occurrences of these events is the same as before and the time that they happen can also be computed similarly.

The above shows that we can calculate the next event in polynomial time and that there are in total at most polynomially many events. It follows that Sweep can be implemented to run in time that is polynomial in the number of clients and facilities.

3.10 Bounding the Distances

Lemma 3.10.1. *By losing a factor* $(1+100/n^2)$ *in the approximation guarantee, we can assume that the squared-distance between any client and any facility is in* $[1, n^6]$*, where* $n = |\mathcal{D}|$.

Proof. We prove that for a given instance of the k-means problem, $\mathcal{I} = (\mathcal{F}, \mathcal{D}, d, k)$, we can in polynomial time output an instance $\mathcal{I}' = (\mathcal{F}, \mathcal{D}, d', k)$ such that:

- The squared distance between any client and any facility is in $[1, n^6]$ in $\mathcal{I}'$, i.e., for any $i \in \mathcal{F}, j \in \mathcal{D}$, we have $1 \leq d'(i,j)^2 \leq n^6$.
- For any constant ρ, any ρ-approximate solution for $\mathcal{I}'$ is a $\rho(1+100/n^2)$-approximate solution for $\mathcal{I}$.

In what follows, we first prove the lemma for the case that d is a metric distance function, then we prove it for the case that d is a Euclidean metric function.

Metric Distance: We focus on the case that d is a metric distance. To that end, we create 3 instances $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}'$ with distances d_1, d_2, d' respectively. Choose M, such that $\mathrm{OPT}(\mathcal{I}) \leq M \leq 100\mathrm{OPT}(\mathcal{I})$. We can use the algorithm presented in [33] to find such M. First, let $d_1(i,j) = \sqrt{\frac{n^3}{M}} d(i,j)$ for all $i \in \mathcal{F}, j \in \mathcal{D}$. This results in $\mathrm{OPT}(\mathcal{I}_1) = \mathrm{OPT}(\mathcal{I})\frac{n^3}{M}$, so $n^3/100 \leq \mathrm{OPT}(\mathcal{I}_1) \leq n^3$. Second, for any $i \in \mathcal{F}, j \in \mathcal{D}$ let $d_2(i,j) = \min(d_1(i,j), n^2)$. Consider any constant ρ-approximate solution, for $\mathcal{I}_1$. This solution cannot use any

of the edges that we updated in the previous step, since the cost of this edge is more than $n^4 \geq n\text{OPT}(\mathcal{I}_1)$. Similarly, any ρ-approximate solution for $\mathcal{I}_2$ cannot use any such edge. Therefore, $\text{OPT}(\mathcal{I}_2) = \text{OPT}(\mathcal{I}_1)$. Third, for any $i \in \mathcal{F}, j \in \mathcal{D}$, assign $d'(i,j) = \max(d_2(i,j), 1)$. Since this step might increase the cost of any solution by at most n, $\text{OPT}(\mathcal{I}_2) \leq \text{OPT}(\mathcal{I}') \leq \text{OPT}(\mathcal{I}_2) + n$. Now it is clear that for any $i \in \mathcal{F}, j \in \mathcal{D}$, $1 \leq d'(i,j)^2 \leq n^4$. We need to show that any good solution for $\mathcal{I}'$ is also a good solution for $\mathcal{I}$. Note that during all these steps, we focused on the distances between clients and facilities. For guaranteeing that the d' is metric, we make the exact same changes on the pairs of facilities and pairs of clients as well.

Consider a ρ-approximate solution for $\mathcal{I}'$. We know that the cost of this solution is at most $\rho \cdot OPT(\mathcal{I}')$. Now consider the same solution for $\mathcal{I}_2$. Since the cost of any solution for $\mathcal{I}_2$ is no more than its cost for $\mathcal{I}'$, the cost of this solution for $\mathcal{I}_2$ is at most $\rho \cdot OPT(\mathcal{I}') \leq \rho \cdot (OPT(\mathcal{I}_2) + n)$. Also the cost of the same solution for $\mathcal{I}_1$ equals to its cost for $\mathcal{I}_2$ so it is at most $\rho(OPT(\mathcal{I}_2)+n) = \rho(OPT(\mathcal{I}_1)+n) \leq \rho \cdot OPT(\mathcal{I}_1)(1+100/n^2)$, where the last inequality is due to the fact that $n^3/100 \leq \text{OPT}(\mathcal{I}_1)$. Thus the cost of the same solution for $\mathcal{I}$ is at most $\frac{M}{n^3}(\rho \cdot \text{OPT}(\mathcal{I}_1)(1 + 100/n^2)) = \rho(1 + 100/n^2) \cdot \text{OPT}(\mathcal{I})$. The lemma then follows by noting that d' is metric since we only rescaled, increased the minimum distance, and decreased the maximum distance of the given metric d.

Euclidean Metric Distance: Now assume that the given distance function is Euclidean. We assume that clients and facilities are points in some ℓ dimensional Euclidean space. We first create a solution $\mathcal{I}_1$, making sure that the $\text{OPT}(\mathcal{I}_1)$ is bounded by a polynomial. Similarly to before, we divide each coordinate by $\sqrt{\frac{n^3}{M}}$.[9] We get that $d_1(i,j) = \sqrt{\frac{n^3}{M}} d(i,j)$ for all $i \in \mathcal{F}, j \in \mathcal{D}$ and $n^3/100 \leq \text{OPT}(\mathcal{I}_1) \leq n^3$. Now we cluster the points in $\mathcal{D} \cup \mathcal{F}$ such that the distance between any two points in different clusters is more than $\Omega(n) \cdot \text{OPT}(\mathcal{I}_1)$. To do that, we create each cluster as follows: Pick any client j that is not part of any cluster and add it to cluster S; we call j the center of cluster S. While there exists a client $j' \in \mathcal{D}$ that is not part of any cluster and the distance between j' to its closest client in S is less than $n^2/4$ add j' to S. Let $S_1, \ldots, S_s$ be the clusters that we create. This gives a partition of the clients. Now we add a facility i to cluster S_ℓ, if there exists a client $j \in S_\ell$, such that $d(i,j) < n^2/8$. This ensures that each facility is at most part of one cluster, since the distance between two clients in different clusters is more than $n^2/4$.

It is easy to see that our clusters have the following properties.

1. $d_1(j,j')^2 < n^6/16$ for any two clients j, j' in the same cluster. The reason is, any client that we add to a cluster, the maximum distance in the cluster increases by

[9] We can still use [33] to find M

less than $n^2/4$ so $d_1(j, j') < n^3/4$.

2. $d_1(i, j)^2 \leq n^6/8$ for any client j and facility i in the same cluster. The reason is, by the triangle inequality we have that $d_1(i, j) \leq d_1(i, j_1) + d_1(j_1, j)$ where j_1 is the client so that $d_1(i, j_1) < n^2/4$. we also know that $d_1(j_1, j) < n^3/4$ by the previous property.

3. $d_1(i, i')^2 \leq n^6/8$ for any two facilities i, i' in the same cluster. Similarly to the previous case, let j, j' be the closest client in this cluster to i, i' respectively. By the triangle inequality we have that $d_1(i, i') \leq d_1(i, j) + d_1(j, j') + d_1(j', i') \leq n^2/4 + n^3/4 + n^2/4$.

4. $d_1(i, j)^2 \geq n^4/64 \geq (n/64) \cdot \mathrm{OPT}(\mathcal{I}_1)$ for any facility i and client j not in the same cluster.

We remove all the facilities that are not part of any cluster, since no client can be connected to them in any solution with approximation guarantee better than $n/64$ (this follows from the above property 4). From above properties 1, 2, and 3 it is clear that the squared-distance between any two points in the same cluster is at most $n^6/8$. Now we continue by moving the center of the clusters to the origin and shift the whole cluster with them. Consider any two points p_1, p_2 in the clusters and let j_1, j_2 be the centers of those clusters respectively. We know that $d_1(p_1, p_2) \leq d_1(p_1, j_1) + d_1(j_1, j_2) + d_1(j_2, p_2)$. Also we know that j_1 and j_2 are both on the origin so $d_1(p_1, p_2) \leq d_1(p_1, j_1) + d_1(j_2, p_2)$ and $d_1(p_1, p_2)^2 \leq 2(d_1(p_1, j_1)^2 + d_1(j_2, p_2)^2)$. Therefore, $d_1(p_1, p_2)^2 \leq n^6/2$. We add s new dimensions, one for each cluster. For each $1 \leq i \leq s$, we assign n^2 to the i^{th} new coordinate for the points in i^{th} cluster and 0 to the rest. Let $\ell' = \ell + s$ the number of the coordinates that the points have right now. Now consider two points and the value of their coordinates, $j = (j_1, j_2, ..., j_{\ell'}), j' = (j'_1, j'_2, ..., j'_{\ell'})$, we know that

$$d(j, j')^2 = \sum_{k=1}^{\ell'} (j_k - j'_k)^2 = \sum_{k=1}^{\ell} (j_k - j'_k)^2 + \sum_{k=\ell+1}^{\ell'} (j_k - j'_k)^2 \leq n^6/2 + 2n^4.$$

Also, it still holds that any solution with an approximation guarantee better than $n/64$ can only connect the clients in a cluster to the facilities in the same cluster, since the squared distance between any facility and any client in different clusters is at least $n^4/64$, which is more than $n/64 \cdot \mathrm{OPT}(\mathcal{I}_1)$. Now we need to make sure that the distance between the facilities and the clients is at least one. To that end, we add one new dimension and assign one for facilities and zero for clients in this coordinate. Similarly to the analysis of the general metric, we can show that any ρ-approximate solution for the new instance is also a $\rho(1 + 100/n^2)$-approximate for $\mathcal{I}$, since we increase the cost of any solution by at most n. Note that the last step does not increase the distance-squared between any two points by more than one so the maximum distance-squared between any two points is at most $n^6/2 + 2n^4 + 1 \leq n^6$.

Clearly, the running time of this procedure is poly(n). □

4 Fast Algorithms and Empirical Results for k-Means

This chapter is based on a joint work with Justin Ward and Ola Svensson.

4.1 Introduction

In this chapter, we focus on the empirical aspects of the clustering problems. We consider the *k-means*, perhaps the most studied clustering problem, from the practical point of view. We begin with a short introduction, followed by a short description of the algorithm we use. Afterwards, we describe the techniques that we use to accelerate our algorithm and present our empirical results. In order to be compatible with the naming convention widely used in the literature, we refer to facilities as centers.

The most widely-used practical algorithms for k-means clustering are based on variants of Lloyd's algorithm [48], sometimes simply called *the k-means algorithm*. Lloyd's algorithm iteratively improves a given clustering by alternating two steps. Given some current set of k centers, the algorithm first partitions the data points into k clusters by assigning each point to its nearest center. Next, a new set of centers is obtained by calculating the center of mass of each of these clusters. In order to initialize Lloyd's algorithm, one must choose a set of k starting centers. Because Lloyd's algorithm is only guaranteed to produce a *local* optimum, it might produce a k-means clustering of cost much higher than the best possible for a given instance when initialized badly.

One way to cope with this difficulty is to randomly select the initial centers for the algorithm. One of the most popular such randomized initialization routines is D^2-sampling. Here, a subset of k data points in C are chosen as the starting centers, by iteratively selecting each successive point with a probability proportional to its current squared Euclidean distance to the closest previously selected points. The K-Means++ algorithm, introduced by Arthur and Vassilvitskii [10] uses this randomized initialization to start Lloyd's algorithm. The resulting algorithm is easily implemented, and the initial sampling can be done in $O(nkd)$ time. Arthur and Vassilvitskii showed that K-Means++

performs very well on a variety of real and synthetic datasets. Furthermore, they showed that it attains a $\Theta(\log k)$ expected approximation guarantee in the worst case. Due to its simplicity, fast running time, and good practical performance, K-Means++ has emerged as one of the dominant solutions for the k-means clustering problem.

However, several algorithms are known with worst-case $O(1)$ approximation guarantees under *no* assumptions on the dataset. In contrast to many algorithms employed in practice, these algorithms are based on local search [42] or primal-dual techniques from linear programming [39]. The practical adoption of these algorithms has been limited by two issues. First, their running times, while polynomial in n, k, and d, are prohibitively large (at least quadratic in n), which prevents their use on even moderately sized instances. Second, it is unclear whether their worst-case performance guarantees translate into improved performance on practical problem instances.

Our contributions. In this chapter, we address both of these issues by giving a fast initialization procedure for Lloyd's algorithm in the primal-dual framework. Our starting point is the Lagrangian-multiplier preserving primal-dual approach for k-median and k-means, first introduced by Jain and Vazirani [39]. When combined with our results from the previous chapter, this algorithm gives the best known worst-case approximation guarantee for k-means. These recent results involve first relaxing the k-means problem to a variant of facility location (in which any number of cluster centers can be opened) and then running a complicated and costly procedure that ensures that exactly k centers are opened. We replace this procedure by a simple binary search on the facility opening price, and use additional insights to further reduce the running time of the overall procedure. This results in an algorithm that runs in total time competitive with that of k-means++, that additionally has a conditional, constant factor guarantee. Specifically, whenever our algorithm successfully finds (via binary search) a set of exactly k centers to open, the resulting clustering is guaranteed to have cost within a constant factor of the optimal clustering. As with the initialization step of K-Means++, it is possible to restrict our algorithm to choose only data points as cluster centers, and so it can also be employed for exemplar clustering.

We prove that on a large family of well-clusterable instances, our algorithm will *always* successfully recover the *optimal* clustering, whereas K-Means++ fails with high probability. We complement this proof with a set of synthetic instances inspired by the original experimental evaluation of K-Means++ [10]. As expected, our algorithm always recovers the optimal clustering for these instances, whereas, K-Means++ (on average) fails to recover 25% to 30% of the centers and produces solutions of cost about 1.7 times that of the optimum.

We also provide a thorough experimental evaluation on several previously studied datasets. We show that our algorithm produces initial clusterings of *significantly* better quality—

often by 30-40 percent—than those produced by D^2-sampling used by K-Means++. Moreover, we show that running Lloyd's algorithm by using our initial centers results in a much better set of final clusters for the k-means problem. Specifically, we show that when initialized with our procedure, Lloyd's algorithm often converges *more rapidly* (up to 50%) to *better* (up to 25%) local optima than K-Means++. Our algorithm's total running time, when combined with Lloyd's algorithm, was comparable to K-Means++ and, indeed, slightly faster over 100 executions on a large instance, owing to its superior initial solutions. We found that our binary search technique outputs exactly k centers, thus delivering a guaranteed constant-factor approximation, at least 90% of the time on every dataset (for each fixed value of k)[1] and, in fact, 100% of the time on all but one dataset we considered.

Related work. A survey by Celebi et al. gives a comprehensive description and experimental evaluation of several initialization procedures, including that employed by K-Means++ [22]. Because K-Means++ offers a good tradeoff between running time and both the solution quality and convergence time of Lloyd's algorithm [10], it has emerged as a de facto standard for k-means clustering. Recent work has focused on improved and scalable implementations of K-Means++ in streaming [1] and parallel settings [17], as well as approximations to its initial seeding using MCMC techniques [16, 15].

Parallel to this, there has been a line of theoretical work focused on obtaining improved approximation guarantees for k-means clustering in polynomial time, even when k and d are considered part of the input. These approaches have been based on two primary algorithmic paradigms: primal-dual [39, 55] and local search [42, 32, 25].

In the standard formulation of the k-means problem, the set S of k centers can be chosen as arbitrary points in $\mathbb{R}^d$. However, primal-dual and local search algorithms must instead operate on a *discrete* k-means problem, in which one can only select centers from a given, discrete set F of *potential centers* in $\mathbb{R}^d$. In theory, there are various constructions that, given a standard k-means instance, yield a set F so that the optimal cost of the resulting discrete problem is at most $(1+\epsilon)$ times that of the original, continuous instance [49, 35, 34, 43, 28]. However, many of these constructions are quite involved, and result in sets F with prohibitively large dependence on ϵ (at least ϵ^{-d}). If instead one does not insist on obtaining the best possible approximation guarantee, it is easy to see that simply using the point set C as the set of discrete centers increases the cost by at most a factor of 2.

[1] In the rare event that this fails to produce a set of exactly k centers, we simply prune the solution down to size k *arbitrarily* for the sake of comparison.

4.2 Preliminaries

As discussed before, the k-means clustering problem is formally defined as follows. The input consists of an integer k (specifying the target number of clusters) and a set C of n points in Euclidean space $\mathbb{R}^d$ of dimension $d \in \mathbb{N}$. The goal is to select k cluster centers $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$ so as to minimize $\sum_{j \in C} \min_{1 \le i \le k} d(j, \mu_i)^2$, where $d(j, \mu_i)$ denotes the Euclidean distance between j and μ_i. As discussed at the end of the introduction, although there are infinitely many choices of cluster centers, we can instead consider the discrete problem in which centers might be chosen from a polynomially large set of candidate points F, and suffer only a small loss in the clustering cost.

We begin by restating the LP relaxation and the Lagrangian relaxation for the k-means problem. Later, we state a simpler version of our algorithm used for this problem and introduce our ideas that makes it faster.

Linear programming relaxation, the Lagrangian relaxation and its dual. Having discretized the set of potential centers, the standard linear programming (LP) relaxation of k-means can now be formulated as follows:

$$\begin{aligned}
\min \quad & \textstyle\sum_{i \in F, j \in C} x_{ij} \cdot d(i,j)^2 && && (4.2.1)\\
\text{s.t.} \quad & \textstyle\sum_{i \in F} x_{ij} \ge 1 && \forall j \in C && (4.2.2)\\
& x_{ij} \le y_i && \forall i \in F, j \in C && (4.2.3)\\
& \textstyle\sum_{i \in F} y_i \le k && && (4.2.4)\\
& x, y \ge 0\,. && && (4.2.5)
\end{aligned}$$

The variables are $(y_i)_{i \in F}$ and $(x_{ij})_{i \in F, j \in C}$ with the intended meaning that y_i should take value 1 if i is selected as a center and x_{ij} should take value 1 if i is the closest selected center to point j. The constraints say that (4.2.2) each client is assigned to at least one center, (4.2.3) clients are only assigned to selected centers, and (4.2.4) no more than k centers are selected.

The primal-dual approach for clustering problems, introduced by Jain and Vazirani [39], is based on reducing the problem to the facility location problem via the Lagrangian relaxation. The Lagrangian relaxation is obtained by multiplying the constraint (4.2.4) times a Lagrange multiplier λ and moving it to the objective. This results, for every $\lambda \ge 0$, in the relaxation and its dual, as shown in Figure 4.1, which we denote by $\mathsf{LP}(\lambda)$ and $\mathsf{DUAL}(\lambda)$, respectively. The dual is slightly simplified by noticing that the dual variables $(\beta_{ij})_{i \in F, j \in C}$ corresponding to the constraints (4.2.3) of the primal can always be set $\beta_{ij} = [\alpha_j - d(j,i)]^+$; the notation $[a]^+$ denotes $\max(a, 0)$.

$\mathsf{LP}(\lambda)$

$$\min \quad \sum_{i\in\mathcal{F}, j\in\mathcal{D}} x_{ij}\cdot d(j,i)^2 + \lambda\cdot\left(\sum_{i\in\mathcal{F}} y_i - k\right)$$

s.t. (4.2.2), (4.2.3), and (4.2.5).

$\mathsf{DUAL}(\lambda)$

$$\max \quad \sum_{j\in\mathcal{D}} \alpha_j - \lambda\cdot k$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{D}} [\alpha_j - d(j,i)^2]^+ \le \lambda \qquad \forall i\in\mathcal{F} \qquad (4.2.6)$$

$$\alpha \ge 0.$$

Figure 4.1 – The Lagrangian relaxation $\mathsf{LP}(\lambda)$ and its dual $\mathsf{DUAL}(\lambda)$.

Facility location and Lagrangian multiplier preserving approximations. An important observation is that, for a fixed λ, the relaxation $\mathsf{LP}(\lambda)$ where we disregard the constant term $\lambda\cdot k$ is exactly the standard relaxation of the facility location problem with squared distances and uniform opening cost λ. Recall that the facility location problem (with uniform opening cost λ) is the problem of selecting a subset $S\subseteq F$ of centers so as to minimize the opening cost $\lambda\cdot|S|$ plus the connection cost $\sum_{j\in C}\min_{i\in S} d(i,j)^2$. The approach of Jain and Vazirani is now to employ an approximation algorithm for facility location and to find a price (Lagrangian multiplier) λ so that k centers are selected. For this to work, the used approximation algorithm for the facility location program has to satisfy a technical condition called *Lagrangian Multiplier Preserving* (LMP). Formally, a ρ-approximation algorithm is LMP for the facility location problem with opening costs λ if it returns a solution $S\subseteq F$ satisfying $\sum_{j\in C}\min_{i\in S} d(j,i)^2 \le \rho(\mathrm{OPT}(\lambda) - |S|\lambda)$, where $\mathrm{OPT}(\lambda)$ denotes the value of an optimal solution to $\mathsf{LP}(\lambda)$ without the constant term $\lambda\cdot k$. The definition of LMP is motivated by the following: suppose that an LMP ρ-approximation algorithm returns a solution S of cardinality k. We claim that S is then a ρ-approximate solution to k-means. To see this note that we have

$$\sum_{j\in C}\min_{i\in S} d(j,i)^2 \le \rho(\mathrm{OPT}(\lambda) - |S|\lambda) = \rho(\mathrm{OPT}(\lambda) - k\lambda),$$

where the right-hand side equals ρ the optimum value of $\mathsf{LP}(\lambda)$ that is a lower bound on the optimal (discrete) k-means solution. This completes our description of the framework. For more information, we refer the reader to the excellent text books [**?**] and [**?**].

4.3 A Fast Primal Dual Algorithm

In this section, we first present the LMP primal-dual algorithm that we use for the facility location problem. This algorithm is a simpler version of the algorithm that we presented in Chapter 3. Then, we explain how we use it to design a fast and provable good initialization algorithm for k-means.

4.3.1 Primal-Dual Algorithm

The primal-dual algorithm takes as input a set of points C, a set of potential centers F, and an opening cost λ, and outputs a set of centers $S \subseteq F$ as its solution. The algorithm contains two main phases, the dual-growth phase and the opening phase. The first phase increases the α-values of the points in C, while keeping the dual constraint (4.2.6) satisfied. Therefore, it finds a feasible solution to DUAL(λ). The second phase finds the centers S using the dual solution that we found in the first phase. These two phases are explained in detail below:

Dual-growth phase: We set all α-values to zero and initialize a set of *active* points A to C. Then we increase the α-values corresponding to the points in A at the same speed. We remove points from A, whenever one of the following events occurs, and terminate once $A = \emptyset$:

- **Event 1:** The constraint (4.2.6) becomes tight for some center $i \in F$ (i.e. $\sum_j [\alpha_j - d(i,j)^2]^+ = \lambda$). In this case, we call this center *tight*. We also remove all the active points j for which $\alpha_j \geq d(i,j)^2$ from A.
- **Event 2:** For some active point $j \in A$, there exists a tight center $i \in F$, such that $\alpha_j = d(i,j)^2$. In this case we remove j from A.

Opening phase: Let $N(i) = \{j \mid \alpha_j > d(i,j)^2\}$ and $t_i = \max_{j \in N(i)} \alpha_j$ for each tight center i. We construct a graph G with a vertex for each tight potential center i and an edge between two centers i_1, i_2 if and only if $d(i_1, i_2)^2 \leq \delta \min(t_{i_1}, t_{i_2})$, where δ is a constant that we fix later. Then we greedily find a maximal independent set of G and return it as the set of centers S.

The dual-growth phase of the primal-dual algorithm is the same for both Jain et al. [39] and our result in previous chapters. The main difference is in the opening phase. Jain and Vazirani's algorithm adds an edge in G between two tight centers i_1, i_2 if $N(i_1) \cap N(i_2) \neq \emptyset$. One can show that this algorithm guarantees an LMP 9-approximation. We add an edge between two tight centers if $N(i_1) \cap N(i_2) = \emptyset$ and $d(i_1, i_2)^2 \leq \delta \min(t_{i_1}, t_{i_2})$. Then they

show that by choosing $\delta = 2.314$, their algorithm results in an LMP 6.357-approximation.

Here, we have observed that our analysis in previous chapter is still valid even if the condition that $N(i_1) \cap N(i_2) \neq \emptyset$ is dropped. This enables us to overcome one of the issues in implementing a fast version of the primal-dual algorithm, as it saves a factor of n in the construction of G during the opening phase, as described below in more detail. There are two other issues to be addressed when translating the LMP primal-dual algorithm for facility location into an algorithm for k-means. First, obtaining a solution of size exactly k requires computationally expensive methods in theory to avoid pathological cases. Here, we employ a more efficient approach that works provably well in theory for well-clusterable instances and also performs well in a variety of experiments. Second, the algorithms operate on a discrete set of potential centers. Transferring a continuous instance to a discrete one while minimizing the effect on the desired approximation guarantee, might result in a large set F of potential centers.

4.3.2 Runtime Improvements

In this section, we describe our ideas for translating the LMP primal-dual algorithm into a fast and provably good primal-dual algorithm (FastPD) for the k-means problem.

Dual-growth phase and opening phase. In the dual-growth phase, we maintain a min-heap of possible events, which allows us to efficiently find the next event. As there are also at most $O(n|F|)$ events, the total running time for this part of the algorithm would be $O(n|F| \cdot \log(n|F|))$. While processing events, we can also easily compute the t-values for each center that becomes tight. Using these t-values computed during the dual-growth phase, we can construct the graph G in time $O(|F|^2)$, in contrast to time $\Omega(n|F|^2)$ as required by primal-dual approaches of Jain and Vazirani. Combining the above observations, the total running time of our primal dual algorithm for any given price λ is $O(n|F| \log(n|F|) + |F|^2)$.

Opening exactly k centers. In order to use the LMP algorithm, we must find a price λ so that the opening phase results in a set of exactly k centers. In order to accomplish this, we run a binary search: if for some value λ, we find a set of centers of size less than k then we decrease λ. Similarly, if we find a set of centers of size greater than k, we increase λ. The maximum value of λ that we need to consider is $O(n\Delta)$, as a result this process takes time at most $O(\log(n\Delta))$, where Δ is the maximum value between a point and a potential centers, which without loss of generality we can assume is poly(n). In general, this procedure might not eventually find a set of exactly k centers, since the number of centers returned by the opening phase is not a smooth function in λ. However, note that whenever our procedure does correctly find a set of k centers via binary search, the LMP

factor 6.357 guarantee holds with respect to its output. Moreover, in the next section, we formally prove that it always finds the correct k for a family of well-clusterable instances. Additionally, we observed in our experiments that on practical instances, binary search indeed almost always produces a solution of size k. In the rare case that it does not, we arbitrarily add or remove some centers from our solution to obtain exactly k centers.

Overall, our binary search ensures that the number of the times that we run the primal-dual algorithm is $O(\log n)$. Therefore, we obtain a total running time $O(n|F|\log^2 n + |F|^2 \log n)$.

Constructing a Set F of potential centers. The previous guarantees all hold for the discrete k-means problem in which centers may only be chosen from a discrete set F. We now consider the problem of obtaining a small such set of centers F so that our approximation guarantees can be translated to the original, continuous k-means problem. To balance approximation performance and running time, here, we simply set $F = C$. That is, we place one potential center at each data point. It can be seen that this results in a loss of at most a factor of 2 compared to the optimal solution that may select centers anywhere in the underlying space $\mathbb{R}^d$. In order to show that, consider any cluster Q in any solution to the given k-means instance. Let q be the centroid of this cluster, i.e., $q = \frac{1}{|Q|}\sum_{x\in Q} x$ (each coordinate of q is the average of the coordinates of the points in Q). Moreover, let $cost(Q) = \sum_{x\in Q} d(x,q)^2$ which is the actual cost of the cluster Q. We know that for any point y:

$$\sum_{x\in Q} d(x,y)^2 \leq \sum_{x\in Q} d(x,q)^2 + |Q|d(y,q)^2$$

Let y be the closest point in Q to q, we get

$$\sum_{x\in Q} d(x,y)^2 \leq 2\sum_{x\in Q} d(x,q)^2 = 2\text{cost}(Q)$$

Therefore, by choosing the closest point to the centroid in each cluster, we obtain that the total cost increases at most by a factor 2. This concludes that, assigning $F = C$ also increases the cost at most by a factor of 2.

By putting the above ideas together we get our fast primal-dual algorithm for k-means that we call FastPD. In summary, FastPD takes as input an instance (C, k) of k-means, sets the potential centers F equal to the set of clients C, then runs a binary search on the opening cost λ in order to find a set S of k opened centers.

For larger instances, we proceed similarly except we first *sparsify* the instance to reduce $|C|$. There is a long series of works investigating such techniques, which transform an instance (C, k) of k-means into an instance (C', k) such that $|C'| \ll |C|$. In this work we use a sampling approach based on Feldman et al. [28]; it runs in time $O(nkd)$, to obtain

an instance with $|C'| = O(k^2/\epsilon)$. We then run our algorithm FastPD on the resulting instance (C', k). Feldman et al. [28] show that for any constant β, any β-approximate solution for (C', k) that opens a subset of clients as centers, the same solution (set of centers) is a $\beta(1+\epsilon)$-approximate solution.

Combining the loss from sparsification with the loss due to selecting only (sparsified) data points as centers, the overall approximation guarantee that we get is $(6.357 \cdot 2)(1+\epsilon)$. Moreover, the overall running time of our algorithm is $O(nkd + \text{poly}(k, \log n))$. We summarize this in the following theorem:

Theorem 4.3.1. *For any $0 < \epsilon < 1$, the* FastPD *algorithm with sparsification, runs in time $O(nkd + poly(k/\epsilon, \log n))$. Moreover, any solution of size k that it finds is a $(12.714 + \epsilon)$-approximate solution.*

4.4 Results for Clusterable Instances

In this section, we compare our algorithm to k-means++ for instances in which the k clusters are well-separated. We start with a theoretical analysis then further manifest the advantage of our algorithm with experiments. We consider a similar model of clusterability as that considered by Awasthi et al. [12] in the study of LP-relaxations. We prove the following theorem:

Theorem 4.4.1. *Consider k spheres of unit radius in $\mathbb{R}^d$ whose centers are separated by distance $10 \leq \Delta \leq \beta$ where $\beta \geq 10$ is an arbitrary constant. For large enough n, if n random points are drawn uniformly and independently from each of the k spheres, then* FastPD *is always guaranteed to find the correct clustering whereas* K-Means++ *initialization fails with high probability to correctly recover a constant fraction (depending on β) of the clusters.*

Proof. Lemma 4.4.2 (analysis of K-Means++ initialization) and Lemma 4.4.3 (analysis of FastPD) imply the theorem.

Lemma 4.4.2. *Under the same setting as in Theorem 4.4.1,* K-Means++ *initialization fails with high probability of correctly recovering a constant fraction (depending on β) of the clusters.*

Proof. We first note that, for large enough n, we have that the cost of each sphere/cluster is with high probability at least $(1-\epsilon)n$ for a small $0 < \epsilon < 1/k$. We now analyze k-means++ initialization, assuming that this holds for each of the k spheres (which, as already mentioned, holds with high probability for large n). The key to the analysis is the following: In any iteration of k-means++ initialization, the probability of sampling a center from a cluster is at most $(2+\beta)^2/k$. This is clearly true for the first iteration (as a center is chosen uniformly at random so the probability is $1/k$ to select a center from a

specific cluster using that all clusters contain the same number of points). For the general case, consider the iteration after that t centers $D = \{d_1, \ldots, d_t\}$ have been sampled. Let $S_1, \ldots, S_k$ be the spheres/clusters of the instance. The probability to sample a center from a sphere S_j is equal to

$$\frac{\sum_{x \in S_j} d(x, D)^2}{\sum_{i=1}^{k} \sum_{x \in S_i} d(x, D)^2}\,, \tag{4.4.1}$$

where $d(x, D) = \min_{i \in D} d(x, i)$. As the distance between two points is at most $1 + \beta + 1$ (using the triangle inequality and that we have unit spheres whose centers are at most distance β apart), the numerator can be bounded by $n \cdot (2 + \beta)^2$. To bound the denominator, let s_1 be the number of spheres that have exactly one center in D and let s_0 be the ones that have none. Then $s_0 \geq k - s_1 - (t - s_1)/2 = k - (t + s_1)/2$. By the initial assumption, the cost of the centers with exactly one center in D is at least $(1-\epsilon)n$. Furthermore, as the distance between points from different centers is at least 8, the cost of the centers with zero centers in D is at least $8^2 n$. It follows that the denominator is bounded from below by at least

$$s_1 \cdot (1 - \epsilon)n + 8^2 n(k - (t + s_1)/2) \tag{4.4.2}$$

which is minimized when s_1 is maximized, i.e., when $s_1 = t$. Hence

$$\begin{aligned}(4.4.2) &\geq t(1-\epsilon)n + 8^2 n(k - t) \geq n\left(8^2(k-t) + t(1-\epsilon)\right) \\ &\geq n\left(8^2 + (k-1)(1-\epsilon)\right) \geq k \cdot n\,,\end{aligned}$$

where for the penultimate inequality we used that $t \leq k - 1$ and, for the last inequality, we used that $\epsilon < 1/k$. Our bounds on the numerator and denominator, imply that (4.4.1) is at most $(2 + \beta)^2/k$.

Having proved that in each iteration the probability to sample a center from a sphere is at most $(2+\beta)^2/k$, we can conclude that the probability to fail to recover a sphere in the initialization is at least $(1 - \frac{(2+\beta)^2}{k})^k$ which is a constant $c \approx e^{-(2+\beta)^2}$ only depending on β. Hence, the expected number of clusters that we fail to recover is at least $c \cdot k$. The fact that we fail to recover a constant fraction (say $c/2$) with high probability follows from basic variance calculations. □

We remark that the following lemma shows that our algorithmic *always* recovers the correct clustering in the setting of Theorem 4.4.1 (i.e., it is not probabilistic). In addition, using the same proof technique, one can see that it is possible to give tighter bounds on the separation assuming more structure of the distances within clusters. This further explains the experimental success of our algorithm since when selecting n random points from a sphere or a Gaussian one expects most pairwise distances to be similar.

Lemma 4.4.3. *Algorithm* FastPD *is guaranteed to return the correct clustering if given an instance in which the (unique) optimal clustering $O_1, \ldots, O_k$ satisfies:*

1. *each cluster contains n points;*
2. *$d(j, j') \leq 2$ for all j, j' points that belong to the same cluster;*
3. *$d(j, j') \geq 2\Delta = 2\sqrt{1+\delta} \approx 3.6$ for all j and j' that belong to different clusters.*

We note that the above conditions are clearly satisfied by the randomly generated instances in the statement of Theorem 4.4.1 (where each O_i corresponds to one of the spheres).

Proof. The proof follows from showing the following:

1. The algorithm opens at least one center in each cluster O_i if $\lambda \leq \frac{4n}{\delta}$;
2. The algorithm opens exactly one center in each cluster O_i if $\lambda = \frac{4n}{\delta}$;
3. The algorithm opens at most one center in each cluster if $\lambda \geq \frac{4n}{\delta}$.

We first prove the case when $\lambda = \frac{n}{\delta k}$. The other cases then follow easily. Consider a cluster O_i and recall that we have a potential center located at each client. We refer to the center co-located with client j as c_j. On the one hand, using that $|O_i| = n$ it is easy to see that no center in O_i can be tight before time $4/\delta$ (it is tight at this time if and only if all clients in O_i are co-located). On the other hand, there must be a center in O_i that is tight at time $t = 4 + \frac{4}{\delta}$. Indeed suppose that at time $t = 4 + \frac{4}{\delta}$ no center has been tight. Then the total contribution to center c_j (that is co-located with client $j \in O_i$) equals

$$\sum_{j' \in O_i} [\alpha_{j'} - d(j, j')^2]^+ = \sum_{j' \in O_i} [t - d(j, j')^2]^+ \geq |O_i|(t-4) = \lambda$$

and so c_j would be tight opened. We can thus conclude that there is at least one tight center in O_i and that any tight center c_j satisfies $4/\delta \leq t_{c_j} \leq 4 + \frac{4}{\delta}$. Hence, by definition, the graph G constructed in the opening phase of our algorithm satisfies[2]:

- It has an edge between any two tight centers of O_i since $d(j, j')^2 \leq \delta(4/\delta)$ (where $4/\delta$ is the minimum opening time).

[2]Recall that graph G has an edge between two tight facilities i_1 and i_2 if $d(i_1, i_2)^2 \leq \delta \min\{t_{i_1}, t_{i_2}\}$ where t_{i_1} and t_{i_2} are the times that these facilities became tight.

- It has no edge from a center in O_i to a tight center from another cluster because if j, j' belong to different clusters then $d(j, j')^2 > 4\Delta^2 = \delta(4 + \frac{4}{\delta})$ (where $4 + \frac{4}{\delta}$ is the maximum opening time).

In other words, G consists of k disjoint cliques of tight centers (one clique for each O_i), hence our algorithm will correctly recover the clusters. The remaining two cases now follow easily from the arguments above. If $\lambda \leq \frac{4n}{\delta}$, then by the same arguments, there must be at least one tight center in each cluster at time $t \leq 4 + \frac{4}{\delta}$, and still G will not contain any edges between this center and tight facilities from different centers. Similarly, if $\lambda \geq \frac{4n}{\delta}$ then no center can be tight before time $t \geq \frac{4}{\delta}$ and so still G must contain an edge between any two centers in the same cluster. □

The same proof technique as the last lemma also gives the following:

Lemma 4.4.4. *Algorithm* FastPD *is guaranteed to return the correct clustering if given an instance in which the (unique) optimal clustering $O_1, \ldots, O_k$ satisfies the following for some $0 \leq \epsilon \leq 1/2$:*

1. *each cluster contains n points;*
2. *$d(j, j') \in [1, 1 + \epsilon]$ for all j, j' points that belong to the same cluster;*
3. *$d(j, j') > 1 + \epsilon$ for all j and j' that belong to different clusters.*

The difference compared to Lemma 4.4.3 is that we assume some regularity of the distances within a cluster and, with this assumption, we get a tight bound on the separation of the clusters. This further explains the experimental success of our algorithm, as when selecting n random points from a sphere or a Gaussian one expects most pairwise distances to be similar.

Proof. The proof is by the same arguments as given in the proof of Lemma 4.4.3. Indeed, consider the opening price λ to be $(1 + \epsilon)^2/\delta$ which by assumption on ϵ is at most 1. Then each center (recall that centers are co-located with clients) will become tight at time $(1 + \epsilon)^2/\delta$. As $d(j, j')^2 \leq \delta(1 + \epsilon)^2/\delta$ for j and j' in the same cluster, every two tight centers belonging to the same cluster will be adjacent in the graph G constructed in the opening phase of FastPD. Similarly, as $d(j, j') > \delta(1 + \epsilon)^2/\delta$ for j and j in different clusters, no tight centers from different clusters will be adjacent in G. It follows that G consists of k disjoint cliques corresponding to the clusters if the opening price is $\lambda = (1 + \epsilon)^2/\delta$. If λ is smaller, then it is clear that we still open at least one center in each cluster (no centers from different clusters are adjacent); and if λ is bigger, then at most one center is opened in each cluster (since they form cliques in G). Hence, the lemma follows.

□

□

The above theorem considers very well-clusterable instances in the sense that the clusters are well separated and it is indeed easy to design customized algorithms for this setting. The advantage here is that our algorithm, which is both fast and has the best known worst-case guarantees, performs as expected whereas k-means++ initialization fails to correctly recover the clusters. We remark that this behavior of k-means++ has been observed before and it continues to hold even after running Lloyd's algorithm (see the experimental analysis below). Bubeck et al. [18] overcome some of these difficulties by sampling more points than k and then performing a heuristic pruning step. This can be seen as having similarities to our sparsification step (where we compress the instance through sampling). However, here we use an algorithm with known guarantees to do the pruning step.

In order to further compare the performance of our algorithm and k-means++ on well-clusterable data, we run a series of experiments using synthetically generated sets of $10,000$ points. For each $k = 25, 50, 200$, we generate a d dimensional instance (where $d = 15$ for $k = 25$ and $d = 20$ for $k = 50, 200$) as follows. We first select k cluster centers on the d-dimensional $\{-1, 1\}$-hypercube uniformly at random. The parameter d is select to ensure that any two selected centers were separated by squared Euclidean distance 8. At each selected position c, we then generate a cluster of $n = 10000/k$ points randomly. We consider two different models for generating points around a cluster center c: in the *Sphere* model, points were selected uniformly at random on the surface of a d-dimensional unit sphere with center c; in the *Gaussian* model, points were selected according to a d-dimensional Gaussian with mean c and covariance matrix $\Sigma = \frac{1}{16} I_d$. We remark that our experimental setup is similar to that used in the original experimental evaluation of K-Means++ [10], except our instances exhibit a much more moderate separation between clusters.[3]

[3]For comparison, note that [10] place clusters at the vertices of a 5 and 15 dimensional hypercube with side-length 500 and generate points from Gaussian distributions of unit variance.

Table 4.1 – Results for synthetic, well-separated instances.

		Sphere			Gaussian		
		$k = 25$	50	200	$k = 25$	50	200
Unrecovered Clusters	K-Means++	30%	26%	28%	29%	27%	29%
	FastPD	0%	0%	0%	0%	0%	0%
Cost	K-Means++	16,993	18,564	17,580	16,163	21,851	20,372
	FastPD	9,975	9,949	9,796	9,382	12,407	12,206
	Rel. Improvement	**41.3%**	**46.4%**	**44.3%**	**42.0%**	**43.2 %**	**40.1 %**

The results of our experiments are shown in Table 4.1. In order to measure the performance, we consider the number of clusters correctly recovered by each algorithm in the following sense. Let C be a randomly generated cluster and $\tilde{C}$ be some cluster returned by the algorithm. We say that $\tilde{C}$ captures C if (1) $\tilde{C}$ contains at least 95% of the points from C, and (2) at most 5% of the points in $\tilde{C}$ come from clusters other than C. We say that an algorithm *recovers* C if it returns any cluster $\tilde{C}$ that captures C. Otherwise, C is unrecovered. In Table 4.1 we report both the cost and the fraction of unrecovered clusters for the considered number of clusters correctly recovered by FastPD and by K-Means++. In all cases FastPD correctly recovers all clusters, whereas K-Means++ (even after running Lloyd's algorithm) fails to recover (on average, over 100 runs) 25-30% of the clusters. In fact K-Means++ failed to correctly recover the clusters in all of the 100 runs for the considered instances. In addition, the clusterings returned by K-Means++ have around 1.7 times the cost of that produced by FastPD.

4.5 Empirical Results

We also tested the performance of our algorithm against that of K-Means++ on a variety of datasets previously used to evaluate k-means clustering algorithms [10, 17, 15]. We considered the Breast Cancer ($n = 569$; $d = 30$), Cloud ($n = 1024$; $d = 10$), Abalone ($n = 4177$; $d = 8$), Spam ($n = 4601$, $d = 58$), and Census datasets ($n = 2,458,285$; $d = 68$) from the UCI Machine Learning Repository[4].

4.5.1 Evaluation of Initial Solutions

In this section, we evaluate the cost of the initial solutions produced by K-Means++ initialization and FastPD on the Cloud dataset and the Breast Cancer dataset. Both algorithms produce k initial centers chosen from the set of all input data points. Thus, these experiments can also be viewed as an instance of exemplar clustering, where we seek the k best representatives from each dataset. For a benchmark, we also compare both algorithms to the cost of the optimum solution for the associated linear program (as given in Section 4.2), which gives a lower-bound on the true optimal cost for each instance. Our results are presented in Table 4.2. In all cases, FastPD outperforms K-Means++ initialization. Moreover, the difference between the cost of FastPD and the LP optimum is smaller than that of the K-Means++ initialization by a factor of 4 in almost all cases.

[4]https://archive.ics.uci.edu/ml/datasets.html

Table 4.2 – Initial solution costs for Cloud and Breast Cancer datasets (costs are $\times 10^6$)

	Cloud Dataset			Breast Cancer Dataset		
	$k=10$	25	50	$k=10$	25	50
FastPD	6.83	2.49	1.37	9.72	3.18	1.34
K-Means++	11.34	3.76	1.74	16.00	4.95	1.97
LP Optimum	5.96	2.14	1.18	8.67	2.87	1.13
FastPD Ratio	**1.14**	**1.16**	**1.16**	**1.11**	**1.10**	**1.18**
K-Means++ Ratio	**1.90**	**1.56**	**1.47**	**1.84**	**1.72**	**1.74**

4.5.2 Comparison after Running Lloyd's Algorithm

Next, we evaluated the performance of running K-Means++ and FastPD followed by Lloyd's algorithm. For the Census dataset, we first employed sparsification before running FastPD as described in Section 4.3.2. After finding a set of k initial centers, we used the full, original dataset to compute the subsequent iterates for Lloyd's algorithm. Note that we always compare to the performance of K-Means++ *without* sparsification. That is, we ran K-Means++ on the full, original dataset, and all solution costs for both algorithms are reported with respect to the original, unsparsified instance. For both initialization methods, we ran Lloyd's algorithm to convergence and measured the number of iterations. In next section, we also provide a series of plots which show that our algorithm also produces solutions of better final costs, even if Lloyd's is terminated after a few iterations (as is common in practice).

Our results are shown in Table 4.3. On all datasets, we report the average of 100 runs of K-Means++. For the Census dataset (the large dataset), we report the average of 100 runs of FastPD (to account for the randomness in our sparsification routine). We present raw cost together with the relative improvement that we obtain compared to K-Means++. That is, we report the ratio $1 - \frac{\text{FastPD cost}}{\text{K-Means++ cost}}$. The results show that our algorithm performs much better than K-Means++ in all the discussed aspects. The cost of our algorithm's initial solution is roughly half the cost of K-Means++. Moreover, even after running Lloyd's algorithm completely to convergence the final clusterings that FastPD produces are on the order of 10% better than K-Means++, which performs roughly twice as many of Lloyd's iterations.

For all small instances, our algorithm always found a set of exactly k centers S. For the Census dataset, our algorithm failed to find exactly k in only 10% of the runs for each of the experiments. In that case, we simply discarded $|S| - k$ centers arbitrarily (when $|S| > k$) or added $k - |S|$ arbitrary data points as centers (when $k < |S|$) to obtain a set of k centers for Lloyd's algorithm. On small instances the running time of our algorithm and K-Means++ were comparable (both took only a few seconds). On the large instance (Census dataset), the total running time of FastPD with sparsification

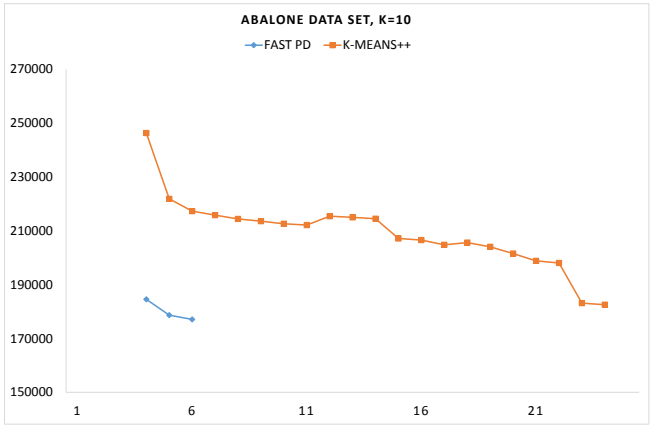
for all experiments was *lower* than that of K-Means++ by a relative factor of 8.1%, 8.9%, and 6.2% ($k = 25, 50, 75$).

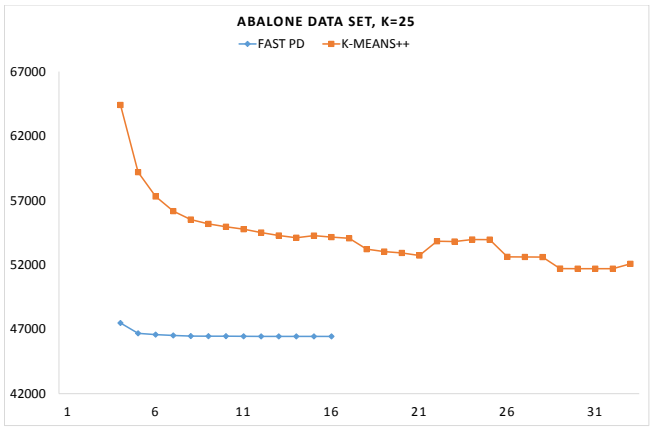
Table 4.3 – Raw experimental results (for simplicity, we divide costs by a suitable power of 10).

Dataset		Initial solution			Final solution			Number of Lloyd's rounds		
		PD	KM++	**Ratio**	PD	KM++	**Ratio**	PD	KM++	**Ratio**
Cloud	k = 10	6.86	11.34	**39.49**%	5.779	6.218	**6.63**%	9.10	26.9	**66.19**%
	k = 25	2.48	3.10	**33.89**%	2.02	2.163	**6.22**%	21.00	19.95	**-5.26** %
	k = 50	1.37	1.96	**30.06**%	1.10	1.10	**6.09**%	10.00	15.36	**34.89**%
Breast Cancer	k = 10	10.6	15.9	**33.65**%	8.64	9.24	**6.52**%	6.00	16.0	**62.68**%
	k = 25	3.18	4.95	**35.72**%	2.72	2.99	**9.12**%	4.00	12.7	**68.67** %
	k = 50	1.34	1.97	**31.53**%	1.11	1.20	**7.14**%	4.66	11.7	**60.44**%
Abalone	k = 10	23.4	36.3	**35.52**%	17.7	19.7	**10.36**%	4.00	11.3	**64.63**%
	k = 25	5.19	9.55	**45.68**%	4.64	5.33	**12.95**%	13.0	20.2	**35.70** %
	k = 50	1.89	3.13	**39.50**%	1.62	1.82	**10.73**%	14.0	27.1	**48.33**%
Spam	k = 10	24.0	34.9	**31.11**%	17.7	19.3	**8.55**%	3.00	10.0	**70.26**%
	k = 25	5.19	9.63	**39.4**%	4.64	54.1	**14.17**%	13.0	19.5	**33.63** %
	k = 50	1.89	3.13	**39.5**%	1.62	1.82	**10.73**%	14.0	27.1	**48.33**%
Census	k = 25	28.2	31.2	**9.34**%	17.1	17.9	**4.13**%	49.3	57.5	**14.21**%
	k = 50	20.7	23.1	**10.45**%	12.6	13.1	**3.93**%	60.5	72.7	**16.77** %
	k = 75	17.7	19.4	**8.83**%	10.8	1.11	**2.05**%	76.9	89.7	**14.34**%

4.5.3 Performance of FastPD Compared to K-Means++ as a Function of the Number of Steps of Lloyd's Algorithm

In Figure 4.2 we compare FastPD with K-Means++ after each step of the Lloyd's algorithm. Notice that regardless of the number of steps that we take, our algorithm always performs better than K-Means++. In fact, the difference is minimized when running Lloyd's algorithm until convergence (which is the numbers reported in the other experiments). Note that the huge improvement in the cost after one step of Lloyd's algorithm is because of the calculation of the centroids of the initial clusters (recall that both FastPD and K-Means++ uses the clients as initial cluster centers). Finally note that we plot the average cost of all the solutions for which Lloyd's has not yet converged after some number of iterations, hence the curves are not monotone decreasing.





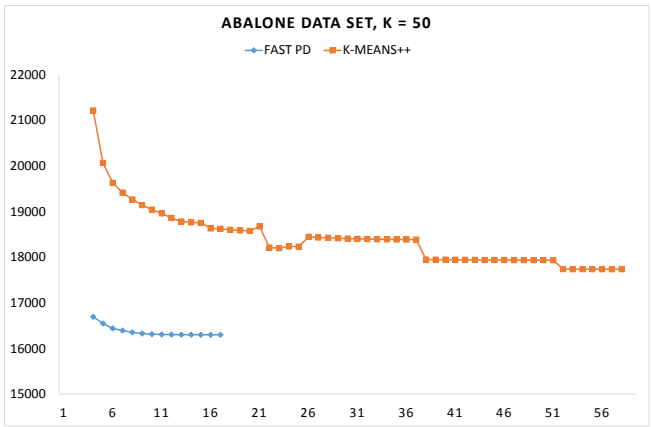


Figure 4.2 – Graphs showing the cost of the solutions obtained by FastPD and K-Means++ after each step of Lloyd's algorithm.

5 Conclusion

In this thesis, we have studied clustering problems from both a theoretical and a practical point of view. We have first considered the dynamic facility location problem. We have presented a new *stable* primal rounding technique by using exponential clock random variables and have achieved a constant-factor approximation guarantee. An interesting direction would be to use these techniques in other evolving problems.

We have also considered the k-means and the k-median problems and have designed algorithms that improve the approximation guarantees for these problems in various settings. We have presented a fairly clean quasi-polynomial time algorithm and we have showed that these techniques can be generalized to achieve a polynomial running time in a rather complex way. An interesting open problem would be to obtain a simpler algorithm with polynomial running time.

Additionally, we note that although Jain, Mahdian and Saberi [38] present an LMP 2-approximation algorithm for the facility location problem, we are not yet able to use their algorithm for our approach (and thereby not able to improve the approximation guarantee for k-median in general metrics). This is because it is unclear how to obtain a sequence of *close* dual solutions by using these techniques, hence we do not know how to guarantee that the algorithm will open exactly k facilities by only losing a factor $(1 + \epsilon)$ in that case. An interesting open question is whether it is possible to combine their techniques with ours to give a 2-approximation for k-median.

The natural LP relaxation of the k-median problem has been proven to be useful in designing approximation algorithms. Therefore understanding it better and finding the integrality gap of this LP is of great importance.

Furthermore, we have presented a fast primal-dual algorithm (FastPD) for the k-means problem; that has theoretical guarantees, as well as an improved experimental performance. We have formally showed that FastPD correctly recovers the clusters in well-clusterable instances (in contrast to K-Means++ that fails for these instances).

Our empirical evaluation have further shown that we outperform K-Means++ in three other aspects: initial solution, final solution, and the number of Lloyd's rounds. Our work is the first to use the primal-dual approach in practice for the k-means problem. An interesting open problem would be to find a systematic ways to improve the running time of primal-dual algorithms and to maintain good theoretical guarantees. It is quite easy to see that this approach can be heavily parallelized and further research in this direction looks fruitful.

Bibliography

[1] M. R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler. StreamKM++: A clustering algorithm for data streams. *J. Exp. Algorithmics*, 17:2.4:2.1–2.4:2.30, May 2012.

[2] S. Ahmadian, A. Norouzi-Fard, O. Svensson, and J. Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 61–72, 2017.

[3] H. An, A. Norouzi-Fard, and O. Svensson. Dynamic facility location via exponential clocks. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 708–721, 2015.

[4] H. An, A. Norouzi-Fard, and O. Svensson. Dynamic facility location via exponential clocks. *ACM Trans. Algorithms*, 13(2):21:1–21:20, 2017.

[5] A. Anagnostopoulos, R. Bent, E. Upfal, and P. V. Hentenryck. A simple and deterministic competitive algorithm for online facility location. *Information and Computation*, 194(2):175 – 202, 2004.

[6] B. M. Anthony and A. Gupta. Infrastructure leasing problems. In *Proc. 12th IPCO*, pages 424–438, 2007.

[7] A. Archer, R. Rajagopalan, and D. B. Shmoys. Lagrangian relaxation for the k-median problem: New insights and continuity properties. In *Proc. 11th ESA*, pages 31–42, 2003.

[8] D. Arthur and S. Vassilvitskii. How slow is the k-means method? In *Proc. 22nd SoCG*, pages 144–153, 2006.

[9] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proc. 18th SODA*, pages 1027–1035, 2007.

[10] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proc. 18th SODA*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[11] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004.

[12] P. Awasthi, A. S. Bandeira, M. Charikar, R. Krishnaswamy, S. Villar, and R. Ward. Relax, no need to round: Integrality of clustering formulations. In *Proc. 2015*, pages 191–200, New York, NY, USA, 2015. ACM.

[13] P. Awasthi, A. Blum, and O. Sheffet. Stability yields a PTAS for k-median and k-means clustering. In *Proc. 51st FOCS*, pages 309–318, 2010.

[14] P. Awasthi, M. Charikar, R. Krishnaswamy, and A. K. Sinop. The hardness of approximation of euclidean k-means. In *Proc. 31st SoCG*, pages 754–767, 2015.

[15] O. Bachem, M. Lucic, H. Hassani, and A. Krause. Fast and provably good seedings for k-means. In *NIPS*, pages 55–63, 2016.

[16] O. Bachem, M. Lucic, S. H. Hassani, and A. Krause. Approximate k-means++ in sublinear time. In *Proc. 30th*, pages 1459–1467, 2016.

[17] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. Scalable k-means++. *PVLDB*, 5(7):622–633, 2012.

[18] S. Bubeck, M. Meila, and U. von Luxburg. How the initialization affects the stability of the k-means algorithm. *ESAIM: Probability and Statistics*, 16:436–452, Jan. 2012.

[19] N. Buchbinder, J. S. Naor, and R. Schwartz. Simplex partitioning via exponential clocks and the multiway cut problem. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, STOC '13, pages 535–544, 2013.

[20] J. Byrka and K. Aardal. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. *SIAM J. Comput.*, 39(6):2212–2231, 2010.

[21] J. Byrka, T. Pensyl, B. Rybicki, A. Srinivasan, and K. Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proc. 26th SODA*, pages 737–756, 2015.

[22] M. E. Celebi, H. A. Kingravi, and P. A. Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1):200 – 210, 2013.

[23] M. Charikar and S. Guha. Improved combinatorial algorithms for facility location problems. *SIAM J. Comput.*, 34(4):803–824, 2005.

[24] F. A. Chudak and D. B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM J. Comput.*, 33(1):1–25, 2004.

[25] V. Cohen-Addad, P. N. Klein, and C. Mathieu. The power of local search for clustering. *CoRR*, abs/1603.09535, 2016.

[26] G. Divéki and C. Imreh. Online facility location with facility movements. *Central European Journal of Operations Research*, 19(2):191–200, 2011.

[27] D. Eisenstat, C. Mathieu, and N. Schabanel. Facility location in evolving metrics. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming*, ICALP '14, 2014. To appear.

[28] D. Feldman, M. Monemizadeh, and C. Sohler. A PTAS for k-means clustering based on weak coresets. In J. Erickson, editor, *Proc. 23rd SoCG*, pages 11–18, 2007.

[29] D. Fotakis. Incremental algorithms for facility location and k-median. *Theoretical Computer Science*, 361(2-3):275 – 313, 2006.

[30] D. Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008.

[31] D. Fotakis. Online and incremental algorithms for facility location. *SIGACT News*, 42(1):97–131, 2011.

[32] Z. Friggstad, M. Rezapour, and M. R. Salavatipour. Local search yields a PTAS for k-means in doubling metrics. *CoRR*, abs/1603.08976, 2016.

[33] A. Gupta and K. Tangwongsan. Simpler analyses of local search algorithms for facility location. *CoRR*, abs/0809.2554, 2008.

[34] S. Har-Peled and A. Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007.

[35] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *Proc. 36th STOC*, pages 291–300. ACM, 2004.

[36] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recogn. Lett.*, 31(8):651–666, June 2010.

[37] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *J. ACM*, 50:795–824, 2003.

[38] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *Proc. 34th STOC*, pages 731–740, 2002.

[39] K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001.

[40] K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001.

[41] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. A local search approximation algorithm for k-means clustering. *Comput. Geom.*, 28(2-3):89–112, 2004.

[42] T. Kanungo, D. M. Mount, N. S. Netanyahu, A. Y. Wu, and C. D. Piatko. A local search approximation algorithm for k -means clustering. *Computational Geometry*, 28(2-3):89–112, 2004.

[43] A. Kumar, Y. Sabharwal, and S. Sen. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM*, 57(2):5:1–5:32, Feb. 2010.

[44] E. Lee, M. Schmidt, and J. Wright. Improved and simplified inapproximability for k-means. *CoRR*, abs/1509.00916, 2015.

[45] S. Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Inf. Comput.*, 222:45–58, 2013.

[46] S. Li and O. Svensson. Approximating k-median via pseudo-approximation. *SIAM J. Comput.*, 45(2):530–547, 2016.

[47] J. Lin and J. S. Vitter. Approximation algorithms for geometric median problems. *Inf. Process. Lett.*, 44:245–249, 1992.

[48] S. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theor.*, 28(2):129–137, Sept. 2006.

[49] J. Matoušek. On approximate geometric k-clustering. *Discrete & Computational Geometry*, 24(1):61–84, 2000.

[50] A. Meyerson. Online facility location. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 426–431, 2001.

[51] C. Nagarajan and D. P. Williamson. Offline and online facility leasing. In *Proc. 13th IPCO*, IPCO'08, pages 303–315, Berlin, Heidelberg, 2008. Springer-Verlag.

[52] M. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[53] R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy. The effectiveness of Lloyd-type methods for the k-means problem. *J. ACM*, 59(6):28:1–28:22, Jan. 2013.

[54] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Phys. Rev. Lett.*, 86:3200–3203, 2001.

[55] M. Shindler, A. Wong, and A. Meyerson. Fast and accurate k-means for large datasets. In *NIPS*, pages 2375–2383, 2011.

[56] D. B. Shmoys, E. Tardos, and K. Aardal. Approximation algorithms for facility location problems (extended abstract). In *Proc. 29th STOC*, pages 265–274, 1997.

[57] J. Stehlé, N. Voirin, A. Barrat, C. Cattuto, L. Isella, J.-F. Pinton, M. Quaggiotto, W. Van den Broeck, C. Régis, B. Lina, and P. Vanhems. High-resolution measurements of face-to-face contact patterns in a primary school. *PLoS ONE*, 6(8):e23176, 2011.

[58] C. Tantipathananandh, T. Berger-Wolf, and D. Kempe. A framework for community identification in dynamic social networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 717–726, 2007.

[59] A. Vattani. k-means requires exponentially many iterations even in the plane. *Discrete & Computational Geometry*, 45(4):596–616, 2011.

[60] D. Williamson and D. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

Ashkan Norouzi-Fard

EPFL IC IIF THL2
INF 136, Station 14
CH-1015 Lausanne

Cell Phone: +41-78-7911017
Email: ashkan.norouzifard@epfl.ch
Home Page: people.epfl.ch/ashkan.norouzifard

RESEARCH INTERESTS

- Combinatorial Optimization
- Approximation and Randomized Algorithms
- Online and Streaming Algorithms
- Machine Learning

EDUCATION

- **École Polytechnique Fédérale de Lausanne**, Switzerland (Sept 2013 - present)
  - Ph.D candidate in Computer Science
  - **Advisor:** Prof. Ola Svensson
- **Sharif University of Technology**, Tehran, Iran (Sept 2009 - July 2013)
  - Thesis: Finding Paths with Minimum Shared Edges
  - Thesis supervisor: Prof. Hamid Zarrabi-Zadeh

PUBLICATIONS

- **Better Guarantees for k-Means and Euclidean k-Median by Primal-Dual Algorithms**
  Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. [1]
  58th Annual IEEE Symposium on Foundations of Computer Science, **FOCS 2017**.
  Invited to a special issue of SICOMP.

- **Streaming Robust Submodular Maximization: A Partitioned Thresholding Approach**
  Ilija Bogunovic, Volkan Cevher, Slobodan Mitrovic, Ashkan Norouzi-Fard and Jakub Tarnawski. [1]
  30th Conference on Neural Information Processing Systems, **NIPS 2017**.

- **An Efficient Streaming Algorithm for the Submodular Cover Problem**
  Ashkan Norouzi-Fard, Abbas Bazzi, Marwa El Halabi, Ilija Bogunovic, Ya-Ping Hsieh, and Volkan Cevher.
  30th Conference on Neural Information Processing Systems, **NIPS 2016**.

- **Dynamic Facility Location via Exponential Clocks**
  Hyung-Chan An, Ashkan Norouzi-Fard, and Ola Svensson.[1]
  26th Annual ACM-SIAM Symposium on Discrete Algorithms, **SODA 2015**.
  Invited to a special issue of ACM Transactions on Algorithms.

- **Towards Tight Lower Bounds for Scheduling Problems**
  Abbas Bazzi and Ashkan Norouzi-Fard.[1]
  23rd Annual European Symposium on Algorithms, **ESA 2015**.

- **A Novel Probabilistic Key Management Algorithm for Large-Scale MANETs**
  Mohammad Gharib, Ehsan Emamjomeh-Zadeh, Ashkan Norouzi-Fard, and Ali Movaghar.
  27th IEEE International Conference on Advanced Information Networking and Applications, **AINA 2013**.

[1] The authors are sorted in alphabetical order.

◇ **The Minimum Vulnerability Problem**
Sepehr Assadi, Ehsan Emamjomeh-Zadeh, Ashkan Norouzi-Fard, Sadra Yazdanbod, and Hamid Zarrabi-Zadeh.[1]
23rd International Symposium on Algorithms and Computation, **ISAAC 2012**.
Full version in **Algorithmica** special issue for ISAAC 2012 papers.

◇ **Some Upper Bounds for Signed Star Domination Number Of Graphs**
Saieed Akbari, Ashkan Norouzi-Fard, Alireza Rezaei, Rahmtin Rotabi, and Sara Sabour. [1]
Discrete Applied Mathematics.

MANUSCRIPTS

◇ **Beyond $1/2$-Approximation for Submodular Maximization on Massive Data Streams**
Ashkan Norouzi-Fard, Jakub Tarnawski, Slobodan Mitrovic, Amir Zandieh, Aida Mousavifar, and Ola Svensson.

◇ **From Theory to Practice: Better Initializations for k-Means via the Primal-Dual Method**
Ashkan Norouzi-Fard, Justin Ward and Ola Svensson.

◇ **Data-Driven Rebalancing Methods for Bike-Share Systems**
Daniel Freund, Ashkan Norouzi-Fard, Alice Paul, Shane G. Henderson, and David B. Shmoys.

RESEARCH AND ACADEMIC ACTIVITIES

◇ **Research Visit** to the Center of Applied Math (CAM), Cornell University (Summer 2016).
  · **Supervisor:** Prof. David Shmoys

◇ **Chair** of the Second Winter School at Sharif University of Technology (Winter 2016).

◇ My paper entitled "Dynamic Facility Location via Exponential Clocks" was **invited to a special issue of ACM Transactions on Algorithms**

◇ My paper entitled "Better Guarantees for k-Means and Euclidean k-Median by Primal-Dual Algorithms" was **invited to a special issue of SICOMP**

HONORS AND AWARDS

◇ **Silver Medal** in 2014 Southwestern European Regional Programming Contest, 2014.

◇ **Silver Medal** in 2013 Southwestern European Regional Programming Contest, 2013.

◇ **Silver Medal** in 17th Iranian National Olympiad in Informatics (INOI), 2008.

◇ **Ranked 1st at Sharif Freshmen ACM Challenge**, 2009.

◇ **Awarded summer school grant** for undergraduate studies from Information Engineering Department at Chinese University of HongKong **(CUHK)**, 2012.

TEACHING EXPERIENCE

◇ **Teaching Assistant**, EPFL
  · **Advanced Algorithms** (Spring 2017)
  · **Advanced Topics in Theoretical Computer Science** (Spring 2016)
  · **Algorithms** (Fall 2014, Fall 2015, Fall 2016)
  · **Theory of Computation** (Spring 2014, Spring 2015)

◇ **Teaching Assistant**, Sharif University of Thechnology
  · **Theory of Machine Languages and Automatas** (Fall 2012, Spring 2012, Fall 2011)
  · **Design and Analysis of Algorithms** (Fall 2012, Fall 2011)
  · **Discrete Mathematics** (Spring 2012)
  · **Fundamentals of Programming** (Fall 2011)
  · **Data Structures and Introduction to Algorithms** (Fall 2011, Spring 2010)
  · **Introduction to Programming** (Fall 2011, Spring 2011)
  · **Advanced Programming** (Spring 2010)

SKILLS

◇ Programming: **C++**, Python, Java, Matlab.