FPT APPROXIMATIONS FOR CONSTRAINED CLUSTERING PROBLEMS

DISHANT GOYAL

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY DELHI

MARCH 2023

FPT APPROXIMATIONS FOR CONSTRAINED CLUSTERING PROBLEMS

by

DISHANT GOYAL

Department of Computer Science and Engineering

Submitted

in fulfillment of the requirements of the degree of **Doctor of Philosophy**

to the

Indian Institute of Technology Delhi

March 2023

Certificate

This is to certify that the thesis titled **FPT APPROXIMATIONS FOR CONSTRAINED CLUSTERING PROBLEMS** being submitted by **Mr. DISHANT GOYAL** for the award of **Doctor of Philosophy** in **Computer Science and Engineering** is a record of bona fide work carried out by him under my guidance and supervision at the **Department of Computer Science and Engineering, Indian Institute of Technology Delhi**. The work presented in this thesis has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma.

Ragesh Jaiswal

Associate Professor

Department of Computer Science and Engineering

Indian Institute of Technology Delhi

New Delhi- 110016

Amit Kumar

Professor

Department of Computer Science and Engineering

Indian Institute of Technology Delhi

New Delhi- 110016

Acknowledgements

I am highly indebted to my advisors Prof. Ragesh Jaiswal and Prof. Amit Kumar, for their constant support and guidance during my PhD and my life in general. I am highly thankful to them for believing in me and encouraging me. A very special thanks to Prof. Ragesh Jaiswal for tolerating my grammar mistakes. My heartfelt gratitude to Prof. Ragesh Jaiswal for keeping a constant check on my well-being all this time.

A very special thanks to Prof. Anup Bhattacharya for his invaluable encouragement and discussion sessions. I am also grateful to my research committee members Prof. Naveen Garg, Prof. Bhawani Sankar Panda, and Prof. Preeti Ranjan Panda, for their valuable and constructive suggestions during the development of this research work. I am deeply fortunate to have been taught and guided by Prof. Venkatesh Raman during my B.Tech. at IIT Jodhpur.

I extend my gratitude to Tata Consultancy Services (TCS) for their support through the TCS Research Scholar Program Fellowship. I am also thankful to the theoretical computer science stack exchange community for the invaluable discussions on the clustering problems.

A special thanks to my SIT 309 friends – Vinayak Gupta, Sandeep Kumar, Arindam Bhattacharya, Dilpreet Kaur, Omais Shafi, and Ovia Seshadri for their constant support via poker nights, cricket games, food hunting trips, and innumerable coffee breaks. My heartfelt thanks to Vinayak Gupta for his realistic life advises. I am also thankful to have immense support from Neetu Jindal during the last phase of my degree. Lastly, I thank my parents and relatives for

standing by me during the course of this degree. I thank my sister for being there with me in all my good and bad times. This thesis is dedicated to my sister – Neha! :)

Dishant Goyal

Abstract

In this work, we study a wide range of *constrained* clustering problems in offline and streaming settings. We study these problems corresponding to three clustering objectives: k-median, k-means, and k-supplier. The (unconstrained) k-median problem is defined as follows. We are given a set of clients C in a metric space $\mathcal{X}$, with distance function $d(.,.)$. We are also given a set of feasible facility locations $L \subseteq \mathcal{X}$. The goal is to open a set $F \subseteq L$ of k facilities that minimizes the objective function: cost$(F, C) \equiv \sum_{j \in C} d(F, j)$, where $d(F, j)$ is the distance of client j to the closest facility in F. The k-means problem is defined in similar manner by replacing the distances with squared distances in the cost function, i.e., cost$(F, C) \equiv \sum_{j \in C} d(F, j)^2$. On the other hand, the k-supplier objective is defined as: cost$(F, C) \equiv \max_{j \in C} \{d(F, j)\}$. Furthermore, for $L = C$, the k-supplier problem is known as the k-center problem.

In many applications, there are additional constraints imposed on the clusters. For example, to balance the load among the facilities in resource allocation problems, a capacity u is imposed on every cluster. That is, no more than u clients can be assigned to any facility/cluster. This problem is known as the *capacitated* clustering problem. Likewise, various other applications have different constraints, which give rise to different *constrained* versions of the problem. In the past, the constrained versions of clustering problems were studied separately as independent problems. Recently, Ding and Xu [72] gave a unified framework for these problems that they called the *constrained clustering* framework. They proposed this framework in the context

of the k-median and k-means objectives in the continuous Euclidean space where $L = \mathbb{R}^p$ (p-dimensional Euclidean space) and C is a finite subset of $\mathbb{R}^p$. In this work, we extend this framework to the k-supplier objective and general metric spaces. The unified framework allows us to obtain results simultaneously for the following constrained versions of the problem: r-gather, r-capacity, balanced, chromatic, fault-tolerant, strongly private, ℓ-diversity, and fair clustering problems. We also study the *outlier* versions of these problems. In the outlier version, a clustering is obtained over at least $|C| - m$ clients instead of the entire client set.

For the constrained k-supplier and k-center problems, we obtain the following results:

(1) We give 3 and 2 approximation algorithms for the constrained k-supplier and k-center problems, respectively, with FPT (fixed-parameter tractable) running time $k^{O(k)} \cdot n^{O(1)}$, where $n = |C \cup L|$. Moreover, we note that the obtained approximation guarantees are tight. That is, for any constant $\varepsilon > 0$, no algorithm can achieve $(3-\varepsilon)$ and $(2-\varepsilon)$ approximation guarantees for the constrained k-supplier and k-center problems, respectively, in FPT time parameterized by k, assuming FPT $\neq$ W[2].

(2) For the outlier versions of the constrained k-supplier and k-center problems, we give 3 and 2 approximation guarantees with FPT running time $(k + m)^{O(k)} \cdot n^{O(1)}$, where $n = |C \cup L|$ and m is the number of outliers. Moreover, we note that the obtained approximation guarantees are tight. That is, for any constant $\varepsilon > 0$, no algorithm can achieve $(3 - \varepsilon)$ and $(2 - \varepsilon)$ approximation guarantees for the constrained k-supplier and k-center problems, respectively, in FPT time parameterized by k and m, assuming FPT $\neq$ W[2].

For the constrained k-median and k-means problems, we obtain the following results:

(3) We give $(3 + \varepsilon)$ and $(9 + \varepsilon)$ approximation algorithms for the constrained k-median and k-means problems, respectively, with FPT running time $(k/\varepsilon)^{O(k)} \cdot n^{O(1)}$, where

$n = |C \cup L|$. For the outlier version of the constrained k-median and k-means problems, we give $(3+\varepsilon)$ and $(9+\varepsilon)$ approximation algorithms, respectively, with FPT running time $\left((k+m)/\varepsilon\right)^{O(k)} \cdot n^{O(1)}$, where $n = |C \cup L|$ and m is the number of outliers.

(4) We also study the problems when $C \subseteq L$, i.e., a facility can be opened at a client location as well. For this special case, we design $(2+\varepsilon)$ and $(4+\varepsilon)$-approximation algorithms for the constrained k-median and k-means problems, respectively, with FPT running time $(k/\varepsilon)^{O(k)} \cdot n^{O(1)}$, where $n = |L|$. For the outlier version, we obtain the same approximation guarantees with FPT running time $\left((k+m)/\varepsilon\right)^{O(k)} \cdot n^{O(1)}$, where $n = |L|$ and m is the number of outliers. Note that the case $C \subseteq L$ subsumes the case $C = L$. Therefore, this result also holds for the case when $C = L$.

(5) We show that the analysis of our algorithm is tight. That is, there are instances for which our algorithm does not provide better than $(3-\delta)$ and $(9-\delta)$ approximation guarantee corresponding to k-median and k-means objectives, respectively, for any arbitrarily small constant $\delta > 0$. Similarly, the analysis of our algorithm is tight for the special case $C \subseteq L$.

(6) Our algorithms are based on a simple sampling-based approach. This approach allows us to convert these algorithms to constant-pass log-space streaming algorithms.

(7) We also study the constrained k-median/means problem in continuous Euclidean space where $L = \mathbb{R}^p$ and C is a finite subset of $\mathbb{R}^p$. We design $(1+\varepsilon)$-approximation algorithm for the outlier version of these problems with FPT running time $O\left(np \cdot ((k+m)/\varepsilon)^{O(k/\varepsilon^{O(1)})}\right)$, where $n = |C|$ and m is the number of outliers. We also convert these algorithms to constant-pass log-space streaming algorithms.

We also study the *socially fair k-median/k-means problem*, which is a generalization of the k-supplier and k-median/means problems. The problem is defined as follows. We are given a set of clients C in a metric space $\mathcal{X}$ with a distance function $d(.,.)$. There are ℓ groups:

$C_1, \ldots, C_\ell \subseteq C$. We are also given a set L of feasible centers in $\mathcal{X}$. The goal in the socially fair k-median problem is to find a set $F \subseteq L$ of k centers that minimizes the maximum average cost over all the groups. That is, find F that minimizes the objective function: fair-cost$(F, C) \equiv \max_j \left\{ \sum_{x \in C_j} d(F, x)/|C_j| \right\}$, where $d(F, x)$ is the distance of x to the closest center in F. The socially fair k-means problem is defined similarly by using squared distances, i.e., $d^2(., .)$ instead of $d(., .)$. We obtain the following results for this problem:

(8) We design $(3+\varepsilon)$ and $(9+\varepsilon)$ approximation algorithms for the socially fair k-median and k-means problems, respectively, in FPT time $f(k, \varepsilon) \cdot n^{O(1)}$, where $f(k, \varepsilon) = (k/\varepsilon)^{O(k)}$ and $n = |C \cup L|$.

(9) Furthermore, these approximation guarantees are tight; that is, for any constant $\varepsilon > 0$, no algorithm can achieve $(3 - \varepsilon)$ and $(9 - \varepsilon)$ approximation guarantees for the socially fair k-median and k-means problems in FPT time parametrized by k, assuming FPT $\neq$ W[2].

Lastly, we give hardness of approximation result for the k-median problem in the continuous Euclidean space where $L = \mathbb{R}^p$ and C is a finite subset of $\mathbb{R}^p$. This solves an open problem posed explicitly in the work of Awasthi *et al.* [19]. More precisely, we obtain the following result:

(10) There exists a constant $\varepsilon > 0$ such that the Euclidean k-median problem in $O(\log k)$ dimensional space cannot be approximated to a factor better than $(1 + \varepsilon)$, assuming the Unique Games Conjecture.

Furthermore, we study the hardness of approximation for the Euclidean k-means/k-median problems in the *bi-criteria setting*. In the bi-criteria setting, algorithms are allowed to output βk centers (for some constant $\beta > 1$), and the approximation ratio is computed with respect to the optimal k-means/k-median cost. We show the following results:

(11) For any constant $1 < \beta < 1.015$, there exists a constant $\varepsilon > 0$ such that there is no $(1 + \varepsilon)$ bi-criteria approximation algorithm for the Euclidean k-median problem in $O(\log k)$ dimensional space assuming the Unique Games Conjecture.

(12) For any constant $1 < \beta < 1.28$, there exists a constant $\varepsilon > 0$ such that there is no $(1 + \varepsilon)$ bi-criteria approximation algorithm for the Euclidean k-means problem in $O(\log k)$ dimensional space assuming the Unique Games Conjecture.

सार

इस काम में, हम ऑफ़लाइन और स्ट्रीमिंग समायोजन में बाध्य क्लस्टरिंग समस्याओं की एक विस्तृत श्रृंखला का अध्ययन करते हैं। हम तीन क्लस्टरिंग उद्देश्यों के अनुरूप इन समस्याओं का अध्ययन करते हैं: k-माध्यिका, k-माध्य, और k-आपूर्तिकर्ता। (अप्रतिबंधित) k-माध्यिका समस्या को निम्नानुसार परिभाषित किया गया है। हमें मेट्रिक जगह X में ग्राहक C का एक सेट दिया गया है, जिसमें डिस्टेंस फंक्शन d(., .) है। हमें सुविधा स्थानों का एक सेट L ⊆ X भी दिया जाता है। हमें लक्ष्य के सुविधाओं का एक सेट F⊆ L खोलना है जो उद्देश्य लागत को कम करता है: लागत (F, C) = ∑j∈C d(F, j), जहां d(F, j) दूरी है ग्राहक j कि F में निकटतम सुविधा के लिए। k-माध्य समस्या को कुछ इसी तरह परिभाषित किया गया है- लागत फलन में दूरियों की जगह वर्ग दूरियों का प्रयोग किया जाता है, अर्थात् d(F,j) की जगह d(F,j)^2। दूसरी ओर, k-आपूर्तिकर्ता उद्देश्य निम्नानुसार परिभाषित परिभाषित किया गया है: लागत(F, C) = max_j∈C {d(F, j)}। इसके अलावा, L = C के लिए, k-आपूर्तिकर्ता समस्या को k-केंद्र समस्या के रूप में जाना जाता है।

कई अनुप्रयोगों में, क्लस्टर पर अतिरिक्त प्रतिबंध लगाए गए हैं। उदाहरण के लिए, संमाध्य आवंटन समस्याओं में सुविधाओं के बीच भार को संतुलित करने के लिये, एक क्षमता हर क्लस्टर पर थोपा गया है। इस समस्या को कैपेसिटेटेड क्लस्टरिंग समस्या के रूप में जाना जाता है। इसी तरह, विभिन्न अन्य अनुप्रयोग, अलग-अलग बाधाएँ लगाते हैं, जो समस्या के विभिन्न बाध्य संस्करणों को जन्म देती हैं। अतीत में, क्लस्टरिंग समस्याओं के बाध्य संस्करणों का स्वतंत्र रूप से अलग से अध्ययन किया गया था। हाल ही में, डिंग और जू [68] ने इन समस्याओं के लिए एक एकीकृत ढांचा दिया जोकि बाध्य क्लस्टरिंग ढांचा कहा जाता है। उन्होंने इस ढांचे को यूक्लिडियन अंतरिक्ष में k-माध्यिका और k- माध्य उद्देश्यों में प्रस्तावित किया जहां L = R^p (p-डिमेंशनल यूक्लिडियन स्पेस) और C एक परिमित उपसमुच्चय है R^p का। इस काम में, हम इसका विस्तार करते हैं k-आपूर्तिकर्ता उद्देश्य और सामान्य मीट्रिक स्थान के लिए। एकीकृत ढांचा समस्या के निम्नलिखित बाध्य संस्करणों के लिए एक साथ परिणाम प्राप्त करने की अनुमति देता है: r-इकट्ठा, r-क्षमता, संतुलित, रंगीन, दोष-सहिष्णु, दृढ़ता से निजी, ℓ-विविधता, और निष्पक्ष क्लस्टरिंग समस्याएं। हम इन समस्याओं के बाहरी

संस्करणों का भी अध्ययन करते हैं। बाहरी संस्करण में, एक क्लस्टरिंग में कम से कम |C| - m ग्राहकों की प्राप्त की जाती है बजाय संपूर्ण ग्राहक सेट के। बाध्य k-आपूर्तिकर्ता और k- केंद्र समस्याओं के लिए, हम निम्नलिखित परिणाम प्राप्त करते हैं:

(1) हम बाध्य k-आपूर्तिकर्ता और k- केंद्र समस्याओं के लिए 3 और 2 सन्निकटन एल्गोरिदम देते हैं तथा FPT (फिक्स्ड-पैरामीटर ट्रैक्टेबल) चलने के समय O(f(k))·n^O(1) के साथ, जहां n = |C ∪L|। इसके अलावा, ध्यान दें कि प्राप्त सन्निकटन गारंटी तंग हैं। अर्थात्, किसी भी स्थिरांक ε > 0 के लिए, कोई भी एल्गोरिथ्म (3−ε) और (2−ε) अनुकरण गारंटी प्राप्त नहीं कर सकता है- बाध्य k-आपूर्तिकर्ता और k- केंद्र समस्याओं के लिए FPT समय में, FPT ≠ W[2] मानकर।

(2) बाध्य k-आपूर्तिकर्ता और k-केंद्र समस्याओं के बाहरी संस्करणों के लिए, हम 3 और 2 सन्निकटन गारंटी देते हैं, FPT चलने के समय (k + m)^O(k) · n^O(1) के साथ, जहां n = |C ∪L| और m बाहरी ग्राहकों (आउटलेर्स) की संख्या है। इसके अलावा, ध्यान दें कि प्राप्त सन्निकटन गारंटी तंग हैं। अर्थात्, किसी भी स्थिरांक ε> 0 के लिए, कोई एल्गोरिदम (3 - ε) और (2 - ε) सन्निकटन गारंटी प्राप्त नहीं कर सकता है सीमित k-आपूर्तिकर्ता और k- केंद्र समस्याओं के लिए, FPT समय में FPT ≠ W[2] मानकर।

बाध्य k-माध्यिका और k-माध्य समस्याओं के लिए, हम निम्नलिखित परिणाम प्राप्त करते हैं:

(3) हम बाध्य k-माध्यिका और k-माध्य समस्याएं के लिए (3 + ε) और (9 +ε) सन्निकटन एल्गोरिदम देते हैं, (k/ε)^O(k)·n^O(1) FPT चलने के समय के साथ, जहां n = |C ∪L|। बाध्य k-माध्यिका और k-माध्य समस्याओं के बाहरी संस्करण के लिए, हम (3 + ε) और (9 + ε) सन्निकटन एल्गोरिदम देते हैं, ((k + m)/ε)O(k) · n^O(1) FPT चलने के समय के साथ, जहां n = |C ∪ L| और m बाहरी ग्राहकों (आउटलेर्स) की संख्या है।

(4) हम उन समस्याओं का भी अध्ययन करते हैं जब C ⊆ L, यानी, ग्राहक स्थान पर एक सुविधा भी खोली जा सकती है। इस विशेष मामले के लिए, हम (2 + ε) और (4 + ε) - सन्निकटन एल्गोरिदम देते हैं, बाध्य

k-माध्यिका और k-माध्य समस्याओं के लिए, (k/ε)^O(k) · n^O(1) FPT चलने के समय के साथ, जहां n = |L|. बाहरी संस्करण के लिए, हम वही सन्निकटन गारंटी प्राप्त करते हैं, FPT चलने के समय के साथ ((k + m)/ε)^O(k) · n^O(1), जहां n = |L| और m बाहरी ग्राहकों (आउटलेर्स) की संख्या है। ध्यान दें कि स्थिति C ⊆ L, स्थिति C = L को समाहित करती है। इसलिए, यह परिणाम उस स्थिति के लिए भी मान्य है जब C = L.

(5) हम दिखाते हैं कि हमारे एल्गोरिथ्म का विश्लेषण कड़ा है। अर्थात्, ऐसे उदाहरण हैं जिनके लिए हमारा एल्गोरिथ्म k-माध्यिका और k-माध्य उद्देश्यों के अनुरूप (3 - ε) और (9 - ε) सन्निकटन गारंटी से बेहतर प्रदान नहीं करता है किसी भी मनमाने ढंग से छोटे ε > 0 के लिए. इसी तरह, विशेष मामले के लिए हमारे एल्गोरिथ्म का विश्लेषण तंग है।

(6) हमारे एल्गोरिदम एक साधारण नमूना-आधारित दृष्टिकोण पर आधारित हैं। यह दृष्टिकोण इन एल्गोरिदम को निरंतर-पास लॉग-स्पेस स्ट्रीमिंग एल्गोरिदम में बदलने के लिए हमें अनुमति देता है।

(7) हम निरंतर यूक्लिडियन अंतरिक्ष में बाध्य k-माध्यिका/माध्य समस्या का भी अध्ययन करते हैं जहां L = R^p और C एक परिमित उपसमुच्चय है R^p का। हम इन समस्याओं के बाहरी संस्करण के लिए (1 + ε) - सन्निकटन एल्गोरिथ्म O(np· ((k + m)/ε)^O(k/ε^O(1))) FPT चलने के समय के साथ रचना करते हैं, जहां n = |C| और m बाहरी ग्राहकों (आउटलेर्स) की संख्या है। हम इन एल्गोरिदम को निरंतर-पास लॉग-स्पेस स्ट्रीमिंग एल्गोरिदम में भी परिवर्तित करते हैं।

हम सामाजिक रूप से निष्पक्ष k-माध्यिका/k-माध्य समस्या का भी अध्ययन करते हैं जिसे निम्नानुसार परिभाषित किया गया है। हमें मेट्रिक स्पेस X में ग्राहक C का एक सेट, डिस्टेंस फंक्शन d(., .) के साथ दिया गया है। समूह हैं: C1, . . . , Cℓ . हमें X में व्यवहार्य केंद्रों का एक सेट L भी दिया गया है। सामाजिक रूप से लक्ष्य निष्पक्ष k-माध्यिका समस्या के केंद्रों का एक सेट F ⊆ L ढूंढना है जो अधिकतम समूहों पर औसत लागत को कम करता है। यही है, F खोजें जो उद्देश्य लागत को कम करता है: लागत (F, C) = max{ x∈C_j d(F,x)/|C_j |} , जहां d(F, x), F में निकटतम केंद्र से x की दूरी है। सामाजिक रूप से निष्पक्ष k- माध्य समस्या को समान रूप से

वर्ग दूरी का उपयोग करके परिभाषित किया जाता है, अर्थात, d(।.।) के बजाय d(|.|)^2। हम इस समस्या के लिए निम्नलिखित परिणाम प्राप्त करते हैं:

(8) हम सामाजिक रूप से निष्पक्ष k-माध्यिका और k-माध्य के लिए (3+ε) और (9+ε) सन्निकटन एल्गोरिदम डिज़ाइन करते हैं, FPT समय में f (k,ε).n^O(1), जहाँ f(k,ε) = (k/ε)^O(k) और n = |C ∪ L|।

(9) इसके अलावा, ये सन्निकटन गारंटी तंग हैं; अर्थात्, किसी अचर ε > 0 के लिए, एल्गोरिदम सामाजिक रूप से निष्पक्ष k-माध्यिका और k-माध्य के लिए (3 -ε) और (9 - ε) सन्निकटन गारंटी प्राप्त नहीं कर सकता है, FPT= W[2] मानते हुए।

आखिर में हम k-माध्यिका समस्या के लिए सन्निकटन परिणाम की कठोरता देते हैं, यूक्लिडियन अंतरिक्ष में जहां L= R^p और C ⊆ R^p। यह एक खुली समस्या का समाधान करता है जो अवस्थी आदि [18] के कार्यों में स्पष्ट रूप से प्रस्तुत किया गया है। हम निम्नलिखित प्राप्त करते हैं:

(10) एक स्थिर ε> 0 मौजूद है, जैसे कि O(log k) यूक्लिडियन k-माध्यिका समस्या को (1 + ε) से बेहतर कारक के रूप में अनुमानित नहीं किया जा सकता है, अद्वितीय खेल अनुमान (UGC) मानते हुए।

इसके अलावा, हम यूक्लिडियन k-माध्य/k-माध्यिका के लिए सन्निकटन की कठोरता का अध्ययन करते हैं, द्वि-मानदंड सेटिंग में। द्वि-मानदंड सेटिंग में, एल्गोरिदम को βk केंद्र (कुछ स्थिर β> 1 के लिए) आउटपुट करने की अनुमति है और सन्निकटन अनुपात की गणना इष्टतम k-माध्य/k-माध्यिका लागत के संबंध में की जाती है। हम निम्नलिखित परिणाम दिखाते हैं:

(11) किसी भी स्थिरांक $1 < \beta < 1.015$ के लिए, एक अचर $\varepsilon > 0$ मौजूद है जिस्से कि कोई $(1+\varepsilon)$ द्वि-मानदंड सन्निकटन एल्गोरिथम नहीं मौजूद हो सकता है, $O(\log k)$ यूक्लिडियन k-माध्यिका समस्या के लिए, अद्वितीय खेल अनुमान (UGC) मानते हुए।

(12) किसी भी अचर $1 < \beta < 1.28$ के लिए, एक अचर $\varepsilon > 0$ मौजूद है जिस्से कि कोई $(1+\varepsilon)$ द्वि-मानदंड सन्निकटन एल्गोरिथम नहीं मौजूद हो सकता है, $O(\log k)$ यूक्लिडियन k-माध्यिका समस्या के लिए, अद्वितीय खेल अनुमान (UGC) मानते हुए।

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Clustering is one of the most important tools for data analysis. The goal of clustering is to partition data objects into groups, called *clusters*, such that similar objects are in the same cluster and dissimilar ones are in different clusters. Clustering has many mathematical formulations and a wide range of known applications (see [138] and [96] for a brief survey). Defining the clustering problem formally requires us to quantify the notion of similarity/dissimilarity, and there are various ways of doing this. The most prominent mathematical formulations are k-*supplier*, k-*median*, and k-*means* formulations. Formally, the k-supplier problem is defined as follows:

Definition 1 (k-Supplier Problem)**.** *Let $(\mathcal{X}, d)$ be a metric space. Let k be any positive integer, and z be any positive real number. Given a set $L \subseteq \mathcal{X}$ of feasible facility locations, and a set $C \subseteq \mathcal{X}$ of clients, find a set $F \subseteq L$ of k facilities that minimises the cost:* supplier-cost$(F, C) \equiv \max_{x \in C} \left\{ \min_{f \in F} \left\{ d(x, f)^z \right\} \right\}$.

When $L = C$, the k-supplier problem is known as the k-*center* problem. The k-means and k-median problems are similar to each other. We combine the discussion on these problems by defining the k-*service* problem that encapsulates both these problems.

Definition 2 (k-Service Problem)**.** *Let $(\mathcal{X}, d)$ be a metric space. Let k be any positive integer, and z be any positive real number. Given a set $L \subseteq \mathcal{X}$ of feasible facility locations, and a set $C \subseteq \mathcal{X}$ of clients, find a set $F \subseteq L$ of k facilities that minimises the cost:* service-cost$(F, C) \equiv \sum_{x \in C} \left\{ \min_{f \in F} \left\{ d(x, f)^z \right\} \right\}$.

The k-service problem for $z = 1$ is known as the k-*median* problem, and for $z = 2$ the problem is known as the k-*means* problem.

The above definitions are motivated by the *facility location* problem [126]. The facility location problem differs from the k-service problem in two ways. Firstly, in the facility location problem, it is allowed to open any number of facilities. Secondly, the optimization function has an additional facility establishment cost for every open facility. Thus the k-service problem is equivalent to the facility location problem for a fixed number of facilities and 0 facility establishment cost. The k-supplier and k-service problems have natural applications in deciding appropriate locations for opening facilities such as hospitals, schools, and post offices in a geographical area [65, 7]. It ensures that no client pays a very high transportation cost for availing of a particular facility. Note that the clients that are assigned to the same facility belong to the same cluster, and the corresponding facility is known as their *cluster center*. Keeping this in mind, we will use the terms *facility* and *center* interchangeably from now on.

1.1 Classical (Unconstrained) Clustering

1.1.1 Polynomial time approximation

The k-supplier and k-service problems are NP-hard [94, 97, 91]. Therefore, we can not obtain an optimal solution to these problems in polynomial time unless $\mathsf{P} = \mathsf{NP}$. Therefore, we design *approximation algorithms* for these problems. Formally, an approximation algorithm is defined as follows:

Definition 3 (Approximation Algorithm)**.** *For constant $\alpha > 0$, an α-approximation algorithm*

for an optimization problem is an algorithm that outputs a solution with an objective value that is within an α factor of the optimal objective value.

In Table 1.1, we mention the best-known lower and upper bound approximation guarantees for these problems.

	k-Supplier	k-Center	k-Median	k-Means
Upper Bound	$\mathbf{3}^z$ [84]	$\mathbf{2}^z$ [84]	$\mathbf{2.675} + \varepsilon$ [35]	$\mathbf{9} + \varepsilon$ [8]
Lower Bound	$\mathbf{3}^z - \varepsilon$ [94]	$\mathbf{2}^z - \varepsilon$ [94]	$\mathbf{1} + \mathbf{2}/e - \varepsilon$ [91]	$\mathbf{1} + \mathbf{8}/e - \varepsilon$ [91]

Table 1.1: The best-known approximation guarantees for the clustering problems. The upper bound algorithms have polynomial running time. The lower bounds hold under the standard assumption of $\mathsf{P} \neq \mathsf{NP}$.

Clustering problems are also studied in an outlier setting. In practical scenarios, it often happens that a few clients are located at faraway locations from the rest of the clients, which are relatively close to each other. The far-located clients are called *outliers*. The presence of outliers forces the algorithm to open the facilities close to the outliers. Due to this, most of the clients have to pay high assignment costs. This leads to poor clustering of the dataset. To overcome this issue, we identify the outliers and cluster the dataset without these outliers. This gives rise to the outlier k-supplier and k-service problems. The outlier k-supplier and k-service problems are defined as follows:

Definition 4 (Outlier k-Supplier Problem)**.** *Let $(\mathcal{X}, d)$ be a metric space. Let k and m be any positive integers, and z be any positive real number. Given a set $L \subseteq \mathcal{X}$ of feasible facility locations, and a set $C \subseteq \mathcal{X}$ of clients, find a subset $Z \subseteq C$ of size at most m clients and a set $F \subseteq L$ of k facilities such that the k-supplier cost of $C' := C \setminus Z$ is minimized:* supplier-cost$(F, C') \equiv \max_{j \in C'} \left\{ \min_{i \in F} \left\{ d(i,j)^z \right\} \right\}$

Definition 5 (Outlier k-Service Problem)**.** *Let $(\mathcal{X}, d)$ be a metric space. Let k and m be any positive integers, and z be any positive real number. Given a set $L \subseteq \mathcal{X}$ of feasible facility locations, and a set $C \subseteq \mathcal{X}$ of clients, find a subset $Z \subseteq C$ of size at most m clients*

and a set $F \subseteq L$ of k facilities such that the k-service cost of $C' \coloneqq C \setminus Z$ is minimized: service-cost$(F, C') \equiv \sum_{j \in C'} \left\{ \min_{i \in F} \left\{ d(i,j)^z \right\} \right\}$

In Table 1.2, we mention the best-known upper and lower bound approximation guarantees for the outlier clustering problems.

	k-Supplier	k-Center	k-Median	k-Means
Upper Bound	$\mathbf{3^z}$ [40]	$\mathbf{2^z}$ [37]	$\mathbf{7 + \varepsilon}$ [106]	$\mathbf{53 + \varepsilon}$ [106]
Lower Bound	$\mathbf{3^z - \varepsilon}$ [94]	$\mathbf{2^z - \varepsilon}$ [94]	$\mathbf{1 + 2/e - \varepsilon}$ [91]	$\mathbf{1 + 8/e - \varepsilon}$ [91]

Table 1.2: The best-known approximation guarantees for the outlier clustering problems. The upper bound algorithms have polynomial running time. The lower bounds hold under the standard assumption of $\mathsf{P} \neq \mathsf{NP}$.

1.1.2 FPT time approximation

The approximation guarantees can be improved by allowing the algorithm to have exponential running time in some parameters. Such algorithms are called FPT (fixed-parameter tractable) algorithms. Formally, a fixed-parameter-tractable algorithm is defined as follows:

Definition 6 (Fixed Parameter Tractable (FPT) Algorithm)**.** *An algorithm that outputs a solution for an optimization problem with input x and parameter k with running time $f(k) \cdot |x|^{O(1)}$ for some computable function f.*

For the k-supplier/median/means problem, a natural parameter is k: the number of clusters. Therefore, an FPT algorithm has running time of $f(k) \cdot n^{O(1)}$, which is polynomial for fixed (or constant) value of k. FPT algorithms have polynomial running time if the parameter under consideration is a constant. This may be relevant even to a practitioner since the parameter k is a small number in many real clustering scenarios. In Table 1.3, we mention the best-known lower and upper bound approximation guarantees for the non-outlier and outlier versions of the clustering problems with FPT running time.

	k-Supplier	k-Center	k-Median	k-Means
Upper Bound (non-outlier)	$\mathbf{3^z}$ [84]	$\mathbf{2^z}$ [84]	$\mathbf{1+2/e+\varepsilon}$ [53]	$\mathbf{1+8/e+\varepsilon}$ [53]
Lower Bound (non-outlier)	$\mathbf{3^z-\varepsilon}$ [79]	$\mathbf{2^z-\varepsilon}$ [79]	$\mathbf{1+2/e-\varepsilon}$ [53]	$\mathbf{1+8/e-\varepsilon}$ [53]
Upper Bound (outlier)	$\mathbf{3^z}$ [40]	$\mathbf{2^z}$ [37]	$\mathbf{1+2/e+\varepsilon}$ [6]	$\mathbf{1+8/e+\varepsilon}$ [6]
Lower Bound (outlier)	$\mathbf{3^z-\varepsilon}$ [79]	$\mathbf{2^z-\varepsilon}$ [79]	$\mathbf{1+2/e-\varepsilon}$ [53]	$\mathbf{1+8/e-\varepsilon}$ [53]

Table 1.3: The best known FPT time approximation guarantees for the clustering problems. The non-outlier versions are parameterized by k, and the outlier versions are parameterized by both k and m. The lower bounds hold under the standard complexity theory assumption of $\mathsf{W}[2] \neq \mathsf{FPT}$. The lower bounds for the outlier versions simply follow from their non-outlier counterparts since an outlier version with $m = 0$ is equivalent to the non-outlier version.

In this work, we design FPT (in k and m) time $(3+\varepsilon)$ and $(9+\varepsilon)$ approximation algorithms for the outlier k-median and k-means problems, respectively. The running time of the algorithm is $O(n \cdot ((k+m)/\varepsilon)^{O(k)})$. Prior to our work, the best known result was $(6+\varepsilon)$-approximation algorithm for the outlier k-means problem for the special case $C \subseteq L$, with FPT running time of $O\left(n \cdot \beta^k \left(\frac{k+m}{\varepsilon}\right)^k\right)$, for some constant $\beta > 0$ [80]. Our work improves upon this result as well; we design an algorithm that gives $(4+\varepsilon)$-approximation guarantee for the outlier k-means problem and an algorithm that gives a $(2+\varepsilon)$-approximation guarantee for the outlier k-median problem when $C \subseteq L$ with FPT running time $O\left(n \cdot \left(\frac{k+m}{\varepsilon}\right)^{O(k)}\right)$. Recently, Agrawal *et al.* [6] improved on our result and gave FPT time tight $(1+2/e+\varepsilon)$ and $(1+8/e+\varepsilon)$ approximation algorithms for the outlier k-median and k-means problems, respectively.

1.2 Constrained Clustering

For many real-world applications, the classical (unconstrained) k-supplier and k-service problems do not entirely capture the desired clustering properties. For example, consider the popular

k-*anonymity* principle [131]. The principle provides anonymity to a public database while keeping it meaningful at the same time. One way to achieve this is to cluster the data in such a way as to release only partial information related to the clusters obtained. Further, to protect the data from the *re-identification* attacks, the clustering should be done in such a way that each cluster gets at least r data points. This method is popularly known as r-*gather* clustering [5]. Similarly, we have the r-*capacity* clustering problem where in addition to minimizing the clustering cost, we have a constraint that no cluster must contain more than r clients [111, 2]. This ensures that the load is almost equally distributed among the facilities. Likewise, there are many other constrained versions of the k-supplier/service problems namely *fault-tolerant* [92, 103], *fair* [26], *uncertain* [60], ℓ-*diversity* [72, 110], etc. Surprisingly, for many of the constrained clustering problems, no polynomial time constant approximation algorithm is known. One such classical problem is the r-*capacity* k-*median/means problem*. Designing a polynomial time constant approximation algorithm for this problem has been a popular open problem for over two decades. Even for the other constrained clustering problems for which polynomial time constant approximation algorithms are known, unfortunately, the approximation guarantees are very high. Therefore, we aim to obtain better (possibly tight) approximation guarantees for the problems by designing FPT time algorithms.

In the past, many constrained versions of the clustering problems were studied separately as independent problems. Recently, Ding and Xu [72] gave a unified framework for these problems that they called the *constrained clustering* framework. They proposed this unified framework in the context of the k-median and k-means problems in the continuous Euclidean spaces where $L = \mathbb{R}^p$ and C is a finite subset of $\mathbb{R}^p$. The authors designed FPT time $(1+\varepsilon)$-approximation algorithms for a range of constrained clustering problems in continuous Euclidean space with running time $O(np \cdot (k/\varepsilon)^{poly(k/\varepsilon)})$. In this work, we extend the framework to k-supplier objective and to general discrete metric spaces. We also extend the framework to the outlier versions of constrained clustering problems. Next, we briefly discuss the constrained clustering frame-

work and the results obtained for metric k-supplier and k-service objectives.

1.2.1 Constrained clustering framework: *k-supplier/center*

Let $\mathbb{O} = \{O_1, \ldots, O_k\}$ be any arbitrary partitioning of the client set C. Let $F \subseteq L$ be any set of k facilities. Let f_i^* be a facility in F that minimizes the 1-supplier cost of partition O_i. That is, f_i^* is the facility in F that minimises the cost: $\max_{x \in O_i}\{d(x, f_i^*)^z\}$. Then, the k-supplier cost of the partitioning $\mathbb{O}$ with respect to the facility set F is given as follows:

$$\Psi(F, \mathbb{O}) \equiv \max_{i=1}^{k} \left\{ \max_{x \in O_i} \left\{ d(x, f_i^*)^z \right\} \right\}$$

In other words, a partition O_i is completely assigned to a facility location f_i^* in F, and the assignment cost of every client in O_i is measured with respect to f_i^*. Then, $\Psi(F, \mathbb{O})$ is simply the maximum assignment cost over all the clients. Furthermore, the optimal k-supplier cost of $\mathbb{O}$ is given as follows: $\Psi^*(\mathbb{O}) \equiv \min_{k\text{-center-set } F} \Psi(F, \mathbb{O})$. Now, suppose that we are given a collection $\mathbb{S} = \{\mathbb{O}_1, \ldots, \mathbb{O}_t\}$ of t different partitionings of C. The goal of the constrained clustering problem is to find a partitioning in $\mathbb{S}$ that has the minimum k-supplier cost. Formally, we define the problem as follows:

Definition 7 (Constrained k-Supplier Problem)**.** *Let $(\mathcal{X}, d)$ be a metric space, k be any positive integer, and z be any positive real number. Given a set $L \subseteq \mathcal{X}$ of feasible facility locations, a set $C \subseteq \mathcal{X}$ of clients, and a set $\mathbb{S}$ of feasible partitionings of C, find a partitioning $\mathbb{O} = \{O_1, O_2, \ldots, O_k\}$ in $\mathbb{S}$, that minimizes the cost function: $\Psi^*(\mathbb{O}) \equiv \min_{k\text{-center-set } F} \Psi(F, \mathbb{O})$.*

The above definition encapsulates the following constrained clustering problems that we study in this work: r-gather, r-capacity, balanced, chromatic, fault-tolerant, strongly private, ℓ-diversity, and fair k-supplier problems. For example, consider the r-gather clustering problem, in which the goal is to find a clustering $\mathbb{O} = \{O_1, \ldots, O_k\}$ of the client set such that the i^{th} cluster has at least r_i clients in it, for some constant $r_i \geq 0$. For this problem, the set $\mathbb{S}$ can be concisely

defined as $\mathbb{S} \coloneqq \{\mathbb{O} \mid \text{for every cluster } O_i \in \mathbb{O}, |O_i| \geq r_i\}$, where $\mathbb{O} = \{O_1, O_2, \ldots, O_k\}$ is a partitioning of the client set. The definitions of the other seven problems are given in Table 1.4.

#	**Problem**	**Description**
1.	r-Gather k-supplier problem	Given k positive integers: $r_1, \ldots, r_k$, find clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Psi^*(\mathbb{O})$ such that for all i, $\lvert O_i\rvert \geq r_i$
2.	r-Capacity k-supplier problem	Given k positive integers: $r_1, \ldots, r_k$, find clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Psi^*(\mathbb{O})$ such that for all i, $\lvert O_i\rvert \leq r_i$
3.	Balanced k-supplier problem	Given positive integers: $\ell_1, \ldots, \ell_k$, and $r_1, \ldots, r_k$, find clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Psi^*(\mathbb{O})$ such that for all i, $\ell_i \leq \lvert O_i\rvert \leq r_i$
4.	Chromatic k-supplier problem	Given that every client has an associated color, find a clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Psi^*(\mathbb{O})$ such that for all i, O_i should not have any two points with the same color.
5.	Fault-tolerant k-supplier problem	Given positive integer $l_x \leq k$ for every client $x \in C$, find a set F of k centers, such that the maximum assignment cost of x to l_x^{th} closest facility is minimized.
6.	Strongly private k-supplier problem	Given a partitioning $C_1, \ldots, C_\omega$ of the client set C, and a set of integers: $\{\ell_1, \ldots, \ell_\omega\}$, find a clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Psi^*(\mathbb{O})$ that satisfies $\lvert C_j \cap O_i\rvert \geq \ell_j$ for every $i \in [k]$ and $j \in [\omega]$.
7.	ℓ-Diversity k-supplier problem	Given a partitioning $C_1, \ldots, C_\omega$ of the client set C, a real number $\ell > 1$, find a clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Psi^*(\mathbb{O})$ such that the fraction of points belonging to the same partition inside O_i is $\leq 1/\ell$.
8.	Fair k-supplier problem	Given ω color classes $C_1, \ldots, C_\omega$ (not necessarily disjoint), such that every C_j is a subset of the client set C, and two fairness vectors $\alpha, \beta \in [0,1]^\omega$, find a clustering $\mathbb{O} = \{O_1, \ldots, O_k\}$ with minimum $\Psi^*(\mathbb{O})$ such that it satisfies that $\beta_j \cdot \lvert O_i\rvert \leq \lvert O_i \cap C_j\rvert \leq \alpha_j \cdot \lvert O_i\rvert$ for every $i \in [k]$ and $j \in [\omega]$.

Table 1.4: List of constrained k-supplier problems that we study in this work.

Note that we are considering the *soft* assignment version of the constrained k-supplier problem. That is, it is allowed to open more than one facility at any particular location in L. This version differs from the *hard* assignment version, where a single copy of a facility can be opened at any particular location in L. Note that the total number of open facilities in both versions is at most k.

Now, we describe a general algorithmic technique to solve any problem that satisfies the defi-

nition of the constrained k-supplier problem. More precisely, we show that any constrained k-supplier problem can be solved using two basic ingredients: the *list k-supplier problem* and a *partition algorithm*. The notion of the list k-supplier problem was formalized by Bhattacharya *et al.* [28] in the context of the k-median and k-means objectives. We extend the notion to the k-supplier objective as follows:

Definition 8 (List k-Supplier Problem)**.** *Let $\mathcal{I} = (L, C, k, d, z)$ be any instance of the k-supplier problem. The goal of the problem is: given $\mathcal{I}$, find a list $\mathcal{L}$ of k-center-sets (i.e., each element of the list is a set of k elements from L) such that for any partitioning $\mathbb{O} = \{O_1, \ldots, O_k\}$ of the client set C, the list $\mathcal{L}$ contains a k-center-set F such that $\Psi(F, \mathbb{O}) \leq \alpha \cdot \Psi^*(\mathbb{O})$ for $\alpha = 3^z$. For the k-center objective $\alpha = 2^z$.*

Furthermore, we define a partition algorithm as follows:

Definition 9 (Partition Algorithm)**.** *Let $\mathcal{I} = (L, C, k, d, z)$ be any instance of the k-supplier problem, and let $\mathbb{S} = \{\mathbb{O}_1, \mathbb{O}_2, \ldots, \mathbb{O}_t\}$ be a collection of clusterings of C. Given a center set $F \subseteq L$, a partition algorithm outputs a clustering in $\mathbb{S}$ that has the least clustering cost $\Psi(F, \mathbb{O})$ with respect to F.*

Note that the set $\mathbb{S}$ differs for different constrained k-supplier problems; therefore, the partition algorithm differs for different constrained k-supplier problems. The simplest example of the partition algorithm is for the unconstrained k-supplier problem. For the unconstrained k-supplier problem, the set $\mathbb{S}$ is the collection of all possible k-partitionings of C and the partition algorithm is simply the standard Voronoi partitioning algorithm.

It is not very difficult to show that an algorithm for the list k-supplier problem together with a partition algorithm for a constrained k-supplier problem gives 3^z approximation to that problem. Note that the algorithm for the list k-supplier problem is common to all the constrained clustering problems. However, the partition algorithm differs for different constrained k-supplier

problems. In this work, we design $k^{O(k)} \cdot n^{O(1)}$ time algorithm for the list k-supplier problem with list size $k^{O(k)} \cdot n$. Thus, we obtain the following results:

Theorem 1 (Main Result: k-Supplier)**.** *For any constrained version of the k-supplier problem that has a partition algorithm with running time T, there exists a 3^z approximation algorithm with running time $T \cdot k^{O(k)} \cdot n + O(n^2 \log n)$.*

Theorem 2 (Main Result: k-Center)**.** *For any constrained version of the k-center problem that has a partition algorithm with running time T, there exists a 2^z approximation algorithm with running time $T \cdot k^{O(k)} \cdot n + O(n^2 \log n)$.*

For all the problems given in Table 1.4, we design FPT time partition algorithms. For this, we reduce each of the partition problems to the circulation problems on flow networks. Since most of the clustering constraints can be modeled as lower and upper bound flow constraints on the edges of the flow network, a feasible flow through the network gives an assignment of the clients to the facility set satisfying the given clustering constraints. Thus using Theorems 1 and 2, we get FPT time 3^z-approximation for the k-supplier and 2^z-approximation for the k-center objective for **all the problems in Table 1.4**. This improves the state-of-art for almost all the problems on the list. The details of these improvements will be highlighted in Chapter 2.

At a high level, our algorithm for the list k-supplier problem is composed of the following two parts:

1. A $(1, O(\ln n))$ bi-criteria approximation algorithm for the unconstrained k-supplier problem. The algorithm outputs a set $S \subseteq L$ of $O(k \ln n)$ facilities such that the cost of assigning any client in C to the closest facility in S at most the optimal unconstrained k-supplier cost. We obtain this bi-criteria approximation algorithm by reducing the problem to the set-cover problem and using the $O(\log n)$-approximation algorithm for the

set-cover problem. A similar reduction has been used earlier to solve a different problem [134].

2. Then, we show that for any arbitrary partitioning $\mathbb{O}$ of C, there exists a k-sized subset $F \subseteq S$ that gives 3^z approximation for $\mathbb{O}$. We prove this using the triangle-inequality property of the metric spaces. Therefore, we create a list $\mathcal{L}$ of all possible k-sized subsets of S. The list $\mathcal{L}$ is the required solution to the list k-supplier problem with list size $= O(k \ln n)^k = k^{O(k)}n$.

We extend the constrained k-supplier framework to the outlier setting in an analogous manner. We design $(k+m)^{O(k)} \cdot n^{O(1)}$ time algorithm for the outlier version of the list k-supplier problem with list size $(k+m)^{O(k)} \cdot n$, where $n = |C \cup L|$ and m is the number of outliers. Moreover, we design FPT time partition algorithms for all the constrained k-supplier problems given in Table 1.4. Thus we get FPT time 3^z and 2^z approximation algorithms for the outlier versions of these problems corresponding to k-supplier and k-center objectives, respectively.

Lastly, we note that the obtained approximation guarantees are tight when parameterized by k for the non-outlier version and parameterized by k and m for the outlier version. That is, for any constant $\varepsilon > 0$, no algorithm can achieve $(3-\varepsilon)$ and $(2-\varepsilon)$ approximation guarantees for the constrained k-supplier and k-center problems, respectively, in FPT time, assuming FPT $\neq$ W[2]. These results follow from previous works: [78, 94, 74, 128, 42].

Note that the results corresponding to this subsection has already been published in Theoretical Computer Science, 2023 [86].

1.2.2 Constrained clustering framework: *k-median/means*

The constrained clustering framework for the k-median/means problem is analogous to the framework for the k-supplier problem, except here, we are considering the hard-assignment

version of the problem. Due to this, the algorithmic techniques are more complex than in the previous subsection. Let $\mathbb{O} = \{O_1, \ldots, O_k\}$ be any arbitrary partitioning of the client set C. Let $F \subseteq L$ be any set of k facilities. Then, the k-service cost of the partitioning $\mathbb{O}$ with respect to the facility set F is given as follows:

$$\Phi(F, \mathbb{O}) \equiv \min_{\text{permutation } \pi} \left\{ \sum_{i=1}^{k} \sum_{x \in O_i} d(x, f_{\pi(i)})^z \right\}.$$

Furthermore, the optimal k-service cost of $\mathbb{O}$ is given as follows: $\Phi^*(\mathbb{O}) \equiv \min_{k\text{-center-set } F} \Phi(F, \mathbb{O})$. Note that here we are considering the *hard* assignment version of the k-service problem. That is, we are not allowed to open more than one facility at any location in L. The soft assignment version is easier than the hard assignment version since we can reduce the soft-assignment version to the hard-assignment version by creating k copies of every location in L. Now, suppose that we are given a collection $\mathbb{S} = \{\mathbb{O}_1, \ldots, \mathbb{O}_t\}$ of t different partitionings of C. The goal of the constrained k-service problem is to find a partitioning in $\mathbb{S}$ that has the minimum k-service cost. Formally, we define the problem as follows:

Definition 10 (Constrained k-Service Problem)**.** *Let $(\mathcal{X}, d)$ be a metric space, k be any positive integer, and z be any positive real number. Given a set $L \subseteq \mathcal{X}$ of feasible facility locations, a set $C \subseteq \mathcal{X}$ of clients, and a set $\mathbb{S}$ of feasible partitionings of C, find a partitioning $\mathbb{O} = \{O_1, O_2, \ldots, O_k\}$ in $\mathbb{S}$, that minimizes the cost function: $\Phi^*(\mathbb{O}) \equiv \min_{k\text{-center-set } F} \Phi(F, \mathbb{O})$.*

In Table 1.4, we defined eight constrained clustering problems with respect to the k-supplier objective. We study the same problems with respect to the k-service objective. We use a technique similar to the one discussed in the previous subsection. Any constrained k-service problem can be solved using two basic ingredients: the *list k-service problem* and a *partition algorithm*.

Definition 11 (List k-Service Problem)**.** *Let $\mathcal{I} = (L, C, k, d, z)$ be an arbitrary instance of*

the k-service problem, $\mathbb{O} = \{O_1, \ldots, O_k\}$ be any arbitrary clustering of the client set C, and $0 < \varepsilon \leq 1$ be an arbitrary constant. The goal of the problem is to find a list $\mathcal{L}$ of k-center-sets (i.e., each element of the list is a set of k elements from L) such that with probability at least $1 - 1/n$, the list $\mathcal{L}$ contains a k-center-set F such that $\Phi(F, \mathbb{O}) \leq \alpha \cdot \Phi^*(\mathbb{O})$ for $\alpha = 3^z + \varepsilon$ and $n = |C \cup L|$. For the special case when $C \subseteq L$, the approximation guarantee is $\alpha = 2^z + \varepsilon$.

Furthermore, we define a partition algorithm as follows:

Definition 12 (Partition Algorithm)**.** *Let $\mathcal{I} = (L, C, k, d, z)$ be any instance of the k-service problem, and let $\mathbb{S} = \{\mathbb{O}_1, \mathbb{O}_2, \ldots, \mathbb{O}_t\}$ be a collection of clusterings of C. Given a center set $F \subseteq L$, a partition algorithm outputs a clustering in $\mathbb{S}$ that has the least clustering cost $\Phi(F, \mathbb{O})$ with respect to F.*

It is easy to show that an algorithm for the list k-service problem together with a partition algorithm for a constrained k-service problem gives $(3^z + \varepsilon)$-approximation to that problem. Note that the algorithm for the list k-service problem is common to all the constrained k-service problems. However, the partition algorithm differs for different constrained k-service problems. In this work, we design $O\left(n \cdot (k/\varepsilon)^{O(k z^2)}\right)$ time algorithm for the list k-service problem with list size $O\left((\log n) \cdot (k/\varepsilon)^{O(k z^2)}\right)$. Thus, we obtain the following results:

Theorem 3 (Main Result: k-Median)**.** *For any constrained version of the k-median problem that has a partition algorithm with running time T, there exists a $(3 + \varepsilon)$-approximation algorithm that succeeds with probability at least $1 - 1/n$ and has running time $T \cdot (k/\varepsilon)^{O(k)} \cdot (\log n) + O\left(n \cdot (k/\varepsilon)^{O(k)}\right)$. For the special case when $C \subseteq L$, the algorithm gives $(2+\varepsilon)$-approximation guarantee.*

Theorem 4 (Main Result: k-Means)**.** *For any constrained version of the k-means problem that has a partition algorithm with running time T, there exists a $(9 + \varepsilon)$-approximation algorithm that succeeds with probability at least $1 - 1/n$ and has running time $T \cdot (k/\varepsilon)^{O(k)} \cdot (\log n) +$*

$O\left(n \cdot (k/\varepsilon)^{O(k)}\right)$. *For the special case when* $C \subseteq L$*, the algorithm gives* $(4+\varepsilon)$*-approximation guarantee.*

Moreover, we design FPT time partition algorithms for the k-service version of the constrained problems given in Table 1.4. The partition algorithms for these problems are similar to their k-supplier counterparts. Thus we get FPT time $(3^z + \varepsilon)$-approximation algorithms for the k-service objective for **all the problems in Table 1.4**. This improves the state-of-art for almost all problems in the table. The details of these improvements will be highlighted in Chapter 3.

The main challenge is to design an FPT time algorithm for the list k-service problem. In the previous subsection, we briefly described the algorithm for the list k-supplier problem. We can design a similar algorithm for the list k-service problem. However, that algorithm has many limitations; one of the limitations is that it can not be extended to the outlier setting. Another limitation is that the algorithm only holds for the soft-assignment version of the problem. To overcome these limitations, we use a sampling-based approach that is similar to the algorithm of Bhattacharya *et al.* [28]. The algorithm of Bhattacharya *et al.* [28] gives $(1+\varepsilon)$-approximation guarantee in FPT time for the constrained k-median and k-means problems in the continuous Euclidean space. In the continuous Euclidean space, the facility set $L = \mathbb{R}^p$ and C is a finite subset of $\mathbb{R}^p$. Our work differs from it in the following ways:

1. Working in a metric space instead of the Euclidean space poses challenges as some of the main tools used for analysis in the Euclidean setting cannot be used in metric spaces. We carefully devise and prove new sampling lemmas that make the high-level analysis of Bhattacharya *et al.* [28] go through.

2. Bhattacharya *et al.* [28] gave an algorithm for the list-k-means problem with list size $|\mathcal{L}| = (k/\varepsilon)^{O(k/\varepsilon)}$ and running time $O(np|\mathcal{L}|)$, where p is the dimension of the Euclidean space and $n = |C|$. Their algorithm explores a rooted tree of size $(k/\varepsilon)^{O(k/\varepsilon)}$ and depth k where the degree of every non-leaf vertex is $(k/\varepsilon)^{O(1/\varepsilon)}$. Every node in

this tree has an associated center, and the path from the root to a leaf node gives one of the k-center-sets for the output list. The algorithm has an unavoidable iteration of depth k since their analysis works only when the centers are picked *one-by-one* in k iterations. We circumvent this inherent restriction by using a constant factor approximate solution F to the unconstrained k-means/median problem for the given dataset (C, L). That is, service-cost$(F, C) \leq \alpha \cdot$ OPT, where OPT denotes the optimal unconstrained k-means/median cost. Then a distance-based sampling algorithm runs in a *single* iteration where poly(k/ε) points from C are D^z-sampled with respect to F. In D^z-sampling technique, a point x from the client set C is sampled with probability $\frac{\text{service-cost}(F,\{x\})}{\text{service-cost}(F,C)} = \frac{\min_{f\in F} d(f,x)^z}{\sum_{y\in C}\min_{f\in F} d(f,y)^z}$. Thus we obtain the list $\mathcal{L}$ in a "single shot". This technique helps us in designing a constant-pass streaming algorithm for the problem.

3. We study the hard-assignment version of the constrained k-service problem, which is harder than the soft-assignment version of the problem. The hard and soft assignment versions are equivalent in the continuous Euclidean space where $L = \mathbb{R}^p$. The reason is that if two facilities are opened at the same facility location, then one of the facilities can be moved by an infinitely small distance to convert a soft assignment to a hard assignment. Therefore, the previous works on the constrained clustering problem did not consider the distinction between soft and hard assignments.

4. Our techniques generalise for distance function $d(.,.)^z$. That is, for any positive real number z, if the cost of a client is defined by $d(.,.)^z$ instead of $d(.,.)$ or $d(.,.)^2$, then our algorithm gives 3^z approximation guarantees for the problems in discrete metric spaces and $(1+\varepsilon)$-approximation in continuous Euclidean space. The previous work only studies the problems with respect to the distance functions $d(.,.)$ and $d(.,.)^2$.

5. We extend the constrained k-service framework to the outlier setting. We design $((k+m)z/\varepsilon)^{O\left(kz/\varepsilon^{O(z)}\right)} \cdot n^{O(1)}$ time algorithm for the outlier version of the constrained k-service problem, where $n = |C \cup L|$ and m is the number of outliers. The algorithm

gives 3^z approximation guarantee in discrete metric space and $(1+\varepsilon)$-approximation guarantee in continuous Euclidean space.

In the second point, we mention that we design a streaming algorithm for constrained clustering problems. Streaming algorithms are useful when dealing with data sets that are too large to fit in the main memory or RAM (Random Access Memory). In clustering applications, we often come across large data sets (see [90, 12] for the related examples). Therefore, the data is stored in secondary storage devices instead of the main memory. This makes classical data processing algorithms extremely time-consuming. This motivates the study of streaming clustering algorithms. A streaming algorithm makes linear scans (usually one) over the data and uses limited memory (usually logarithmic in the input size). We design constant pass, log-space, FPT time streaming algorithms for the constrained k-median and k-means problems. We design these algorithms both in the continuous Euclidean and discrete metric spaces. Note that for the unconstrained k-median/means problems, there already exists various constant-pass log-space streaming algorithms [44, 32, 90].

Note that in this work, we do not study the k-supplier problem in the hard-assignment setting, streaming setting, and continuous Euclidean setting. However, studying these problems, which may require a different set of techniques, is one of our future work directions.

Note that the results corresponding to this subsection has already been published in International Symposium on Parameterized and Exact Computation (IPEC 2020) [88] and in IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020) [29].

1.3 Socially Fair Clustering Problem

In recent years, the topic: *fairness* in *machine learning*, has gained considerable attention. For example, see [25] and [48] for the recent developments in this area. The main motivation is that

in many human-centric applications, the input data is biased towards a particular demographic group that may be based on age, gender, ethnicity, occupation, nationality, etc. We do not want algorithms to discriminate among different groups due to biases in the dataset. In other words, we aim to design *fair* algorithms for problems.

In the context of clustering, in particular the k-median/k-means/k-center clustering, various notions of fair clustering have recently been proposed (see [46, 26, 11, 27, 104, 45, 116]). One such notion is the socially fair clustering that is defined as follows:

Definition 13 (Socially Fair Clustering)**.** *We are given a set C of clients and set L of feasible facility locations in a metric space $(\mathcal{X}, d)$. There are ℓ groups (possibly non-disjoint) of clients $C_1, ..., C_\ell \subseteq C$ with weight function $w_j \colon C_j \to \mathbb{R}^+$ for each $j \in \{1, \ldots, \ell\}$. Let z be any real number ≥ 1. In socially fair clustering, the goal is to pick a facility set $F \subseteq L$ of size k so as to minimize the objective function:* $\max_{j\in[\ell]}$ service-cost(F, C_j)*, which we call the fair cost:*

$$\text{fair-cost}(F, C) \equiv \max_{j\in[\ell]} \Big\{\text{service-cost}(F, C_j)\Big\},$$

where $\text{service-cost}(F, C_j) \equiv \sum_{x\in C_j} \Big\{w_j(x) \cdot \min_{f\in F} \big\{d(f, x)\big\}\Big\}$.

Informally, in the above definition, each group of clients C_j represents a particular demographic group, and the goal of the clustering is to find a set F of k centers such that maximum over the service costs of the groups is minimized. Thus it tries to ensure that one group does not pay a very high service cost in comparison to other groups. Note that for $\ell = 1$, the problem is equivalent to the k-service problem. For $\ell = |C|$ and if each client forms a singleton group, then the problem is equivalent to the k-supplier problem. Therefore, the socially fair clustering problem is a generalization of the k-supplier and k-service problems.

Also note that for the socially fair clustering problem, the *soft-assignment* and *hard-assignment* versions are equivalent. If two facilities f_1, f_2 are opened at a facility location $f \in L$, then

we can remove facility f_1 and re-assign the clients that were assigned to f_1, to f_2. The cost of the solution remains the same. Therefore, a solution to the soft-assignment version can be converted to a solution to the hard-assignment version. Moreover, a hard-assignment solution is trivially a soft-assignment solution. Therefore, both versions are equivalent in the context of the socially fair clustering problem.

Makarychev and Vakilian [120] and Chlamtác *et al.* [47] gave polynomial time $O\left(\frac{\log \ell}{\log \log \ell}\right)$ approximation algorithms for the socially fair k-median/means problem. In this work, we design a $(3^z + \varepsilon)$-approximation algorithm for the socially fair clustering problem with FPT time of $(zk/\varepsilon)^{O(k)} \cdot n^{O(1)}$. At a high level, the algorithm is composed of the following two parts:

1. A polynomial time $\left(1+\varepsilon, O\left((z/\varepsilon^2) \cdot \ln^2 n\right)\right)$ bi-criteria approximation algorithm for the socially fair clustering problem. The algorithm outputs a center set $S \subseteq L$ of size $O((kz/\varepsilon^2) \cdot \ln^2 n)$ and gives $(1+\varepsilon)$-approximation with respect to the optimal solution with k centers. We obtain this result by modifying a bi-criteria approximation algorithm for the unconstrained clustering problem [140], which in turn follows from an LP-rounding technique with respect to the most natural linear programming formulation of the problem.

2. We use the above bi-criteria algorithm to obtain a center set $S \subseteq L$ of size $O((kz/\varepsilon^2) \cdot \ln^2 n)$. We then show that there exists a k-sized subset $S' \subset S$ that gives $(3^z + \varepsilon)$ approximation. Note that since one needs to try all possible k-sized subsets of S, the overall running time of the algorithm has a multiplicative factor of $O\binom{|S|}{k}$. This results in an FPT algorithm.

Formally, we state the result as follows:

Theorem 5. *For the socially fair clustering problem, there is a randomized $(3^z + \varepsilon)$ approximation algorithm with* FPT *running time of $(zk/\varepsilon)^{O(k)} \cdot n^{O(1)}$ that succeeds with probability at least $1 - 1/n$.*

Furthermore, we show that these approximation guarantees are tight; we obtain the following lower bound result for the problem:

Theorem 6 (FPT Hardness for Parameter k)**.** *For any constant $z \geq 0$, $\varepsilon > 0$, and function $g\colon \mathbb{R}^+ \to \mathbb{R}^+$, the socially fair clustering problem can not be approximated to factor $(3^z - \varepsilon)$ in time $g(k) \cdot n^{O(1)}$ assuming $\mathsf{FPT} \neq \mathsf{W}[2]$ and in time $g(k) \cdot n^{o(k)}$ assuming ETH.*

Note that the results corresponding to this subsection has already been published in Information Processing Letters, 2023 [85].

1.4 Hardness of Approximation: *Euclidean k-Median*

We study the k-median problem in the continuous Euclidean space where $L = \mathbb{R}^p$ and C is a finite subset of $\mathbb{R}^p$. The cost of assigning a client $x \in C$ to a facility $f \in L$ is the Euclidean distance $\|x - f\|$. It is known that for general metric spaces, the k-median problem is hard to approximate to a factor less than $1 + 2/e$ [91, 97]. However, no hardness of approximation result is known for the Euclidean k-median problem. Resolving the hardness of approximation for the Euclidean k-median problem was left as an open problem in the work of Awasthi *et al.* [19]. In Table 1.5, we mention the best-known results for the Euclidean k-median and k-means problem.

Awasthi *et al.* [19] proved the first hardness of approximation result for the Euclidean k-means problem. They asked whether their techniques for proving the inapproximability results for Euclidean k-means can be used to prove the hardness of approximation result for the Euclidean k-median problem. Quoting from their paper,

> "*It would also be interesting to study whether our techniques give hardness of approximation results for the Euclidean k-median problem.*"

In this work, assuming Unique Games Conjecture (UGC), we solve this open problem by ob-

	Fixed k		Fixed d		General	
	Lower Bound	Upper Bound	Lower Bound	Upper Bound	Lower Bound	Upper Bound
k-Median	OPEN	$(1+\varepsilon)$ [107]	NP-hard [123]	$(1+\varepsilon)$ [54]	$(1+\varepsilon)$ **(this work)**	**2.633** [8]
k-Means	NP-hard [13]	$(1+\varepsilon)$ [107]	NP-hard [117]	$(1+\varepsilon)$ [54]	**1.07** [51]	**6.129** [89]

Table 1.5: The known approximation guarantees for the Euclidean k-Median and k-Means problems. For fixed k or d, all the mentioned $(1+\varepsilon)$-approximation algorithms have FPT running time. In the outlier setting, no upper bound result or any better lower bound result is known yet.

taining the hardness of approximation result for the Euclidean k-median problem. The following is one of the main results of this work.

Theorem 7 (k-Median Hardness)**.** *There exists a constant $\varepsilon > 0$ such that the Euclidean k-median problem cannot be approximated to a factor better than $(1+\varepsilon)$, assuming the Unique Games Conjecture.*

We build on the techniques of Awasthi *et al.* [19] to prove the inapproximability result for the Euclidean k-median problem. The authors gave a reduction from the *vertex cover problem* to the Euclidean k-means problem. We use the same construction to reduce the *vertex cover problem* to the Euclidean k-median problem. However, to show the correctness of the reduction, we need to analyze the cost of a cluster in the Euclidean space. This is the first obstacle one encounters in this direction; unlike the 1-mean problem, there does not exist a closed-form expression for the 1-median problem. Therefore, we don't know the exact value of the optimal cost of a given 1-median instance. The 1-median problem, popularly known as the *Fermat-Weber* problem [75], is a difficult problem, and designing efficient approximation algorithms for this problem is a separate line of research in itself – see for e.g. [105, 137, 38, 33, 49]. We overcome this barrier by obtaining upper and lower bounds on the optimal 1-median cost and showing that these bounds suffice for our purpose. More concretely, to upper bound the optimal 1-median

cost, we use the centroid as the 1-median and compute the 1-median cost with respect to the centroid. To obtain a lower bound on the 1-median cost of a cluster, we use a decomposition technique to break a cluster into smaller subclusters. Here we use a simple observation that the optimal 1-median cost of a cluster is at least the sum of the optimal 1-median costs of the subclusters. For the small subclusters, we compute exact or good approximate lower bounds on the 1-median cost. This gives a good approximate lower bound for any arbitrary cluster. Using this approximation, we show hardness of approximation for the Euclidean k-median problem assuming Unique Games Conjecture. To prove a similar result under a weeker assumption of $\mathsf{P} \neq \mathsf{NP}$, we would require a better approximation of the 1-median costs of clusters than the one done in this work. Note that Awasthi *et al.* [19] showed the hardness of Euclidean k-means problem under the assumption of $\mathsf{P} \neq \mathsf{NP}$ since it was easier to estimate the 1-means cost of the clusters.

Note that the results corresponding to this subsection has already been published in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021) [30].

1.5 Bi-Criteria Hardness of Approximation: *Euclidean k-Median and k-Means*

Having established the hardness of approximation results for k-means and k-median, the next natural step in the *beyond worst-case* discussion is to allow more flexibility to the algorithm. One possible relaxation is to allow an approximation algorithm to choose more than k centers, say, βk centers (for some constant $\beta > 1$), and produce a solution that is close to the optimal solution with respect to k centers. This is known as a bi-criteria approximation, and the following definition formalizes this notion.

Definition 14 ((α, β)-approximation algorithm)**.** *An algorithm $\mathcal{A}$ is called an (α, β) approx-*

imation algorithm for the Euclidean k-means/k-median problem if given any instance $\mathcal{I} = (C, k)$ with $C \subset \mathbb{R}^p$, $\mathcal{A}$ outputs a center set $F \subset \mathbb{R}^p$ of size βk that has the cost at most α times the optimal cost with k centers. That is,

$$\sum_{x \in C} \min_{f \in F} \{D(x, f)\} \leq \alpha \cdot \min_{\substack{F' \subseteq \mathbb{R}^p \\ |F'|=k}} \left\{ \sum_{x \in C} \min_{f \in F'} \{D(x, f)\} \right\}$$

For the Euclidean k-means problem, $D(q, r) \equiv \|q - r\|^2$ and for the k-median problem $D(q, r) \equiv \|q - r\|$.

One expects that as β grows, there would exist efficient (α, β)-approximation algorithms with smaller values of α. This is indeed observed in the work of Makarychev *et al.* [118]. For example, their algorithm gives a $(9+\varepsilon)$ approximation for $\beta = 1$; 2.59 approximation for $\beta = 2$; 1.4 approximation for $\beta = 3$. In other words, the approximation factor of their algorithm decreases as the value of β increases. Furthermore, their algorithm gives a $(1+\varepsilon)$-approximation guarantee with $O(k \log(1/\varepsilon))$ centers. Ideally, we would like to obtain a PTAS with a small violation of the number of output centers. More specifically, we would like to address the following question:

> *Does the Euclidean k-means or Euclidean k-median problem admit an efficient $(1 + \varepsilon, 1 + \varepsilon)$-approximation algorithm?*

Note that such type of bi-criteria approximation algorithms that outputs $(1 + \varepsilon)k$ centers have been extremely useful in obtaining a constant approximation for the *capacitated* k-median problem [111, 112] for which no true constant approximation is known yet [1]. Therefore, the above question is worth exploring. Note that here we are specifically aiming for a PTAS since the k-means and k-median problems already admit a constant factor approximation algorithm. In

[1] In the capacitated k-median/k-means problem there is an additional constraint on each center that it cannot serve more than a specified number of clients (or points).

this work, we give a negative answer to the above question by showing that there exists a constant $\varepsilon > 0$ such that an efficient $(1+\varepsilon, 1+\varepsilon)$-approximation algorithm for the k-means and k-median problems does not exist assuming the Unique Games Conjecture. The following two theorems state this result more formally.

Theorem 8 (Bi-criteria Hardness: k-Median)**.** *For any constant $1 < \beta < 1.015$, there exists a constant $\varepsilon > 0$ such that there is no $(1+\varepsilon, \beta)$-approximation algorithm for the Euclidean k-median problem assuming the Unique Games Conjecture.*

Theorem 9 (Bi-criteria Hardness: k-Means)**.** *For any constant $1 < \beta < 1.28$, there exists a constant $\varepsilon > 0$ such that there is no $(1+\varepsilon, \beta)$-approximation algorithm for the Euclidean k-means problem assuming the Unique Games Conjecture. Moreover, the same result holds for any $1 < \beta < 1.1$ under the assumption that $\mathsf{P} \neq \mathsf{NP}$.*

To prove these results, we use a similar reduction from the vertex cover problem and show that the reduction holds even if one is allowed to use βk centers, for some $\beta > 1$.

Note that the results corresponding to this subsection has already been published in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021) [30].

1.6 Notations

In this section, we define the notations that we frequently use in the thesis. We define the unconstrained k-supplier cost of a client set S with respect to a center set F as

$$\text{supplier-cost}(F, S) \coloneqq \max_{x \in S} \left\{ d(F, x)^z \right\}, \quad \text{where } d(F, x) = \min_{f \in F} \{ d(f, x) \}.$$

For a singleton set $\{f\}$, we denote supplier-cost($\{f\}, S$) shortly by supplier-cost(f, S). We define the unconstrained k-service cost of a client set S with respect to a center set F as

$$\text{service-cost}(F, S) \coloneqq \sum_{x \in S} \Big\{ d(F, x)^z \Big\}, \quad \text{where } \; d(F, x) = \min_{f \in F} \{d(f, x)\}.$$

For a singleton set $\{f\}$, we denote service-cost($\{f\}, S$) shortly by service-cost(f, S). For any instance $\mathcal{I} = (L, C, k, d, z)$ of the unconstrained k-supplier or k-service problem, we denote the optimal cost of the instance by OPT(L, C, k). It would be clear from the context if OPT(L, C, k) corresponds to the k-supplier or k-service objective.

1.7 Organization of Thesis

In Chapter 2, we design FPT time constant-approximation algorithms for the constrained k-supplier and k-center problems in general metric spaces. In Chapter 3, we design FPT time constant-approximation algorithms for the constrained k-median and k-means problems in general metric spaces. Moreover, we extend the algorithms to streaming and outlier settings. In Chapter 4, we study the outlier and streaming versions of the constrained k-median and k-means problems in continuous Euclidean spaces. We design FPT time $(1 + \varepsilon)$-approximation for these problems. In Chapter 5, we design FPT time constant-approximation algorithm for the socially fair clustering problem. In Chapter 6, we show the hardness of approximation results for the Euclidean k-median problem. Furthermore, we extend these hardness results to the bi-criteria versions of Euclidean k-median and k-means problems. In Chapter 7, we conclude the thesis with some open problems.

Chapter 2

Tight FPT Approximation for Constrained k-Center/Supplier

In this chapter, we study a range of *constrained* versions of the k-*supplier* and k-*center* problems. In the classical (*unconstrained*) k-supplier problem, we are given a set of clients C in a metric space $\mathcal{X}$, with distance function $d(.,.)$. We are also given a set of feasible facility locations $L \subseteq \mathcal{X}$. The goal is to open a set F of k facilities in L to minimize the maximum distance of any client to the closest open facility, i.e., minimize, supplier-cost$(F, C) \equiv \max_{j \in C} \big\{d(F, j)\big\}$, where $d(F, j)$ is the distance of client j to the closest facility in F. The k-center problem is a special case of the k-supplier problem where $L = C$. We study various constrained versions of the k-supplier problem such as: *capacitated*, *fault-tolerant*, ℓ-diversity, etc. These problems fall under a broad framework of *constrained clustering*. A unified framework for constrained clustering was proposed by Ding and Xu [72] in the context of the k-median and k-means objectives. We extend this framework to the k-supplier and k-center objectives in this chapter. This unified framework allows us to obtain results simultaneously for the following constrained versions of the k-supplier problem: r-*gather,* r-*capacity, balanced, chromatic, fault-tolerant, strongly private,* ℓ-*diversity, and fair* k-*supplier problems, with and*

without outliers. We design Fixed-Parameter Tractable (FPT) algorithms for these problems. FPT algorithms have polynomial running time if the parameter under consideration is a constant. This may be relevant even to a practitioner since the parameter k is a small number in many real clustering scenarios. We obtain the following results:

- We give 3 and 2 approximation algorithms for the constrained k-supplier and k-center problems, respectively, with FPT running time $k^{O(k)} \cdot n^{O(1)}$, where $n = |C \cup L|$. Moreover, these approximation guarantees are tight; that is, for any constant $\varepsilon > 0$, no algorithm can achieve $(3 - \varepsilon)$ and $(2 - \varepsilon)$ approximation guarantees for the constrained k-supplier and k-center problems in FPT time, assuming FPT $\neq$ W[2].

- We study the constrained k-supplier and k-center problems with outliers. Our algorithm gives 3 and 2 approximation guarantees for the constrained *outlier* k-supplier and k-center problems, respectively, with FPT running time $(k + m)^{O(k)} \cdot n^{O(1)}$, where $n = |C \cup L|$ and m is the number of outliers.

- Our techniques generalise for distance function $d(.,.)^z$. That is, for any positive real number z, if the cost of a client is defined by $d(.,.)^z$ instead of $d(.,.)$, then our algorithm gives 3^z and 2^z approximation guarantees for the constrained k-supplier and k-center problems, respectively.

2.1 Overview

We defined the k-supplier problem in Definition 1 in Chapter 1. We redefine it here for easy reference.

Definition 15 (k-Supplier Problem)**.** *Let $(\mathcal{X}, d)$ be a metric space. Let k be any positive integer, and z be any positive real number. Given a set $L \subseteq \mathcal{X}$ of feasible facility locations, and a set $C \subseteq \mathcal{X}$ of clients, find a set $F \subseteq L$ of k facilities that minimises the cost:* supplier-cost$(F, C) \equiv \max_{j \in C} \left\{ \min_{i \in F} \left\{ d(i,j)^z \right\} \right\}$.

When $L = C$, the problem is known as the *k-center* problem. In the above definition, we did not impose any constraints on the clients. Therefore, this version is also known as the *unconstrained* k-supplier problem. In this paper, we will use the term k-supplier problem and *unconstrained* k-supplier problem interchangeably. The k-supplier and k-center problems have natural applications in deciding the optimal location of placing facilities such as hospitals, schools, post offices, etc., in a geographical area [65, 7]. It ensures that no client pays a very high transportation cost for availing of a particular facility. Furthermore, these problems are extensively used for clustering large data sets in data mining, pattern recognition, information retrieval, etc. The clients assigned to the same facility belong to the same cluster, and the corresponding facility is known as the *cluster center*. Keeping this in mind, we will use the terms *facility* and *center* interchangeably from now on.

In many applications, additional constraints are imposed on the clusters. For example, every cluster must have at least r clients in privacy-preserving clustering. This problem is known as the *r-gather k-supplier/center* problem or the *lower bounded k-supplier/center* problem [4, 131]. Similarly, in many resource allocation problems, we impose an upper bound constraint on every facility location. That is, a facility can not serve more than u clients for some integer constant $u > 0$. This ensures that the load is almost equally distributed among the facilities. This problem is known as the *capacitated k-supplier/center* problem [24, 102, 61, 14]. Likewise, there are many other constrained versions of the k-supplier/center problems namely *fault-tolerant* [103], *fair* [26], *chromatic* [72], *ℓ-diversity* [110], *non-uniform* [22] problems, etc.

In the past, many constrained versions of the clustering problems were studied separately as independent problems. Recently, in 2015, Ding and Xu [72] gave a unified framework for these problems that they called the *constrained clustering* framework. They proposed this unified framework in the context of the k-median and k-means problems in the continuous Euclidean spaces. In this work, we extend the unified framework to the k-supplier and k-center objectives.

Using this, we design FPT time approximation algorithms for a range of constrained k-supplier problems. To put it another way, the unified framework allows us to use a common algorithmic technique for various constrained k-supplier problems. A natural question we need to address at the beginning of this work is: "*why should one be interested in designing approximation algorithms with* FPT *running time for these problems?*" The answer lies in the fact that the k-supplier and k-center problems are W[2]-hard parameterized by k [66, 78]. Therefore, we can not obtain exact algorithms for the k-supplier and k-center problems in FPT running time unless W[2] = FPT. Very recently, stronger FPT hardness results have been established for these problems. The following are these two results that easily follow from [78], [94], [74], [128] and [42].

Theorem 10. *For any constant $\varepsilon > 0$, $z > 0$, and any function $g\colon \mathbb{Z}^+ \to \mathbb{R}^+$, the k-supplier problem cannot be approximated to factor $(3^z - \varepsilon)$ in time $g(k) \cdot n^{o(k)}$ assuming ETH, and in time $g(k) \cdot n^{O(1)}$ assuming* W[2] $\neq$ FPT.

Theorem 11. *For any constant $\varepsilon > 0$, $z > 0$, and any function $g\colon \mathbb{Z}^+ \to \mathbb{R}^+$, the k-center problem can not be approximated to factor $(2^z - \varepsilon)$ in time $g(k) \cdot n^{o(k)}$ assuming ETH, and in time $g(k) \cdot n^{O(1)}$ assuming* W[2] $\neq$ FPT.

For the sake of completeness, we prove these two theorems in Section 2.6. Note that the above hardness results apply to all the constrained k-supplier problems that we study in this paper since the unconstrained version of the problem can be reduced to any of the constrained versions in polynomial time. Our main contribution is to give matching upper bounds for all the constrained k-supplier problems. That is, we design FPT time 3^z and 2^z approximation algorithms for various constrained k-supplier and k-center problems, respectively. Next, we define the constrained clustering framework for the k-supplier and k-center problems.

2.1.1 Constrained k-supplier framework

Let $\mathbb{O} = \{O_1, \ldots, O_k\}$ be any arbitrary partitioning of the client set C. Let $F \subseteq L$ be any set of k facilities. Let f_i^* be a facility in F that minimizes the 1-supplier cost of partition O_i. That is, f_i^* is the facility in F that minimises $\max_{x \in O_i}\{d(x, f_i^*)^z\}$. Then, the k-supplier cost of the partitioning $\mathbb{O}$ with respect to the facility set F is given as follows:

$$\Psi(F, \mathbb{O}) \equiv \max_{i=1}^{k} \left\{ \max_{x \in O_i} \left\{ d(x, f_i^*)^z \right\} \right\}$$

In other words, a partition O_i is completely assigned to a facility location f_i^* in F, and the assignment cost of every client in O_i is measured with respect to f_i^*. Then, $\Psi(F, \mathbb{O})$ is simply the maximum assignment cost over all the clients. Furthermore, the optimal k-supplier cost of $\mathbb{O}$ is given as follows: $\Psi^*(\mathbb{O}) \equiv \min_{k\text{-center-set } F} \Psi(F, \mathbb{O})$. Now, suppose that we are given a collection $\mathbb{S} = \{\mathbb{O}_1, \ldots, \mathbb{O}_t\}$ of t different partitionings of C. The goal of the constrained k-supplier problem is to find a partitioning in $\mathbb{S}$ that has the minimum k-supplier cost. Formally, we define the problem as follows:

Definition 16 (Constrained k-Supplier Problem)**.** *Let $(\mathcal{X}, d)$ be a metric space, k be any positive integer, and z be any positive real number. Given a set $L \subseteq \mathcal{X}$ of feasible facility locations, a set $C \subseteq \mathcal{X}$ of clients, and a set $\mathbb{S}$ of feasible partitionings of C, find a partitioning $\mathbb{O} = \{O_1, O_2, \ldots, O_k\}$ in $\mathbb{S}$, that minimizes the cost function: $\Psi^*(\mathbb{O}) \equiv \min_{k\text{-center-set } F} \Psi(F, \mathbb{O})$.*

The above definition encapsulates many constrained k-supplier problems. For example, consider the r-gather k-supplier problem, in which the goal is to find a clustering $\mathbb{O} = \{O_1, \ldots, O_k\}$ of the client set such that the i^{th} cluster has at least r_i clients in it, for some constant $r_i \geq 0$. For this problem, the set $\mathbb{S}$ can be concisely defined as $\mathbb{S} \coloneqq \{\mathbb{O} \mid \text{for every cluster } O_i \in \mathbb{O}, |O_i| \geq r_i\}$, where $\mathbb{O} = \{O_1, O_2, \ldots, O_k\}$ is a partitioning of the client set. We study seven other constrained k-supplier problems that satisfy the above definition of constrained k-supplier

problem. These problems are r-capacity, balanced, chromatic, fault-tolerant, strongly private, ℓ-diversity, and fair k-supplier problems. The definitions of these problems are given in Table 2.1.

We want to point out that we are considering the *soft* assignment version of the constrained k-supplier problems. That is, it is allowed to open more than one facility at any particular location in L. This version differs from the *hard* assignment version, where a single copy of a facility can be opened at any particular location in L. Note that the total number of open facilities in both versions is at most k. The soft assignment version is easier than the hard assignment version since the soft assignment version can be reduced to the hard assignment version by creating k copies of every location in L. The hardness results stated in Theorems 10 and 11 hold for both the soft and hard assignment versions of the constrained k-supplier problems.

Note that all the problems in the table satisfy the Definition 16 of the constrained k-supplier problem except the *fault-tolerant k-supplier* problem since its objective function is different from $\Psi(F, \mathbb{O})$. Next, we show that the fault-tolerant k-supplier problem can be reduced to the *chromatic k-supplier* problem, which satisfies the definition of the constrained k-supplier problem. In the fault-tolerant k-supplier problem, there is no explicit constraint imposed on any cluster. However, the cost function is different from the classical k-supplier problem. That is, for every client $x \in C$, we are given an integer $\ell_x \leq k$. Then, for a set $F = \{f_1, \ldots, f_k\}$ of facility locations, the assignment cost of a client x is proportional to the distance to its ℓ_x^{th} closest facility location in F. Thus, the overall cost of a fault-tolerant instance is given as: fault-tolerant-cost$(F, C) \equiv \max_{x \in C} \left\{ d'(F, x)^z \right\}$, where $d'(F, x)$ is the distance of x to the ℓ_x^{th} closest facility location in F. Therefore, the problem does not satisfy Definition 16 of the constrained k-supplier problem. We reduce an arbitrary instance of the fault-tolerant k-supplier problem to an instance of the chromatic k-supplier problem so that it satisfies the constrained k-supplier framework. The reduction is the same as the one used by Ding and Xu [72] for the k-median/means objective. Let $\mathcal{I} = (L, C, k, d, z, S_x)$ be any instance of the fault-tolerant

#	**Problem**	**Description**
1.	r-Gather k-supplier problem	Given k positive integers: $r_1, \ldots, r_k$, find clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Psi^*(\mathbb{O})$ such that for all i, $\lvert O_i\rvert \geq r_i$
2.	r-Capacity k-supplier problem	Given k positive integers: $r_1, \ldots, r_k$, find clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Psi^*(\mathbb{O})$ such that for all i, $\lvert O_i\rvert \leq r_i$
3.	Balanced k-supplier problem	Given positive integers: $\ell_1, \ldots, \ell_k$, and $r_1, \ldots, r_k$, find clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Psi^*(\mathbb{O})$ such that for all i, $\ell_i \leq \lvert O_i\rvert \leq r_i$
4.	Chromatic k-supplier problem	Given that every client has an associated color, find a clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Psi^*(\mathbb{O})$ such that for all i, O_i should not have any two points with the same color.
5.	Fault-tolerant k-supplier problem	Given positive integer $l_x \leq k$ for every client $x \in C$, find a set F of k centers, such that the maximum assignment cost of x to l_x^{th} closest facility is minimized.
6.	Strongly private k-supplier problem	Given a partitioning $C_1, \ldots, C_\omega$ of the client set C, and a set of integers: $\{\ell_1, \ldots, \ell_\omega\}$, find a clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Psi^*(\mathbb{O})$ that satisfies $\lvert C_j \cap O_i\rvert \geq \ell_j$ for every $i \in [k]$ and $j \in [\omega]$.
7.	ℓ-Diversity k-supplier problem	Given a partitioning $C_1, \ldots, C_\omega$ of the client set C, a real number $\ell > 1$, find a clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Psi^*(\mathbb{O})$ such that the fraction of points belonging to the same partition inside O_i is $\leq 1/\ell$.
8.	Fair k-supplier problem	Given ω color classes $C_1, \ldots, C_\omega$ (not necessarily disjoint), such that every C_j is a subset of the client set C, and two fairness vectors $\alpha, \beta \in [0,1]^\omega$, find a clustering $\mathbb{O} = \{O_1, \ldots, O_k\}$ with minimum $\Psi^*(\mathbb{O})$ such that it satisfies that $\beta_j \cdot \lvert O_i\rvert \leq \lvert O_i \cap C_j\rvert \leq \alpha_j \cdot \lvert O_i\rvert$ for every $i \in [k]$ and $j \in [\omega]$.

Table 2.1: List of constrained k-supplier problems with FPT time partition algorithms (see Section 2.5).

k-supplier problem, where $S_x = \{\ell_x \mid x \in C\}$ is a set of integers for the clients in C. Firstly, we color each client in C with different color. This coloring is given by a bijective function $f\colon C \to R$, where $R = \{r_1, \ldots, r_{|C|}\}$ is a set of $|C|$ different colors. Then, we create ℓ_x copies of each client x and assign each copy the same color as x. Let C' be this new client set and color on a client is denoted by function $g\colon C' \to R$. This completes the construction of the chromatic k-supplier instance. Let this new instance be $\mathcal{I}' = (L, C', k, d, z, g)$. Now, we show the following lemma:

Lemma 1. *Let $\mathcal{I} = (L, C, k, d, z, S_x)$ be any instance of the fault-tolerant k-supplier problem*

and $\mathcal{I}' = (L, C', k, d, z, g)$ be the corresponding instance of the chromatic k-supplier problems as defined above. For any set $F = \{f_1, \ldots, f_k\}$ of facility locations, the fault-tolerant k-supplier cost of C with respect to F is the same as the chromatic k-supplier cost of C' with respect to F.

Proof. The fault-tolerant k-supplier cost of C with respect to F is $\max_{x \in C} \left\{d'(F, x)^z\right\}$, where $d'(F, x)$ is the distance of x to its ℓ_x^{th} closest facility location in F. Now, let us evaluate the chromatic k-supplier cost of C'. For a client $x \in C$, let $\{x_1, \ldots, x_{\ell_x}\}$ denote the copies of x in C'. These copies share the same color. By the definition of the chromatic k-supplier problem, a cluster can not contain two clients of the same color. In other words, $\{x_1, \ldots, x_{\ell_x}\}$ must get assigned to different facility locations in F. Since all these clients are co-located, the assignment cost is minimized when each of them is assigned to one of the ℓ_x closest facility locations in F. Without loss of generality, let x_{ℓ_x} is the client that is assigned to ℓ_x^{th} closest facility location in F. Note that, x_{ℓ_x} has the maximum assignment cost among $\{x_1, \ldots, x_{\ell_x}\}$. This cost is the same as the assignment cost of $x \in C$ in the fault-tolerant k-supplier instance. Therefore, the overall chromatic k-supplier cost of the clients in C' is: $\max_{x \in C} \left\{d'(F, x)^z\right\}$. This cost is the same as the fault-tolerant k-supplier cost of C with respect to F. This completes the proof of the lemma. □

From the above lemma, we can also say that any α-approximate solution to the reduced chromatic k-supplier instance is also an α-approximate solution to the original fault-tolerant k-supplier instance, and vice-versa.

Now, we describe a general algorithmic technique to solve any problem that satisfies the definition of the constrained k-supplier problem. More precisely, we show that any constrained k-supplier problem can be solved using two basic ingredients: the *list k-supplier problem* and a *partition algorithm*. The notion of the list k-supplier problem was formalized by Bhattacharya *et al.* [28] in the context of the k-median and k-means objectives. We extend the notion

to the k-supplier objective as follows:

Definition 17 (List k-Supplier)**.** *Let $\mathcal{I} = (L, C, k, d, z)$ be any instance of the k-supplier problem. The goal of the problem is: given $\mathcal{I}$, find a list $\mathcal{L}$ of k-center-sets (i.e., each element of the list is a set of k elements from L) such that for any partitioning $\mathbb{O} = \{O_1, \ldots, O_k\}$ of the client set C, the list $\mathcal{L}$ contains a k-center-set F such that $\Psi(F, \mathbb{O}) \leq \alpha \cdot \Psi^*(\mathbb{O})$ for $\alpha = 3^z$. For the k-center objective $\alpha = 2^z$.*

Furthermore, we define a partition algorithm as follows:

Definition 18 (Partition Algorithm)**.** *Let $\mathcal{I} = (L, C, k, d, z)$ be any instance of the k-supplier problem, and let $\mathbb{S} = \{\mathbb{O}_1, \mathbb{O}_2, \ldots, \mathbb{O}_t\}$ be a collection of clusterings of C. Given a center set $F \subseteq L$, a partition algorithm outputs a clustering in $\mathbb{S}$ that has the least clustering cost $\Psi(F, \mathbb{O})$ with respect to F.*

Note that the set $\mathbb{S}$ differs for different constrained k-supplier problems; therefore, the partition algorithm differs for different constrained k-supplier problems. The simplest example of the partition algorithm is for the unconstrained k-supplier problem. For the unconstrained k-supplier problem, the set $\mathbb{S}$ is the collection of all possible k-partitionings of C and the partition algorithm is simply the standard Voronoi partitioning algorithm. In other words, the algorithm assigns a client to its closest facility location in F, and the obtained partitioning has the least clustering cost with respect to F.

Suppose we have an algorithm for the list k-supplier problem and a partition algorithm for a particular constrained k-supplier problem. Then we can obtain an approximation algorithm for that constrained k-supplier problem. The following theorem proves this result:

Theorem 12. *Let $\mathcal{I} = (L, C, k, d, z, \mathbb{S})$ be any instance of the constrained k-supplier problem, and let $A_{\mathbb{S}}$ be a partition algorithm for $\mathbb{S}$ with running time T_A. Let B be any algorithm for the list k-supplier problem with running time T_B. Then, there is an algorithm that outputs a*

clustering $\mathbb{O} \in \mathbb{S}$ that is an α-approximate solution for the constrained k-supplier instance $\mathcal{I}$. For the k-supplier objective, $\alpha = 3^z$, and for the k-center objective, $\alpha = 2^z$. Moreover, the running time of the algorithm is $O(T_B + |\mathcal{L}| \cdot T_A)$.

Proof. The algorithm is simple. We first run algorithm B to obtain a list $\mathcal{L}$. For every k-center-set in the list, the algorithm runs the partition algorithm $A_{\mathbb{S}}$ on it. Then the algorithm outputs a center set that gives the minimum clustering cost. Let F' be this k-center-set and $\mathbb{O}'$ be the corresponding clustering. We claim that $(F', \mathbb{O}')$ is an α-approximation for the constrained k-supplier problem.

Let $\mathbb{O}^*$ be an optimal solution for the constrained k-supplier instance $(L, C, k, d, z, \mathbb{S})$ and F^* denote the corresponding k-center-set. By the definition of the list k-supplier problem there is a k-center-set F in the list $\mathcal{L}$, such that $\Psi(F, \mathbb{O}^*) \leq \alpha \cdot \Psi(F^*, \mathbb{O}^*)$. Let $\mathbb{O} = A_{\mathbb{S}}(F) \in \mathbb{S}$ be the optimal clustering corresponding to F. Thus, $\Psi(F, \mathbb{O}) \leq \alpha \cdot \Psi(F^*, \mathbb{O}^*)$. Since $(F', \mathbb{O}')$ gives the minimum cost clustering in the list, we have $\Psi(F', \mathbb{O}') \leq \Psi(F, \mathbb{O})$. Therefore, $\Psi(F', \mathbb{O}') \leq \alpha \cdot \Psi(F^*, \mathbb{O}^*)$.

The running time analysis is also simple. T_B is the time to obtain the list $\mathcal{L}$. Then, the algorithm runs the partition procedure $A_{\mathbb{S}}$ for every center set in the list; the running time of this step is $|\mathcal{L}| \cdot T_A$. Picking a minimum cost clustering from the list takes $O(|\mathcal{L}|)$ time. Hence the overall running time is $O(T_B + |\mathcal{L}| \cdot T_A)$. □

The goal now becomes to design an algorithm for the list k-supplier problem and the partition algorithms for different constrained k-supplier problems. In Section 2.4, we design an FPT time algorithm for the list k-supplier problem. Formally, we state the result as follows:

Theorem 13. *Given an instance $\mathcal{I} = (L, C, k, d, z)$ of the k-supplier problem. There is an algorithm for the list k-supplier problem that outputs a list $\mathcal{L}$ of size at most $k^{O(k)} \cdot n$. Moreover, the running time of the algorithm is $k^{O(k)} \cdot n + O(n^2 \log n)$, which is* FPT *in k.*

Using Theorems 12 and 13, we further obtain the following two corollaries:

Corollary 1 (Main Result)**.** *For any constrained version of the k-supplier problem that has a partition algorithm with running time T, there exists a 3^z approximation algorithm with running time $T \cdot k^{O(k)} \cdot n + O(n^2 \log n)$.*

Corollary 2 (Main Result)**.** *For any constrained version of the k-center problem that has a partition algorithm with running time T, there exists a 2^z approximation algorithm with running time $T \cdot k^{O(k)} \cdot n + O(n^2 \log n)$.*

The remaining task is designing partition algorithms for different constrained k-supplier problems. Note that all the problems described in Table 2.1 satisfy the definition of the constrained k-supplier problem. Furthermore, in Section 2.5, we design FPT time partition algorithms for these problems. Therefore, the above two corollaries imply FPT time 3^z and 2^z approximation algorithms for these problems. For the first six problems in Table 2.1, the running time of the partition algorithms is $k^k \cdot n^{O(1)}$. For the seventh problem in the table, i.e., the ℓ-diversity k-supplier problem, the running time of the partition algorithm is $(\omega k)^{\omega k} \cdot n^{O(1)}$. For the last problem in the table, i.e., the fair k-supplier problem, the running time of the partition algorithm is $(k\Gamma)^{O(k\Gamma)} \cdot n^{O(1)}$, where Γ is the number of distinct collection of color classes induced by the colors of clients. We define the notation Γ more clearly in Section 2.5.2. Thus, we get FPT time 3^z and 2^z approximation guarantees for these problems with respect to k-supplier and k-center objectives, respectively. Formally, we state these results as follows:

Theorem 14. *There is an FPT time 3^z approximation algorithm for the following constrained versions of the k-supplier problem:*

1. *r-gather k-supplier problem*
2. *r-capacity k-supplier problem*
3. *Balanced k-supplier problem*
4. *Chromatic k-supplier problem*
5. *Fault-tolerant k-supplier problem*
6. *Strongly private k-supplier problem*

The running time of the algorithm is $k^{O(k)} \cdot n^{O(1)}$. Furthermore, for the k-center version, the approximation guarantee is 2^z.

All the abovementioned problems generalize the unconstrained k-supplier and k-center problems. Therefore, from Theorems 10 and 11, it follows that better approximation guarantees are not possible for the above problems in FPT time assuming ETH or $\mathsf{W}[2] \neq \mathsf{FPT}$. Thus, it settles the complexity of the above problems, parameterized by k. For the ℓ-diversity and fair k-supplier problems, we obtain the following results:

Theorem 15. *There is* FPT *time* 3^z *and* 2^z *approximation algorithms for the* ℓ*-diversity* k*-supplier and* k*-center problems, respectively, with running time* $(k\omega)^{O(k\omega)} \cdot n^{O(1)}$.

Theorem 16. *There is* FPT *time* 3^z *and* 2^z *approximation algorithms for the fair* k*-supplier and* k*-center problems, respectively, with running time* $(k\Gamma)^{O(k\Gamma)} \cdot n^{O(1)}$*, where* Γ *denote the number of distinct collection of color classes induced by the colors of clients. Moreover, if the color classes are pair-wise disjoint, then* $\Gamma = \omega$*, and the running time of the algorithm is* $(k\omega)^{O(k\omega)} \cdot n^{O(1)}$.

The ℓ-diversity k-supplier problem for $\omega = 1$ and $\ell = 1$, is equivalent to the unconstrained k-supplier problem. Also, the fair k-supplier problem for $\omega = 1$, $\beta_j = 0$, and $\alpha_j = 1$, is equivalent to the unconstrained k-supplier problem. Therefore, better approximation guarantees are not possible for these problems in FPT time parameterized by k and Γ (or ω). The statement simply follows from Theorems 10 and 11.

We extend the constrained k-supplier framework to the outlier setting in the next subsection. The discussion is analogous to the above discussion.

2.1.2 Constrained k-supplier framework with outliers

In practical scenarios, it often happens that a few clients are located at faraway locations from the majority of the clients. These clients are called *outliers*. The presence of outliers forces the algorithm to open the facilities close to the outliers. Due to this, the majority of the clients have to pay high assignment costs. This leads to poor clustering of the dataset. To overcome

this issue, we cluster the dataset without the outliers. This gives rise to the *outlier k-supplier* problem. A mathematical formulation of the problem was given by Charikar *et al.* [40] for which they gave a polynomial time 3-approximation algorithm. The following is the definition of the outlier k-supplier problem:

Definition 19 (Outlier k-Supplier)**.** *Let $(\mathcal{X}, d)$ be a metric space. Let k and m be any positive integers, and z be any positive real number. Given a set $L \subseteq \mathcal{X}$ of feasible facility locations, and a set $C \subseteq \mathcal{X}$ of clients, find a subset $Z \subseteq C$ of size at most m clients and a set $F \subseteq L$ of k facilities such that the k-supplier cost of $C' \coloneqq C \setminus Z$ is minimized:* supplier-cost$(F, C') \equiv \max_{j \in C'} \left\{ \min_{i \in F} \left\{ d(i,j)^z \right\} \right\}$.

Similarly, we generalize the definition of constrained k-supplier problem to its outlier version, as follows:

Definition 20 (Constrained Outlier k-Supplier Problem)**.** *Let (L, C, k, d, z, m) be any instance of the outlier k-supplier problem and $\mathbb{S}$ be any collection of partitionings such that any partitioning $\mathbb{O} \in \mathbb{S}$ is a partitioning of some subset $C' \subseteq C$ of size at least $|C| - m$. Find a clustering $\mathbb{O} = \{O_1, O_2, \ldots, O_k\}$ in $\mathbb{S}$, that minimizes the objective function: $\Psi^*(\mathbb{O}) \equiv \min\limits_{k\text{-center-set } F} \Psi(F, \mathbb{O})$.*

Furthermore, we define the *list outlier k-supplier problem* and *outlier partition algorithm*, as follows:

Definition 21 (List Outlier k-Supplier)**.** *Let $\mathcal{I} = (L, C, k, d, z, m)$ be any instance of the outlier k-supplier problem and $\mathbb{S}$ be the collection of all possible k-partitionings of every subset $C' \subseteq C$ of size at least $|C| - m$. Find a list $\mathcal{L}$ of k-center-sets (i.e., each element of the list is a set of k elements from L) such that for any partitioning $\mathbb{O} \in \mathbb{S}$, the list $\mathcal{L}$ contains a k-center-set F such that $\Psi(F, \mathbb{O}) \leq \alpha \cdot OPT(\mathbb{O})$ for $\alpha = 3^z$. For the k-center version $\alpha = 2^z$.*

Definition 22 (Outlier Partition Algorithm)**.** *Let $\mathcal{I} = (L, C, k, d, z, m, \mathbb{S})$ be any instance of the constrained outlier k-supplier problem. Given a center set F, an outlier partition algorithm outputs a clustering in $\mathbb{S}$ that has the least clustering cost with respect to F.*

Suppose we have an algorithm for the list outlier k-supplier problem and a partition algorithm for a particular constrained outlier k-supplier problem. Then, we can obtain an approximation algorithm for that constrained outlier k-supplier problem. The following theorem state this result and is analogous to Theorem 12 for the non-outlier version.

Theorem 17. *Let $\mathcal{I} = (L, C, k, d, z, m, \mathbb{S})$ be any instance of the constrained outlier k-supplier problem, and let $A_{\mathbb{S}}$ be any outlier partition algorithm for $\mathbb{S}$ with running time T_A. Let B be any algorithm for the list outlier k-supplier problem with running time T_B. Then, there is an algorithm that outputs a clustering $\mathbb{O} \in \mathbb{S}$ that is an α-approximate solution for the constrained outlier k-supplier instance $\mathcal{I}$. For the k-supplier objective, $\alpha = 3^z$, and for the k-center objective, $\alpha = 2^z$. Moreover, the running time of the algorithm is $O(T_B + |\mathcal{L}| \cdot T_A)$.*

Proof. The proof is analogous to Theorem 12. □

We design an FPT time algorithm for the list outlier k-supplier problem, parameterized by k and m. Formally, we state the result as follows:

Theorem 18. *Given an instance $\mathcal{I} = (L, C, k, d, z, m)$ of the outlier k-supplier problem. There is an algorithm for the list outlier k-supplier problem that outputs a list $\mathcal{L}$ of size at most $(k+m)^{O(k)} \cdot n$. Moreover, the running time of the algorithm is $(k+m)^{O(k)} \cdot n + O(n^2 \log n)$, which is* FPT *in k and m.*

Using Theorems 17 and 18, we obtain the following two corollaries:

Corollary 3 (Main Result)**.** *For any constrained version of the outlier k-supplier problem that has a partition algorithm with running time T, there exists a 3^z approximation algorithm with running time $T \cdot (k+m)^{O(k)} \cdot n + O(n^2 \log n)$.*

Corollary 4 (Main Result)**.** *For any constrained version of the outlier k-center problem that has a partition algorithm with running time T, there exists a 2^z approximation algorithm with running time $T \cdot (k+m)^{O(k)} \cdot n + O(n^2 \log n)$.*

We consider the outlier versions of all the problems described in Table 2.1. In Section 2.5, we design FPT time partition algorithms for the outlier versions of all these problems. Thus, we get FPT time 3^z and 2^z approximation algorithm for these problems with respect to the outlier k-supplier and k-center objectives, respectively. Formally, we state the results as follows:

Theorem 19. *There is an* FPT *time* 3^z *approximation algorithm for the following constrained versions of the outlier* k*-supplier problems:*

1. r*-gather outlier* k*-supplier problem*
2. r*-capacity outlier* k*-supplier problem*
3. *Balanced outlier* k*-supplier problem*
4. *Chromatic outlier* k*-supplier problem*
5. *Fault-tolerant outlier* k*-supplier problem*
6. *Strongly private outlier* k*-supplier problem*

The running time of the algorithm is $(k+m)^{O(k)} \cdot n^{O(1)}$*. Furthermore, for the* k*-center version, the approximation guarantee is* 2^z*.*

All the abovementioned problems generalize the unconstrained outlier k-supplier and k-center problems. Unsurprisingly, better approximation guarantees are not possible for these problems in FPT (in k and m) time, assuming ETH. We give a formal proof of this result in Section 2.7. Thus, it settles the complexity of the abovementioned k-supplier problems, parameterized by k and m. For the ℓ-diversity and fair outlier k-supplier problems, we obtain the following results:

Theorem 20. *There is* FPT *time* 3^z *and* 2^z *approximation algorithms for the* ℓ*-diversity outlier* k*-supplier and* k*-center problems, respectively, with running time* $(k\omega)^{O(k\omega)} \cdot (k+m)^{O(k)} \cdot n^{O(1)}$*.*

Theorem 21. *There is* FPT *time* 3^z *and* 2^z *approximation algorithms for the fair outlier* k*-supplier and* k*-center problems, respectively, with running time* $(k\Gamma)^{O(k\Gamma)} \cdot (k+m)^{O(k)} \cdot n^{O(1)}$*, where* Γ *denote the number of distinct collection of color classes induced by the colors of clients. Moreover, if the color classes are pair-wise disjoint, then* $\Gamma = \omega$*, and the running time of the algorithm is* $(k\omega)^{O(k\omega)} \cdot (k+m)^{O(k)} \cdot n^{O(1)}$*.*

The ℓ-diversity outlier k-supplier problem for $m = 0$, $\omega = 1$, and $\ell = 1$, is equivalent to the unconstrained k-supplier problem. Also, the fair outlier k-supplier problem for $m = 0$, $\omega = 1$,

$\beta_j = 0$, and $\alpha_j = 1$, is equivalent to the unconstrained k-supplier problem. Therefore, better approximation guarantees are not possible for these problems in FPT time parameterized by m, k, and Γ (or ω). The statement simply follows from Theorems 10 and 11.

Note that when the number of outliers is 0, any constrained version of the *outlier k-supplier problem* simply corresponds to its non-outlier variant. Therefore, in Section 2.4, we simply design an algorithm for the *list outlier k-supplier* problem, and in Section 2.5, we design partition algorithms for the outlier versions of the constrained k-supplier problems. That would imply the results for their non-outlier counterparts too. In the next section, we discuss the known results for all the problems described in Table 2.1.

2.2 Related Work

In this section, we mention the best-known results for different constrained k-supplier problems that we study in this paper. These results are summarized in Table 2.2.

We want to point out that the r-gather, r-capacity, and balanced k-supplier problems that we study in this paper impose cluster-wise constraints. However, the alternate definitions of the r-gather, r-capacity, and balanced k-supplier problems impose constraints on individual facility locations. Formally, the balanced k-supplier problem with location-wise constraints is defined as follows:

Definition 23 (Balanced k-Supplier Problem with Location-Wise Constraints)**.** *Given an instance $\mathcal{I} = (L, C, k, d, z)$ of the k-supplier problem, a lower bound function $g\colon L \to \mathbb{Z}_+$, and an upper bound function $h\colon L \to \mathbb{Z}_+$, find a set $F \subseteq L$ of k facility and assignment $\phi\colon C \to L$ that minimizes the assignment cost $\max_{x \in C} \left\{d(x, \phi(x))^z\right\}$ and satisfies that $g(f) \leq |\phi^{-1}(f)| \leq h(f)$ for every facility location $f \in F$.*

[1]Bera *et al.* [26] did not explicitly mention the results for the ℓ-diversity clustering problem. However, the results follow from [26] since the ℓ-diversity problem is a special case of fair clustering problem with disjoint color classes as noted by Bandyapadhyay *et al.* [21].

#	**Problem**	k**-Supplier Objective**		k**-Center Objective**	
		Without Outliers	With Outliers	Without Outliers	With Outliers
1.	r-Gather Clustering (non-uniform version with cluster-wise constraint)	-	-	-	-
	r-Gather Clustering (uniform version)	**3** [10] (polynomial time)	**5** [10] (polynomial time)	**2** [5] (polynomial time)	**4** [5] (polynomial time)
2.	r-Capacity Clustering (non-uniform version with cluster-wise constraint)	-	-	-	-
	r-Capacity Clustering (uniform version)	**9** [73] (polynomial time)	**13** [62] (polynomial time)	**5** [102] (polynomial time)	**13** [62] (polynomial time)
3.	Balanced Clustering (non-uniform version with cluster-wise constraint)	-	-	-	-
	Balanced Clustering (uniform version)	**9** [73] (polynomial time)	**13** [73] (polynomial time)	**6** [73] (polynomial time) **4** [68] (FPT time)	**13** [73] (polynomial time)
4.	Chromatic Clustering	-	-	-	-
5.	Fault Tolerant Clustering (non-uniform version)	-	-	-	-
	Fault Tolerant Clustering (uniform version)	**3** [103] (polynomial time)	-	**2** [103] (polynomial time)	**6** [95] (polynomial time)
6.	Strongly Private Clustering	**5** [127] (polynomial time)	-	**4** [127] (polynomial time)	-
7.	ℓ-Diversity Clustering [1]	**(5,3)** [26] (polynomial time)	-	**(3,3)** [93] (polynomial time)	-
8.	Fair Clustering (disjoint color classes)	**(5,3)** [26] (polynomial time)	-	**(3,3)** [93] (polynomial time)	-
	Fair Clustering (overlapping color classes)	$(\mathbf{5}, \mathbf{4\Delta + 3})$ [26] (polynomial time)	-	$(\mathbf{3}, \mathbf{4\Delta + 3})$ [93] (polynomial time)	-

Table 2.2: The known results for the constrained k-supplier/center problems with and without outliers for $z = 1$. We only mention the results for the *soft assignment* version of the problems; however, some of these results also hold for the *hard assignment* version that we did not mention explicitly for the sake of simplicity. No polynomial time constant factor approximation algorithm is known for the ℓ-diversity and fair clustering problems. The algorithms of [26] and [93] give 5 and 3 approximation guarantees corresponding to the k-supplier and k-center objective respectively; however, they violate the fairness constraint by an additive factor of $4\Delta + 3$, where Δ denotes the maximum number of color classes a client can be part of. When the color classes are disjoint, the algorithms violate the fairness constraint by an additive factor of 3.

The above definition also encapsulates the r-gather and r-capacity k-supplier problems with location-wise constraints. For the r-gather problem, $h(f) = |C|$ for every facility location $f \in L$, and for the r-capacity problem, $g(f) = 0$ for every facility location $f \in L$. Moreover, when every facility location has the same values of $g(f)$ and $h(f)$, then the problems are known as the *uniform r-gather, r-capacity, and balanced k-supplier problems*. It is easy to see that for the uniform version, the problem with location-wise constraints is equivalent to the problem with cluster-wise constraints. In other words, Definition 23 is the same as the definition given in Table 2.1 for the uniform case. To the best of our knowledge, the non-uniform variant of the problem with cluster-wise constraints has not been studied before. Furthermore, we believe that it is non-trivial to obtain any polynomial time reduction between the problems with cluster-wise constraints and location-wise constraints.

From Table 2.2, we note that most known approximation algorithms have polynomial running time; however, they give much worse approximation guarantees than 3 and 2 for the k-supplier and k-center versions, respectively. Furthermore, the approximation guarantees worsen for the outlier version of these problems. Among the FPT time algorithms, the only known result is due to Hu Ding [68] for the uniform balanced k-center problem without outliers. The author used the classical 2-approximation algorithm of Teofilo F. Gonzalez [84] as a subroutine to obtain the FPT time 4-approximation algorithm for the problem. The algorithms of [26] and [27] convert an α-approximate solution of the unconstrained k-supplier problem to an $(\alpha + 2)$-approximate solution of the fair k-supplier problem (problem 8 in Table 2.1) in polynomial time. These are, however, not generalized to the outlier and constrained clustering settings. Moreover, the algorithms do not give a tight approximation guarantee since $\alpha = (1+\varepsilon)$-approximate solution for the unconstrained k-supplier problem is not possible with only k centers. We approach the constrained problems through bi-criteria approximation for the unconstrained k-supplier problem. The bi-criteria relaxation allows us to obtain $\alpha = (1+\varepsilon)$-approximation using $O(k \log n)$ centers. On the other hand, we show that from the point of view of tight approximation guar-

antees, a bi-criteria approximation is not too restrictive. In particular, we show that any bi-criteria solution contains an approximate solution (i.e., k centers) for the constrained version of the problem. This gives a 3-approximation for the fair k-supplier problem in FPT time. In Section 2.4, we describe the algorithm in detail. Note that our algorithm holds for both the k-supplier and k-center objectives with and without outliers. Moreover, our algorithm works for several other constrained clustering problems described in Table 2.1, and we expect this list to grow further when new interesting problems are discovered. Moreover, these are the best approximation guarantees possible for these problems, parameterized by k (and m for the outlier version). Next, we discuss the known results for each of the problems mentioned in Table 2.1 in more detail:

1. **r-gather k-supplier problem:** For the r-gather k-supplier problem with and without outliers, Ahmadian and Swamy [10] gave 5 and 3 approximation algorithms, respectively (see Theorems 15 and 16 of [10]). The running time of their algorithm is polynomial in the input size, and it also holds for the non-uniform variant of the problem with location-wise constraints. For the r-gather k-center problem (with uniform lower bounds) with and without outliers, Aggarwal *at al.* [5] gave 4 and 2 approximation algorithms, respectively (see Section 2.4 of [5]). The running time of their algorithm is polynomial in the input size.

2. **r-capacity k-supplier problem:** For the r-capacity k-supplier and k-center problems (without outliers), An *et al.* [14] gave 11 and 9 approximation algorithms, respectively (see Theorems 1 and 6 of [14]). Their algorithm is polynomial time and also works for the non-uniform variant of the problem with location-wise constraints. For the r-capacity k-supplier problem (without outliers) with uniform upper bounds, Khuller and Sussmann [102] gave 5 approximation algorithm.[2] For the outlier version, Cygan and

[2]The authors call the soft assignment version of the r-capacity k-center problem as the *capacitated multi-k-center problem*.

Kociumaka [62] gave a 25-approximation algorithm for the r-gather k-supplier and k-center problems with non-uniform upper bounds imposed on facility locations. Moreover, the authors improved the approximation guarantee to 13 for the uniform r-gather k-supplier and k-center problems with outliers (see Theorem 1 and Corollary 1 of Theorem 1 of [62]). All the abovementioned algorithms have polynomial running time.

3. **Balanced k-supplier problem:** For the balanced k-supplier/center problem with non-uniform lower and upper bounds, no constant factor approximation is known yet. However, for the uniform lower bounds and non-uniform upper bounds on locations (without outliers), Ding *et al.* [73] gave 13 and 9 approximation algorithms for the balanced k-supplier and k-center problems, respectively. Furthermore, for the same variant, the authors gave a 25 approximation algorithm for both the balanced k-supplier and k-center problems with outliers. For the uniform lower bounds and uniform upper bounds (without outliers), the authors improved the approximation bounds to 9 and 6 for the balanced k-supplier and k-center problems, respectively. Furthermore, for the same variant, the authors gave a 13 approximation algorithm for both the balanced k-supplier and k-center problems with outliers. All the abovementioned algorithms have polynomial running time.

   For the balanced k-center problem (without outliers) with uniform lower and upper bounds, Hu Ding [68] gave a 4-approximation algorithm with FPT running time $k^{O(k)} \cdot n^{O(1)}$. In this work, we improve this approximation guarantee to 2. Furthermore, we give FPT time 3-approximation algorithm for the balanced k-supplier problem.

4. **Chromatic k-supplier problem:** For the chromatic k-supplier problem, no constant factor approximation is known yet. However, for the k-median and k-means objectives, FPT time approximation algorithms are known in the continuous Euclidean space and general discrete metric spaces [72, 70, 21].

5. **Fault-tolerant k-supplier problem:** In the fault-tolerant k-supplier problem, given a

facility set $F \subseteq L$, the cost of a client $x \in C$ is proportional to the distance to its ℓ_x closest facility location in F. If ℓ_x is the same for every x in C, then we call the problem the *uniform fault-tolerant k-supplier problem*. On the other hand, if ℓ_x is not the same for every x, then we call the problem the non-uniform fault-tolerant k-supplier problem. For the non-uniform version, no constant factor approximation is known yet. However, for the uniform fault-tolerant k-supplier and k-center problem (without outliers), Khuller *et al.* [103] gave 3 and 2 approximation algorithms, respectively. Recently, Inamdar and Varadarajan gave a 6 approximation algorithm for the uniform fault-tolerant *k-center* problem with outliers. However, no constant factor approximation is known for the uniform fault-tolerant *k-supplier* problem with outliers.

6. **Strongly private k-supplier problem:** This problem has recently been proposed by Rösner and Schmidt [127]. The authors gave 5 and 4 approximation algorithms for the k-supplier and k-center versions, respectively, without outliers. No constant factor approximation algorithm is known for the problem with outliers.

7. **ℓ-diversity k-supplier problem:** Bandyapadhyay *et al.* [21] noted that the ℓ-diversity clustering problem is a special case of the fair clustering problem when the color classes are disjoint, and $\alpha_j = 1/\ell$ and $\beta_j = 0$ for every color class $C_j \in \{C_1, \ldots, C_\omega\}$. Therefore, the problem admit 7 and 5 approximation algorithms for the k-supplier and k-center versions, respectively, without outliers [27]; however, the algorithms violate the constraint by an additive factor of 3, i.e., $0 \leq |O_i \cap C_j| \leq |O_i|/\ell + 3$ for every cluster $O_i \in \{O_1, \ldots, O_k\}$ and color class $C_j \in \{C_1, \ldots, C_\omega\}$.

   Another variant of the ℓ-diversity clustering problem was proposed by Li *et al.* [110]. Given an integer constant $\ell \geq 0$, the task is to find a clustering $\mathbb{O} = \{O_1, ..., O_t\}$ of the client set C with minimum $\Psi^*(\mathbb{O})$ such that for every cluster O_i, $|O_i| \geq \ell$ and O_i should not have any two clients with the same color. Note that, unlike other clustering problems, here, we do not have any restriction on the number of open centers. The authors gave a 2

approximation algorithm for the problem for $L = C$.

8. **Fair k-supplier problem:** In this problem, we are given ω color classes: $C_1, \ldots, C_\omega$ that are subsets of the client set C and two fairness vectors $\alpha, \beta \in [0,1]^\omega$. When the color classes are pair-wise disjoint and $\alpha_j = \beta_j = |C_j|/|C|$, then 7 and 5 approximation algorithms are known for the k-supplier and k-center versions, respectively, without outliers [27]. On the other hand, when the color classes are not necessarily disjoint, no true constant factor approximation algorithm is known yet. The existing algorithms give 5 and 3 approximation guarantees for the k-supplier and k-center objectives, respectively; however, they violate the fairness constraint by an additive factor of $4\Delta + 3$, where Δ denote the maximum number of groups a client can be part of [26, 93].

2.3 Notations

Let $\mathcal{I} = (L, C, k, d, z, m)$ denote any instance of the unconstrained outlier k-supplier problem. For the non-outlier version, $m = 0$. Let $F \subseteq L$ be any given center set. Let C' be any subset of C. Then, we define the unconstrained k-supplier cost of C' as:

$$\text{supplier-cost}(F, C') \equiv \max_{x \in C'} \left\{ d(F,x)^z \right\}, \quad \text{where } d(F,x) = \min_{f \in F} \left\{ d(f,x) \right\}.$$

If F is a singleton set $\{f\}$, then we simply use the notation: supplier-cost(f, C') instead of supplier-cost$(\{f\}, C')$. Let Z^* denote an optimal set of outliers, and F^* denote an optimal k-center set for the unconstrained outlier k-supplier instance $\mathcal{I}$. Then, we denote the optimal unconstrained outlier k-supplier cost of the instance by OPT.

That is, OPT $\equiv$ supplier-cost$(F^*, C \setminus Z^*)$.

Let $\mathbb{O} = \{O_1, \ldots, O_k\}$ be any partitioning of a subset C' of C. Let F be a set of facility locations. Let f_i^* be a facility in F that minimizes the 1-supplier cost of partition O_i. Then, the k-supplier cost of $\mathbb{O}$ with respect to the facility set F is given as follows:

$$\Psi(F, \mathbb{O}) \equiv \max_{i=1}^{k} \Big\{ \max_{x \in O_i} \{d(x, f_i^*)^z\} \Big\}.$$

In other words, a partition O_i is completely assigned to the facility location f_i^* in F, and the cost of every client in O_i is measured with respect to f_i^*. Then, $\Psi(F, \mathbb{O})$ is simply the maximum assignment cost over all the clients. Furthermore, the optimal k-supplier cost of $\mathbb{O}$ is given as follows: $\Psi^*(\mathbb{O}) \equiv \min_{k\text{-center-set } F} \Psi(F, \mathbb{O})$.

2.4 Algorithm for List Outlier k-Supplier

In this section, we design an FPT (in m and k) time algorithm for the list outlier k-supplier problem with running time $(k+m)^{O(k)} \cdot n^{O(1)}$. It implies a $k^{O(k)} \cdot n^{O(1)}$ time algorithm for the list k-supplier problem without outliers. The algorithm is surprisingly simple. Let $\mathcal{I} = (L, C, k, d, z, m)$ be any instance of the outlier k-supplier problem. The algorithm consists of the following two parts:

1. A $(1, O(\ln n))$ bi-criteria approximation algorithm for the outlier k-supplier problem. The algorithm outputs a set $S \subseteq L$ of $O(k \ln n)$ facilities and a set of $Z \subseteq C$ of m outliers such that every client in $C \setminus Z$ has assignment cost at most the optimal cost OPT of the outlier k-supplier instance $\mathcal{I}$.

2. For every outlier x in Z, let $g(x)$ denote a facility in L that is closest from x. Let $G = \{g(x) \mid x \in Z\}$ be the set of such facilities. Let C' be any arbitrary subset of C of size at least $|C| - m$. We then show that for any arbitrary partitioning $\mathbb{O}$ of C', there exists a k-sized subset $S' \subseteq S \cup G$ that gives 3^z approximation for $\mathbb{O}$. Therefore, we create a list $\mathcal{L}$ of all possible k-sized subsets of $S \cup G$. The list $\mathcal{L}$ is the required solution to the list outlier k-supplier problem. Moreover, the size of the list is $|\mathcal{L}| \leq O(m + k \ln n)^k = (k+m)^{O(k)} n$.

We discuss the above two parts in more detail in Sections 2.4.1 and 2.4.2, respectively.

2.4.1 Bi-criteria approximation

In this subsection, we design a $(1, O(\ln n))$ bi-criteria approximation algorithm for the outlier k-supplier problem. An (α, β) bi-criteria approximation algorithm is defined as follows:

Definition 24 (Bi-criteria Approximation)**.** *Let $\mathcal{I} = (L, C, k, d, z, m)$ be any instance of the outlier k-supplier problem. An (α, β) bi-criteria approximation algorithm is an algorithm that outputs a set $F' \subseteq L$ of βk facilities and a set $Z' \subseteq C$ of at most m outliers such that the cost of the client set $C \setminus Z'$ with respect to F' is at most α times the optimal cost of the instance. That is,*

$$\text{supplier-cost}(F', C \setminus Z') \leq \alpha \cdot \min_{\text{k-center-set } F \text{ and } |Z| \leq m} \Big\{\text{supplier-cost}(F, C \setminus Z)\Big\} = \alpha \cdot \text{OPT}$$

In the following lemma, we design an (α, β) bi-criteria approximation algorithm for the problem with $\alpha = 1$ and $\beta = O(\ln n)$.

Lemma 2. *Let $\mathcal{I} = (L, C, k, d, z, m)$ be any instance of the outlier k-supplier problem. Then, a $(1, O(\ln n))$ bi-criteria approximation algorithm exists for the problem. Moreover, the running time of the algorithm is $O(n^2 \log n)$.*

Proof. We reduce the outlier k-supplier instance to a *max k-coverage instance*. Some similar reductions have been previously carried out in the following works: [60, 134]. A max k-coverage instance is denoted by $(U, \mathscr{C}, k)$, where U denotes the universal set and the set $\mathscr{C}$ is a collection of subsets of U. The task of the max k-coverage problem is to select at most k sets from $\mathscr{C}$ whose union covers the maximum number of elements from U. For now, assume we know the optimal cost OPT of the outlier k-supplier instance. We then create a max k-coverage instance $(U, \mathscr{C}, k)$, where U corresponds to the client set C. In other words, for every client $j \in C$, there is an element e_j in U. The set $\mathscr{C}$ corresponds to the facility location set L. That is, for every facility location $f \in L$, there is a set $S_f \in \mathscr{C}$. Furthermore, an element e_j belongs

to a set S_f if and only if $d(j, f)^z \leq$ OPT.

Since there are k facilities in L that gives the optimal cost OPT for at least $|C| - m$ clients in C, there exists k sets in $\mathscr{C}$ that cover all but m elements of U. For the max k-coverage problem, there exists a standard polynomial-time greedy algorithm that selects $O(k \ln n)$ sets from $\mathscr{C}$ that covers at least as many elements of U as covered by the optimal k sets (see Section 35.3 of [59]). We use this greedy algorithm on $(U, \mathscr{C})$ to obtain a collection $\mathscr{C}' \subset \mathscr{C}$ of $O(k \ln n)$ sets that covers at least $|U| - m$ elements of U. Let F' be the set of facility locations corresponding to $\mathscr{C}'$. Let C' be the clients covered by F'. Then, every client in C' has an assignment cost of at most OPT to one of the facilities in F'. Therefore, F' is a $(1, O(\ln n))$ bi-criteria approximate solution to the problem and $Z' = C \setminus C'$ is the set of at most m outlier points.

The only remaining task is to guess the value OPT of the optimal solution. Since there are at most $|L| \cdot |C|$ possible distances between a client and facility, we execute the greedy algorithm for every possibility. We choose the smallest distance for which the greedy algorithm outputs at most $O(k \ln n)$ sets and covers at least $|U| - m$ elements of U. The overall running time of the algorithm is polynomial in n. It can further be improved by performing a binary search on the $|L| \cdot |C|$ possible distances. More precisely, the reduction to the max k-coverage instance and the greedy algorithm both take $O(n^2)$ time. After that, $O(\log n)$ multiplicative factor due to binary search gives the overall time of $O(n^2 \log n)$ for the algorithm. This completes the proof of the lemma. □

The above algorithm also applies to the outlier k-center problem since the outlier k-center problem is a special case of the outlier k-supplier problem. Next, we convert the bi-criteria approximation algorithm to a list outlier k-supplier/center algorithm.

2.4.2 Conversion: *bi-criteria approximation to list outlier k-supplier algorithm*

We prove the following lemma.

Lemma 3. *Let $\mathcal{I} = (L, C, k, d, z, m)$ be any instance of the outlier k-supplier problem. Let S be any $(1, O(\ln n))$ bi-criteria approximate solution of $\mathcal{I}$ and let Z be the corresponding set of at most m outliers. Let $g(x)$ denote a facility location in L that is closest to an outlier $x \in Z$. Let $G = \{g(x) \mid x \in Z\}$ be the set of such facilities. Let C' be any subset of C of size at least $|C| - m$. Then, for any arbitrary partitioning $\mathbb{O} = \{O_1, \ldots, O_k\}$ of C', there exists a k-sized subset S' of $S \cup G$ that gives 3^z approximation for the cost of $\mathbb{O}$. That is,*

$$\Psi(S', \mathbb{O}) \leq 3^z \cdot \Psi^*(\mathbb{O}).$$

Proof. Suppose that for every client in C, the closest facility location in $S \cup G$ is given by a function $h \colon C \to S \cup G$. Let $F_{\mathbb{O}} = \{f_1, \ldots, f_k\}$ denote the optimal facility set of $\mathbb{O}$ such that partition O_i is assigned to center f_i. Let x_i be any client in O_i. First, we observe that the cost of assigning x_i to facility $h(x_i)$ is at most $\Psi^*(\mathbb{O})$. That is, $d(x_i, h(x_i))^z \leq \Psi^*(\mathbb{O})$. To prove this statement, consider the case when $x_i \in Z$. Then $h(x_i) = g(x_i)$. In other words, $h(x_i)$ is the facility location in L that is closest to x_i. Therefore, $d(x_i, h(x_i))^z \leq d(x_i, f_i)^z \leq \Psi^*(\mathbb{O})$. On the other hand, if $x_i \in C \setminus Z$, then the assignment cost $d(x_i, h(x_i))^z$ is at most OPT since S is a $(1, O(\ln n))$ bi-criteria approximate solution of $\mathcal{I}$. And, therefore, $d(x_i, h(x_i))^z \leq \text{OPT} \leq \Psi^*(\mathbb{O})$.

Now, we show that the set $S' \coloneqq \{h(x_1), \ldots, h(x_k)\}$ is a 3^z approximation to the clustering cost of $\mathbb{O}$. That is, $\Psi(S', \mathbb{O}) \leq 3^z \cdot \Psi^*(\mathbb{O})$. The proof follows from the following sequence of inequalities:

$$\Psi(S', \mathbb{O}) \leq \max_{i=1}^{k} \Big\{ \text{supplier-cost}(h(x_i), O_i) \Big\}$$

$$
\begin{aligned}
&\leq \max_{i=1}^{k} \left\{ \max_{x \in O_i} \left\{ d(h(x_i), x)^z \right\} \right\} \\
&\leq \max_{i=1}^{k} \left\{ \max_{x \in O_i} \left\{ \Big(d(x, f_i) + d(f_i, x_i) + d(x_i, h(x_i)) \Big)^z \right\} \right\} \\
&\qquad \text{(using triangle inequality)} \\
&\leq \max_{i=1}^{k} \left\{ \max_{x \in O_i} \left\{ \Big(d(x, f_i) + d(f_i, x_i) + \Psi^*(\mathbb{O})^{1/z} \Big)^z \right\} \right\} \\
&\leq \max_{i=1}^{k} \left\{ \max_{x \in O_i} \left\{ \Big(2 \cdot \Psi^*(\mathbb{O})^{1/z} + \Psi^*(\mathbb{O})^{1/z} \Big)^z \right\} \right\} \\
&\leq \max_{i=1}^{k} \left\{ \max_{x \in O_i} \left\{ \Big(3 \cdot \Psi^*(\mathbb{O})^{1/z} \Big)^z \right\} \right\} \\
&= 3^z \cdot \Psi^*(\mathbb{O})
\end{aligned}
$$

This completes the proof of the lemma. □

We show a similar result for the outlier k-center problem with an improved approximation guarantee, as follows:

Lemma 4. *Let $\mathcal{I} = (C, C, k, d, z, m)$ be any instance of the outlier k-center problem. Let S be any $(1, O(\ln n))$ bi-criteria approximate solution of $\mathcal{I}$ and let Z be the set of at most m outliers. Let C' be any subset of C of size at least $|C| - m$. Then, for any arbitrary partitioning $\mathbb{O} = \{O_1, \ldots, O_k\}$ of C', there exists a k-sized subset S' of $S \cup Z$ that gives 2^z approximation for the cost of $\mathbb{O}$. That is,*

$$\Psi(S', \mathbb{O}) \leq 2^z \cdot \Psi^*(\mathbb{O})$$

Proof. Suppose that for every client in C, the closest facility location in $S \cup Z$ is given by a function $h \colon C \to S \cup Z$. Let $F_{\mathbb{O}} = \{f_1, \ldots, f_k\} \subseteq C$ denote the optimal facility set of $\mathbb{O}$ such that partition O_i is assigned to facility f_i. Note that f_i is also a client location. Now, observe that the cost of assigning a client f_i to facility location $h(f_i)$ is at most $\Psi^*(\mathbb{O})$. That is, $d(f_i, h(f_i))^z \leq \Psi^*(\mathbb{O})$. To prove this statement, consider the case when $f_i \in Z$, then $h(f_i) = f_i$. Therefore, $d(f_i, h(f_i))^z = 0 \leq \Psi^*(\mathbb{O})$. On the other hand, if $f_i \in C \setminus Z$, then the

assignment cost $d(f_i, h(f_i))^z$ is at most the optimal cost OPT since S is a $(1, O(\ln n))$ bi-criteria approximate solution of $\mathcal{I}$. And, therefore, $d(f_i, h(f_i))^z \leq \text{OPT} \leq \Psi^*(\mathbb{O})$.

Now, we show that the set $S' \coloneqq \{h(f_1), \ldots, h(f_k)\}$ is a 2^z approximation to the clustering cost of $\mathbb{O}$. That is, $\Psi(S', \mathbb{O}) \leq 2^z \cdot \Psi^*(\mathbb{O})$. The proof follows from the following sequence of inequalities:

$$
\begin{aligned}
\Psi(S', \mathbb{O}) &\leq \max_{i=1}^{k} \Big\{ \text{supplier-cost}(h(f_i), O_i) \Big\} \\
&\leq \max_{i=1}^{k} \Big\{ \max_{x \in O_i} \big\{ d(h(f_i), x)^z \big\} \Big\} \\
&\leq \max_{i=1}^{k} \Big\{ \max_{x \in O_i} \Big\{ \Big(d(x, f_i) + d(f_i, h(f_i)) \Big)^z \Big\} \Big\}, \quad \text{(using triangle inequality)} \\
&\leq \max_{i=1}^{k} \Big\{ \max_{x \in O_i} \Big\{ \Big(d(x, f_i) + \Psi^*(\mathbb{O})^{1/z} \Big)^z \Big\} \Big\} \\
&\leq \max_{i=1}^{k} \Big\{ \max_{x \in O_i} \Big\{ \Big(\Psi^*(\mathbb{O})^{1/z} + \Psi^*(\mathbb{O})^{1/z} \Big)^z \Big\} \Big\} \\
&= 2^z \cdot \Psi^*(\mathbb{O})
\end{aligned}
$$

This completes the proof of the lemma. □

The above two lemmas give the following corollary for the list outlier k-supplier/center problem.

Corollary 5 (Main Result)**.** *Given any instance $\mathcal{I} = (L, C, k, d, z, m)$ of the outlier k-supplier problem. There is an algorithm that outputs a list $\mathcal{L}$ of k-center-sets such that for any arbitrary partitioning $\mathbb{O}$ of any subset C' of C of size at least $|C| - m$, there is a center set F in the list that gives 3^z approximation for the clustering cost of $\mathbb{O}$. That is,*

$$\Psi(F, \mathbb{O}) \leq \alpha \cdot \Psi^*(\mathbb{O}), \quad \textit{for } \alpha = 3^z$$

For the outlier k-center instance $\mathcal{I} = (C, C, k, d, z, m)$, the approximation factor is $\alpha = 2^z$. Moreover, the size of the list is at most $(k + m)^{O(k)} \cdot n$ and the running time of the algorithm is

$(k+m)^{O(k)} \cdot n + O(n^2 \log n)$*, which is* FPT *in* k *and* m*.*

Proof. By Lemma 3, we have that there is a k-sized subset $S' \subseteq S \cup G$ that gives 3^z approximation for the clustering $\mathbb{O}$. The size of the set $S \cup G$ is at most $m + O(k \ln n)$. We create the list $\mathcal{L}$ by adding to it all possible k-sized subsets of $S \cup G$. Therefore, $|\mathcal{L}| = (m + O(k \ln n))^k$. If $m \leq k \ln n$, then $|\mathcal{L}| = k^{O(k)} \cdot (\ln n)^k$. Further, using the inequality that $(\ln n)^k \leq k^{O(k)} \cdot n$, (for proof, see Hint 3.18 from the book: Parameterized Algorithms [63]) we get $|\mathcal{L}| \leq k^{O(k)} \cdot n$. On the other hand, if $m > k \ln n$, then $|\mathcal{L}| \leq m^{O(k)}$. This proves that $|\mathcal{L}| \leq (m+k)^{O(k)} \cdot n$ for the list outlier k-supplier problem.

Moreover, the sets S and Z can be computed in time $O(n^2 \log n)$ using Lemma 2. And, the set G can be computed from the set Z in time $O(m \cdot n) = O(n^2)$. Thus, the overall running time of the algorithm is $(k+m)^{O(k)} \cdot n + O(n^2 \log n)$. A similar proof for the k-center version follows from Lemma 4. This proves the corollary. □

2.5 Partition Algorithms

In this section, we design FPT time partition algorithms for all the problems mentioned in Table 2.1 along with their outlier version. For the first six problems in the table, we define a new hybrid problem that encapsulates all the constraints of these problems. Therefore, instead of designing a partition algorithm for each problem separately, we design a partition algorithm for the hybrid problem.

2.5.1 Partition Algorithms: *r-Gather, r-Capacity, Balanced, Chromatic, Fault-Tolerant, and Strongly-Private k-Service Problems*

Formally, the problem is defined as follows:

Definition 25 (Hybrid k-Supplier)**.** *Given an instance $\mathcal{I} = (L, C, k, d, z, m)$ of the outlier k-supplier problem, a partitioning of the client set C into ω color classes: $C_1, \ldots, C_\omega$, vectors:*

$\ell, r \in \mathbb{Z}^k$ and $\alpha, \beta \in \mathbb{Z}^\omega$, find a subset $Z \subseteq C$ of at most m outliers and a partitioning $\mathbb{O} = \{O_1, \ldots, O_k\}$ of the set $C \setminus Z$ with minimum $\Psi^*(\mathbb{O})$ such that it satisfies that $\ell_i \leq |O_i| \leq r_i$ and $\alpha_j \leq |O_i \cap C_j| \leq \beta_j$ for every $i \in [k]$ and $j \in [\omega]$.

The first six problems given in Table 2.1 along with their outlier versions are the special cases of the hybrid k-supplier problem. Furthermore, the hybrid k-supplier problem satisfies the definition of the constrained outlier k-supplier problem, i.e., Definition 20. For the sake of completeness, we describe how the hybrid k-supplier problem encapsulates the first six problems in Table 2.1:

1. For every $i \in [k]$ and $j \in [\omega]$, if $r_i = |C|$, $\alpha_j = 0$, and $\beta_j = |C|$, then the hybrid k-supplier problem is equivalent to the r-gather outlier k-supplier problem.

2. For every $i \in [k]$ and $j \in [\omega]$, if $\ell_i = 0$, $\alpha_j = 0$, and $\beta_j = |C|$, then the hybrid k-supplier problem is equivalent to the r-capacity outlier k-supplier problem.

3. For every $j \in [\omega]$, if $\alpha_j = 0$ and $\beta_j = |C|$, then the hybrid k-supplier problem is equivalent to the balanced outlier k-supplier problem.

4. For every $i \in [k]$ and $j \in [\omega]$, if $\ell_i = 0$, $r_i = |C|$, $\alpha_j = 0$, and $\beta_j = 1$, then the hybrid k-supplier problem is equivalent to the chromatic outlier k-supplier problem.

5. The fault-tolerant k-supplier problem can be reduced to the chromatic k-supplier problem as described in Section 2.1.1. Therefore, the fault-tolerant outlier k-supplier problem is also a special case of the hybrid k-supplier problem.

6. For every $i \in [k]$ and $j \in [\omega]$, if $\ell_i = 0$, $r_i = |C|$, and $\beta_j = |C|$, then the hybrid k-supplier problem is equivalent to the strongly private outlier k-supplier problem.

Now, we give a partition algorithm for the hybrid k-supplier problem. The partition algorithm generalizes the algorithms given in [72].

Lemma 5. *There is a $k^{O(k)} \cdot n^{O(1)}$ time partition algorithm for the hybrid k-supplier problem.*

Proof. Let F be a given set of k facility locations. Let Z^* be an optimal outlier set, and let $\mathbb{O}^* = \{O_1^*, \ldots, O_k^*\}$ be an optimal partitioning of $C \setminus Z^*$ that minimizes the objective function $\Psi(F, \mathbb{O})$. Every partition O_i^* is assigned to one of the facility locations in F. Since the algorithm does not know O_i^*, it guesses that O_i^* is assigned to a facility $f \in F$. Since there are k facilities in F, there are total k^k possibilities of assigning every O_i to facilities in F. The algorithm also makes a guess on the value of the optimal solution $\Psi(F, \mathbb{O}^*)$ over $|L| \cdot |C|$ possible distances between clients in C and facility locations in L. For each particular guess, the algorithm checks if there is some feasible assignment of clients to F satisfying the hybrid constraints. Then, the algorithm outputs the feasible assignment with the minimum assignment cost over all the possible guesses. Now, suppose that in a particular guess, O_i^* is assigned to a facility $f_i \in F$ and the optimal cost is λ. The algorithm reduces the assignment problem to a circulation problem on a flow network $G = (V, E)$. The flow network is shown in Figure 2.1.

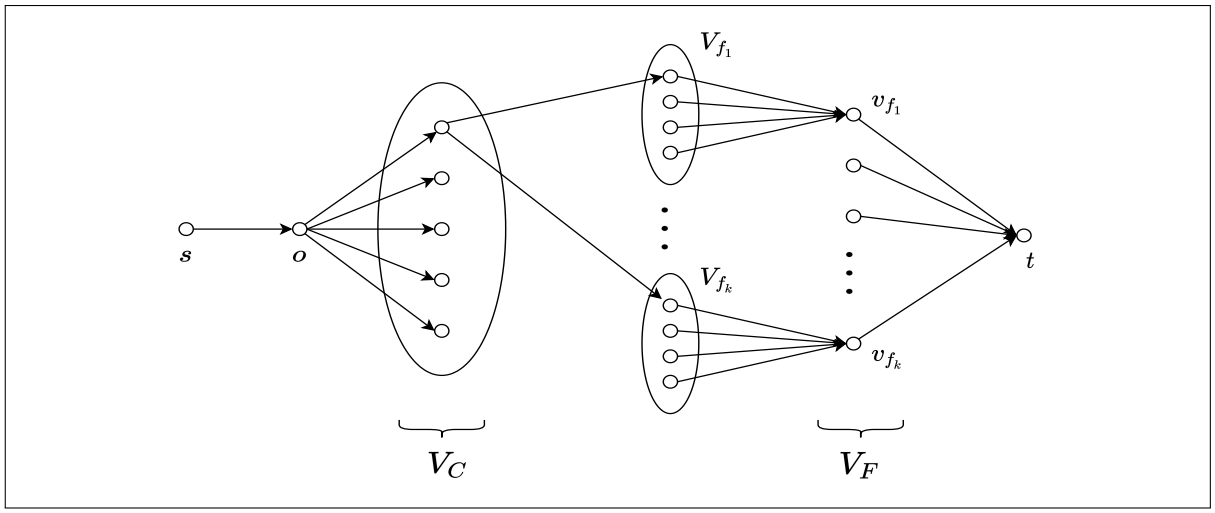

Figure 2.1: The flow network $G = (V, E)$ that is used in the partition algorithm of the hybrid k-supplier problem.

The vertex set V is partitioned into the following sets:

1. A source vertex s, sink vertex t, and vertex o.

2. A vertex set V_C corresponding to the client set C. In other words, for every client $x \in C$ there is a vertex v_x in V_C.

3. A vertex set V_F that corresponds to the facility set F. In other words, for every facility $f_i \in F$ there is a vertex $v_{f_i} \in V_F$.

4. The vertex sets: $V_{f_1}, \ldots, V_{f_k}$. A vertex set V_{f_i} is defined as $\{v_{f_i,1}, \ldots, v_{f_i,\omega}\}$ such that a vertex $v_{f_i,j}$ corresponds to a facility $f_i \in F$ and a color class $j \in [\omega]$.

Furthermore, the edge set E is defined with the lower and upper bound flow constraints as follows:

1. There is a directed edge (s, o) with a lower bound flow of $|C| - m$ and an upper bound flow of $|C|$. This edge ensures that at least $|C| - m$ clients are assigned to F.

2. For every vertex $v_x \in V_C$, there is a directed edge (o, v_x) with upper bound flow of 1 and lower bound flow of 0. These edges ensure that a client is assigned at most one facility in F.

3. For every vertex $v_x \in V_C$ and vertex $v_{f_i,j} \in V_{f_i}$, there is a directed edge $(v_x, v_{f_i,j})$ if and only if the client x belongs to color class C_j and $d(x, f_i)^z \leq \lambda$. These edges have the upper bound flow of 1 and the lower bound flow of 0. These edges ensure a feasible assignment of cost at most λ.

4. For every $i \in [k]$ and $j \in [\omega]$, there is a directed edge $(v_{f_i,j}, v_{f_i})$ with lower bound flow of α_j and upper bound flow of β_j. These edges ensures that for every color class $j \in [\omega]$ and every facility $f_i \in F$, the constraint $\alpha_j \leq |O_i^* \cap C_j| \leq \beta_j$ is satisfied.

5. For every $i \in [k]$, there is a directed edge (v_{f_i}, t) with lower bound flow ℓ_i and upper bound flow r_i. These edges ensures that for every partition $O_i^* \in \{O_1^*, \ldots, O_k^*\}$, the constraint $\ell_i \leq |O_i^*| \leq r_i$ is satisfied.

It is easy to see that a feasible integral flow through the network G corresponds to an assignment of at least $|C| - m$ clients in C to F that satisfies the hybrid constraints. Moreover, the assignment cost is at most λ. The circulation problem on G can be solved in polynomial time using the algorithms in [76, 132, 23]. Since we run this algorithm for $k^k \cdot |C| \cdot |L|$ possible guesses, the overall running time of the partition algorithm is $k^k \cdot n^{O(1)}$. This completes the proof of the lemma. □

In the next subsection, we design the partition algorithms for the remaining two problems in Table 2.1.

2.5.2 Partition Algorithms: *ℓ-Diversity and Fair Outlier k-Supplier Problems*

Bandyapadhyay *et al.* [21] noted that the ℓ-diversity clustering problem is a special case of the fair clustering problem. In other words, if the fair clustering problem has disjoint color classes, and $\alpha_j = 1/\ell$ and $\beta_j = 0$ for every color class $C_j \in \{C_1, \ldots, C_\omega\}$, then the problem is equivalent to the ℓ-diversity clustering problem. Therefore, designing a partition algorithm for the fair outlier k-supplier problem is sufficient.

For the fair k-*median* problem (without outliers), Bandyapadhyay *et al.* [21] designed an FPT time partition algorithm. We simply extend their algorithm to the fair k-*supplier* problem with outliers. Let $\mathcal{I} = (L, C, k, d, z, m, C_1, \ldots, C_\omega, \alpha, \beta)$ be any instance of the fair outlier k-supplier problem. Recall that the sets $C_1, \ldots, C_\omega$ are the color classes, each of which is a subset of the client set C. Moreover, any two color classes can overlap with each other. In other words, a client in C might belong to a different color class. To simplify the problem,

we partition the client set into Γ disjoint groups: $P_1, \ldots, P_\Gamma$ such that the points belonging to the same group P_i belong to the same set of colored classes. In other words, if clients x and y belong to the same group P_i, then for every color class $C_t \in \{C_1, \ldots, C_\omega\}$, $x \in C_t$ if and only if $y \in C_t$. Also, if x and y belong to different groups P_i and P_j, respectively, then there exists a color class $C_t \in \{C_1, \ldots, C_\omega\}$ such that $x \in C_t$ and $y \notin C_t$, or $x \notin C_t$ and $y \in C_t$. Note that if the color classes are pair-wise disjoint, then Γ equals the number of color classes, i.e., $\Gamma = \omega$. Now, we design FPT time partition algorithm for the fair outlier k-supplier problem with running time $(k\Gamma)^{k\Gamma} \cdot n^{O(1)}$, where Γ is the number of distinct collection of color classes induced by the colors of clients. Formally, we state the result as follows:

Lemma 6. *For the fair outlier k-supplier problem, there is a $(k\Gamma)^{k\Gamma} \cdot n^{O(1)}$ time partition algorithm, where Γ is the number of distinct collections of color classes induced by the colors of clients.*

Proof. Let $F = \{f_1, \ldots, f_k\}$ be the given set of facility locations for which we want an optimal partitioning satisfying the fair outlier constraints. Let Z denote the optimal set of outliers and OPT denote the optimal k-supplier cost of assigning $C \setminus Z$ to F while satisfying the fair constraints. Since there are $|F| \cdot |C|$ possible distances between C and F, the algorithm tries each possibility for OPT. For each possibility, the algorithm finds a feasible assignment of clients to facilities. Then, the algorithm outputs that assignment that gives the minimum assignment cost. The assignment problem can be modeled as an integer linear program; however, solving an integer linear program is NP-hard in general. Therefore, we model the problem as a mixed integer linear program, which can be solved in FPT time parameterized by the number of integer variables. The following is a formal statement for the same:

Proposition 1 (Proposition 8.1 of [21])**.** *Given a real-valued matrix $A \in \mathbb{R}^{m \times d}$, vector $b \in \mathbb{R}^m$, vector $c \in \mathbb{R}^d$, and a positive integer $p \leq d$. There is an* FPT *time algorithm that finds a vector $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ that minimizes $c \cdot x$, and satisfies that $A \cdot x \leq b$ and $x_1, \ldots, x_p \in \mathbb{Z}$.*

The running time of the algorithm is $O(p^{2.5p+o(p)}d^4B)$, and the space complexity is polynomial in B, where B is the bit size of the given instance.

Now, we model the assignment problem as a mixed integer linear program (MILP), as follows:

Constraints:	$\sum_{f\in F} g_{x,f} \leq 1$	for every client $x \in C$
$\sum_{x\in P_i} g_{x,f} = h_{f,i}$		for every group $P_i \in \{P_1, \ldots, P_\Gamma\}$ and facility $f \in F$
$\sum_{f\in F, x\in C} g_{x,f} \geq \lvert C\rvert - m$		for client set C and facility set F
$\sum_{x\in C_j} g_{x,f} \leq \alpha_j \cdot \sum_{x\in C} g_{x,f}$		for every facility $f \in F$ and color class $C_j \in \{C_1, \ldots, C_\omega\}$
$\sum_{x\in C_j} g_{x,f} \geq \beta_j \cdot \sum_{x\in C} g_{x,f}$		for every facility $f \in F$ and color class $C_j \in \{C_1, \ldots, C_\omega\}$
$0 \leq g_{x,f} \leq 1$		for every client $x \in C$ and facility $f \in F$
$h_{f,i} \in \mathbb{Z}_{\geq 0}$		for every group $P_i \in \{P_1, \ldots, P_\Gamma\}$ and facility $f \in F$

In the above MILP, for every client $x \in C$ and facility $f \in F$, there is a fractional variable $g_{x,f} \leq 1$ that denotes the fraction of client x assigned to facility f. Moreover, for every group $P_i \in \{P_1, \ldots, P_\Gamma\}$ and facility $f \in F$, there is an integer variable $h_{f,i}$ that denotes the total fraction of clients in P_i that is assigned to facility f. In other words, $h_{f,i} = \sum_{x\in P_i} g_{x,f}$. The third constraint of the MILP corresponds to the number of outliers being at most m. Lastly, the fourth and fifth constraints of MILP correspond to the fairness constraints.

We solve the above mixed integer linear program in time $O(k\Gamma)^{k\Gamma} \cdot n^{O(1)}$ as per Proposition 1. However, the obtained optimal solution contains fractional $g_{x,f}$ values. Next, we show that there

always exists a solution with integral $g_{x,f}$ values that can be obtained in polynomial time. We reduce the problem to the *circulation* problem on directed graphs. We construct a flow network $G = (V, E)$ with upper and lower bound flow requirements on every edge. The construction is shown in Figure 2.2.

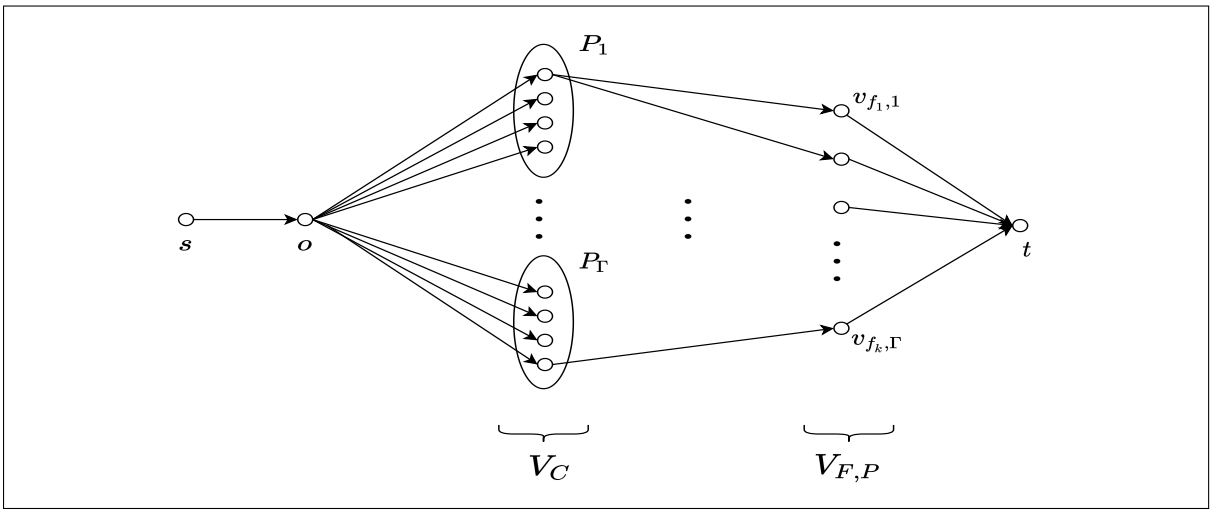

Figure 2.2: The flow network $G = (V, E)$ that is used by the partition algorithm of the fair outlier k-supplier problem.

The vertex set V contains a source vertex s, a sink vertex t, and an *outlier regulating vertex* o. We will describe the functioning of o shortly. The rest of the vertex set is partitioned into two sets: V_C and $V_{F,P}$. The vertex set V_C corresponds to the client set C. In other words, for every client $x \in C$ there is a vertex $v_x \in C$. The vertex set $V_{F,P}$ corresponds to facility set F and groups $P_1, \ldots, P_\Gamma$. In other words, for every facility $f \in F$ and group $P_i \in \{P_1, \ldots, P_\Gamma\}$, there is a vertex $v_{f,i}$ in $V_{F,P}$. Furthermore, the edge set E is partitioned as follows. There is an edge (s, o) with lower and upper bound flow requirement of exactly $\sum_{x \in C, f \in F} g_{x,f}$. Note that $\sum_{x \in C, f \in F} g_{x,f}$ is an integer since $\sum_{x \in C, f \in F} g_{x,f} = \sum_{i=1}^{\Gamma} \sum_{f \in F} h_{i,f}$ and $h_{i,f}$'s are integers. Also note that $\sum_{x \in C, f \in F} g_{x,f} \geq |C| - m$ as per Constraint 2 of the MILP. Therefore, it ensures that at least $|C| - m$ clients are assigned to F, and that's why we call the vertex o the outlier regulating vertex. Furthermore, the edge set E contains for every vertex $v_x \in C$, an edge (o, v_x) with lower bound flow requirement 0 and upper bound flow requirement 1. These edges ensure that a client is assigned to at most one facility in F. Furthermore, for every vertex $v_x \in V_C$ and

$v_{f,i} \in V_{F,P}$, there is an edge $(v_x, v_{f,i})$ if and only if $x \in P_i$. Each edge has a lower bound flow requirement of 0 and an upper bound flow requirement of 1. Lastly, for every vertex $v_{f,i} \in V_{F,P}$, there is an edge $(v_{f,i}, t)$ with lower and upper bound flow requirement of exactly $h_{f,i}$. These edges ensure that exactly $h_{f,i}$ clients of P_i are assigned to any facility $f \in F$. It is easy to see that the constructed flow network G admit a feasible flow if we send a flow of $g_{x,f}$ through every edge $(v_x, v_{f,i})$. Moreover, this is the maximum flow that we can send through the network since the capacity of edge (s, o) is $\sum_{x \in C, f \in F} g_{x,f}$. Since the lower and upper bound requirements on every edge is an integer, a feasible integral flow exists through the network. We find that flow in polynomial time using the algorithms in [132, 76, 23]. Let the new integral flow through $(v_x, v_{f,i})$ is $g'_{x,f}$. Then, we show that the values $g'_{x,f}$'s also satisfy the MILP constraints. The constraints (1)-(3) of the MILP are trivially satisfied from the flow network. Also note that $\sum_{x \in P_i} g'_{x,f} = h_{f,i}$ due to the flow network. Therefore, the fair constraints (4) and (5) are satisfied since $\sum_{x \in C} g_{x,f} = \sum_{x \in C} g'_{x,f}$ and $\sum_{x \in C_j} g_{x,f} = \sum_{x \in C_j} g'_{x,f}$ for every facility $f \in F$ and every color class $C_j \in \{C_1, \ldots, C_\omega\}$. These two equalities follow from the following two sequences of equalities:

1. $$\sum_{x \in C} g_{x,f} = \sum_{i=1}^{\Gamma} \sum_{x \in P_i} g_{x,f} = \sum_{i=1}^{\Gamma} h_{f,i} = \sum_{x \in C} g'_{x,f}$$

2. $$\sum_{x \in C_j} g_{x,f} = \sum_{i:\ P_i \subseteq C_j} \sum_{x \in P_i} g_{x,f} = \sum_{i:\ P_i \subseteq C_j} h_{f,i} = \sum_{x \in C_j} g'_{x,f}$$

This completes the proof of the lemma. □

2.6 FPT Hardness: k-*Supplier and* k-*Center*

In this section, we prove the FPT hardness of approximation result for the k-supplier and k-center problems. We use a reduction from the *dominating set problem* that is defined as follows:

Definition 26 (Dominating Set Problem)**.** *Given an integer $k > 0$, an undirected graph $G = (V, E)$, determine if there a subset $S \subseteq V$ of k vertices such that every vertex in $V \setminus S$ is adjacent to at least one vertex in S.*

Now, note the following FPT hardness result for the dominating set problem.

Theorem 22 ([74, 42, 128])**.** *For any function $g\colon \mathbb{Z}^+ \to \mathbb{R}^+$, there is no $g(k) \cdot n^{o(k)}$ time algorithm for the dominating set problem assuming ETH, and no $g(k) \cdot n^{O(1)}$ time algorithm assuming $\mathsf{W}[2] \neq \mathsf{FPT}$.*

Furthermore, there exists a parameterized reduction from dominating set problem to the k-center problem (see Theorem 9 in [78])[3]. This straightaway gives the following hardness result for the k-center problem:

Theorem 11. *For any constant $\varepsilon > 0$, $z > 0$, and any function $g\colon \mathbb{Z}^+ \to \mathbb{R}^+$, the k-center problem can not be approximated to factor $(2^z - \varepsilon)$ in time $g(k) \cdot n^{o(k)}$ assuming ETH, and in time $g(k) \cdot n^{O(1)}$ assuming $\mathsf{W}[2] \neq \mathsf{FPT}$.*

This completes the hardness proof for the k-center problem. For the k-supplier problem, we use a reduction from the set coverage problem. The set coverage problem is defined as follows:

Definition 27 (Set Coverage Problem)**.** *Given an integer $k > 0$, a set U, and a collection $\mathscr{C} = \{S_1, \ldots, S_m\}$ of subsets of U, i.e., $S_j \subseteq U$ for every $j \in [m]$, determine if there exist k sets in $\mathscr{C}$ that cover all elements in U.*

Now, note that there exists a trivial reduction from the dominating set problem to the set coverage problem (see [99] for details). Therefore, Theorem 22 implies the following hardness result for the set coverage problem:

[3] In [78], the definition of the k-center problem has $z = 1$.

Theorem 23. *For any function $g\colon \mathbb{Z}^+ \to \mathbb{R}^+$, there is no $g(k) \cdot n^{o(k)}$ time algorithm for the set coverage problem assuming ETH, and no $g(k) \cdot n^{O(1)}$ time algorithm assuming* $\mathsf{W}[2] \neq \mathsf{FPT}$.

Furthermore, a parameterized reduction exists from the set coverage problem to the k-supplier problem. The reduction is similar to the reduction from the *hitting set problem* (see Theorem 6 of [94]). For the sake of completeness, we describe the reduction here: given a set coverage instance $(U, \mathscr{C}, k)$, we construct a k-supplier instance (C, L, d, k) as follows. For every set $S_i \in \mathscr{C}$, we define a center $f_i \in L$. For every element $e \in U$, we define a client $x_e \in C$. Let us define the distance function $d(.,.)$ as follows. For any two points $x_e, x_{e'} \in C$, or $f_i, f_j \in L$, the distance $d(x_e, x_{e'}) = d(f_i, f_j) = 2$. For any point $x_e \in C$ and $f_i \in L$, if $e \notin S_i$, the distance $d(x_e, f_i) = 3$; otherwise $d(x_e, f_i) = 1$. Furthermore, assume that $d(.,.)$ is a symmetric function, i.e., $d(x, y) = d(y, x)$ for every $x, y \in C \cup L$. Also assume that $d(x, x) \geq 0$ for every $x \in C \cup L$. It is easy to see that $d(.,.)$ satisfies all the properties of a metric space.

Now, suppose that there exist k sets: $S_{i_1}, \ldots, S_{i_k}$ in $\mathscr{C}$ that cover all elements of U, i.e., $S_{i_1} \cup \ldots \cup S_{i_k} = U$, then the center set $F = \{f_{i_1}, \ldots, f_{i_k}\}$ gives the k-supplier cost 1. On the other hand, if there does not exist any k sets in $\mathscr{C}$ that could cover all elements of U, then for any center set $F \subseteq L$ of size k there would exist a point $x \in C$ at a distance of 3 from F, i.e., $d(F, x) = 3$. Therefore, the k-supplier cost would be 3^z. Since the set coverage problem is $\mathsf{W}[2]$-hard, it implies that the k-supplier problem can not be approximated to any factor better than 3^z, in polynomial time, assuming $\mathsf{W}[2] \neq \mathsf{FPT}$. This reduction together with Theorem 23 give the following hardness result for the k-supplier problem:

Theorem 10. *For any constant $\varepsilon > 0$, $z > 0$, and any function $g\colon \mathbb{Z}^+ \to \mathbb{R}^+$, the k-supplier problem cannot be approximated to factor $(3^z - \varepsilon)$ in time $g(k) \cdot n^{o(k)}$ assuming ETH, and in time $g(k) \cdot n^{O(1)}$ assuming* $\mathsf{W}[2] \neq \mathsf{FPT}$.

This completes the hardness proof for the k-supplier problem.

2.7 FPT Hardness: *Outlier k-Supplier and k-Center*

In this subsection, we establish FPT hardness of approximation results for the outlier k-supplier and k-center problems. The proofs are trivial. For the outlier k-supplier problem, we obtain the following result:

Theorem 24. *For any constant $\varepsilon > 0$, $z > 0$, and any function $g\colon \mathbb{Z}^+ \times \mathbb{Z}^{\geq 0} \to \mathbb{R}^+$, the outlier k-supplier problem can not be approximated to factor $(3^z - \varepsilon)$ in time $g(k,m) \cdot n^{m+o(k)}$, assuming ETH.*

Proof. For the sake of contradiction, assume that for some constant $\varepsilon > 0$, $z > 0$, and function $g\colon \mathbb{Z}^+ \times \mathbb{Z}^{\geq 0} \to \mathbb{R}^+$ the outlier k-supplier problem can be approximated to factor $(3^z - \varepsilon)$ in time $g(k,m) \cdot n^{m+o(k)}$. For $m = 0$, the problem is the classical (non-outlier) k-supplier problem. Therefore, the k-supplier problem can be approximated to factor $(3^z - \varepsilon)$ in time $g(k,0) \cdot n^{o(k)} = h(k) \cdot n^{o(k)}$ for some function $h\colon \mathbb{Z}^+ \to \mathbb{R}^+$. It contradicts Theorem 10. Thus, it proves the theorem. □

Similarly, we obtain the following FPT hardness of approximation result for the outlier k-center problem:

Theorem 25. *For any constant $\varepsilon > 0$, $z > 0$, and any function $g\colon \mathbb{Z}^+ \times \mathbb{Z}^{\geq 0} \to \mathbb{R}^+$, the outlier k-center problem can not be approximated to factor $(2^z - \varepsilon)$ in time $g(k,m) \cdot n^{m+o(k)}$, assuming ETH.*

Proof. For the sake of contradiction, assume that for some constant $\varepsilon > 0$, $z > 0$, and function $g\colon \mathbb{Z}^+ \times \mathbb{Z}^{\geq 0} \to \mathbb{R}^+$ the outlier k-center problem can be approximated to factor $(2^z - \varepsilon)$ in time $g(k,m) \cdot n^{m+o(k)}$. For $m = 0$, the problem is the classical (non-outlier) k-center problem. Therefore, the k-center problem can be approximated to factor $(2^z - \varepsilon)$ in time $g(k,0) \cdot n^{o(k)} = h(k) \cdot n^{o(k)}$ for some function $h\colon \mathbb{Z}^+ \to \mathbb{R}^+$. It contradicts Theorem 11. Thus, it proves the theorem. □

Note that the above hardness results apply to all the outlier versions of constrained k-supplier problems that we study in this paper since the unconstrained outlier version of the problem can be reduced to any of the constrained versions in polynomial time.

Also note that the above two results do not eliminate the possibility of having the polynomial time 3^z and 2^z approximation algorithms for the outlier k-supplier and k-center problems, respectively. In fact, there exists polynomial time 3^z and 2^z approximation algorithms for the outlier k-supplier [40] and k-center problem [37], respectively. Moreover, we can even obtain the optimal solutions to the outlier k-supplier and k-center problems using a trivial $O(n^{k+1}k)$ running time algorithm.

Chapter 3

FPT Approximation for Constrained k-Median/Means

In this chapter, we study a range of *constrained* versions of the k-*median* and k-*means* problem. In the classical (*unconstrained*) k-median problem, we are given a set of clients C in a metric space $\mathcal{X}$, with distance function $d(.,.)$. We are also given a set of feasible facility locations $L \subseteq \mathcal{X}$. The goal is to open a set F of k facilities in L to minimize the sum of distances of clients to the closest open facility, i.e., minimize, cost$(F, C) \equiv \sum_{j \in C} \{d(F, j)\}$, where $d(F, j)$ is the distance of client j to the closest facility in F. The k-means problem is defined similarly using squared distances (i.e., $d^2(.,.)$ instead of $d(.,.)$). In many applications, there are additional constraints imposed on the clusters. For example, to balance the load among the facilities in resource allocation problems, a capacity u is imposed on every cluster. In other words, no more than u clients can be assigned to any facility/cluster. This problem is known as the *capacitated* k-means/k-median problem. Likewise, various other applications have different constraints, which give rise to different *constrained* versions of the problem. Surprisingly, no constant-approximation algorithm is known for many of the constrained problems. Moreover, the *unconstrained* problem itself is known to be $\mathsf{W}[2]$-hard when parameterized by k. We work

within the unified framework of Ding and Xu [72] that allows us to simultaneously obtain FPT algorithms for a wide range of constrained clustering problems, namely: *r-gather, r-capacity, balanced, chromatic, fault-tolerant, ordered-weighted-average, strongly private, ℓ-diversity, fair, and uncertain k-median/means problems, with and without outliers.* In particular, here are some of the main highlights of this work:

1. We give $(3+\varepsilon)$ and $(9+\varepsilon)$-approximation algorithms for the constrained k-median and k-means problems, respectively, with FPT running time $(k/\varepsilon)^{O(k)} \cdot n^{O(1)}$, where $n = |C \cup L|$.

2. We also study the problems when $C \subseteq L$, i.e., a facility can be opened at a client location. For this special case, we design algorithms that give $(2+\varepsilon)$ and $(4+\varepsilon)$-approximation guarantees for the constrained k-median and k-means problems, respectively, with FPT running time $(k/\varepsilon)^{O(k)} \cdot n^{O(1)}$, where $n = |L|$. Note that the case $C \subseteq L$ subsumes the case $C = L$. Therefore, this result also holds for the case when $C = L$.

3. We also study the constrained k-median/means problem with outliers. Our algorithms give $(3+\varepsilon)$ and $(9+\varepsilon)$-approximation guarantees for the constrained *outlier* k-median and k-means problems, respectively, with FPT running time $((k+m)/\varepsilon)^{O(k)} \cdot n^{O(1)}$, where $n = |C \cup L|$ and m is the number of outliers. For the special case, when $C \subseteq L$, the algorithms give $(2+\varepsilon)$ and $(4+\varepsilon)$-approximation guarantees for the constrained outlier k-median and k-means problems, respectively.

4. Since our algorithms are based on a simple sampling-based approach; we also obtain constant-pass log-space streaming algorithms for the constrained k-median/means problems with and without outliers.

5. We show that the analysis of our algorithm is tight. That is, there are instances for which our algorithm does not provide better than $(3-\delta)$ and $(9-\delta)$ approximation guarantee corresponding to k-median and k-means objectives, respectively, for arbitrarily small

constant $\delta > 0$. Similarly, the analysis of our algorithm is tight for the special case $C \subseteq L$.

3.1 Overview

The k-means and k-median problems are similar. We combine the discussion of these problems by defining the k-service problem that encapsulates both these problems.

Definition 28 (k-Service Problem)**.** *Let $(\mathcal{X}, d)$ be a metric space. Let k be any positive integer, and z be any positive real number. Given a set $L \subseteq \mathcal{X}$ of feasible facility locations, and a set $C \subseteq \mathcal{X}$ of clients, find a set $F \subseteq L$ of k facilities that minimises the cost:* service-cost$(F, C) \equiv \sum_{x \in C} \left\{ \min_{f \in F} \left\{ d(f, x)^z \right\} \right\}$.

For $z = 1$ the problem is known as the *k-median* problem, and for $z = 2$ the problem is known as the *k-means* problem. The above definition is motivated by the *facility location* problem [126] and differs from it in two ways. First, in the facility location problem, one is allowed to open any number of facilities. Second, one has to pay for an additional facility establishment cost for every open facility. Thus the k-service problem is equivalent to the facility location problem for a fixed number of facilities and 0 facility establishment costs.

The k-service problem can also be viewed as a clustering problem, where the goal is to group the objects that are similar to each other. The clustering algorithms are commonly used in data mining, pattern recognition, and information retrieval [96]. Note that the clients assigned to the same facility belong to the same cluster, and the corresponding facility is known as their *cluster center*. Keeping this in mind, we will use the terms *facility* and *center* interchangeably from now on.

The classical (unconstrained) k-means and k-median problems do not entirely capture the desired clustering properties for many real-world applications. For example, consider the popular *k-anonymity* principle [131]. The principle provides anonymity to a public database while

keeping it meaningful simultaneously. One way to achieve this is to cluster the data in such a way as to release only partial information related to the clusters obtained. Further, to protect the data from the *re-identification* attacks, the clustering should be done so that each cluster gets at least r data points. This method is popularly known as r-*gather* clustering [5]. Similarly, we have r-*capacity* clustering problem where in addition to minimizing the clustering cost, we have a constraint that no cluster can contain more than r clients [111, 2]. Likewise, there are many other constrained versions of the k-median/means problems namely *fault-tolerant* [92], *fair* [26], *uncertain* [60], ℓ-*diversity* [72], etc.

An important distinction between the constrained and unconstrained clustering version is the *locality* property. In simple words, the *locality* property says that the points close to each other should be part of the same cluster. This property holds for the unconstrained version of the problem. However, this may not necessarily hold for many of the constrained versions of the problem where minimizing clustering cost is not the *only* requirement. To understand this, consider a center-set $F = \{f_1, f_2, \ldots, f_k\}$ and let $\mathbb{O} = \{O_1, \ldots, O_k\}$ denote the clustering of the dataset such that each client is assigned to its closest center in F. Note that the clustering $\mathbb{O}$ minimizes the distance-based cost function but may not satisfy any additional constraints on the clusters. For example, for the r-capacity problem, a cluster O_i might contain more than r clients. In a constrained setting, we may need an algorithm that given a center-set $\{f_1, ..., f_k\}$ as input, outputs a clustering $\mathbb{O} = \{O_1, \ldots, O_k\}$ which in addition to minimising $\sum_i \sum_{x \in O_i} d(x, f_i)^z$ also satisfies the given constraints on the clusters. Such an algorithm is called a *partition algorithm*. In the unconstrained setting, the partition algorithm simply assigns clients to their closest centers in F. However, designing efficient partition algorithms for the constrained versions of the problem is non-trivial. Ding and Xu [72] designed a partition algorithm for the r-capacity problem and a range of other constrained clustering problems. Later, we will see that the partition algorithms are crucial in designing FPT algorithms for constrained clustering problems.

A partition algorithm allows us to go from a center set to an optimal clustering. What about

the reverse direction? Given a clustering $\mathbb{O} = \{O_1, O_2, \ldots, O_k\}$, can we find a center set that gives minimum clustering cost? The solution to this problem is simple. Construct a complete weighted bipartite graph $G = (V_l, V_r, E)$, where a vertex in V_l corresponds to a facility location in L, and a vertex in V_r corresponds to a cluster $O_j \in \mathbb{O}$. The weight on an edge $(i, j) \in V_l \times V_r$ is equal to the cost of assigning the cluster O_j to the i^{th} facility, i.e., $\sum_{x \in O_j} d(x, i)^z$. We can easily obtain an optimal assignment by finding a minimum cost perfect matching in graph G. For a given set of k distinct facilities $F = \{f_1, ..., f_k\}$ and a clustering $\mathbb{O} = \{O_1, \ldots, O_k\}$, we define optimal assignment cost by the cost function:

$$\Phi(F, \mathbb{O}) \equiv \min_{\text{permutation } \pi} \left\{ \sum_{i=1}^{k} \sum_{x \in O_i} d(x, f_{\pi(i)})^z \right\}. \tag{3.1}$$

In other words, $\Phi(F, \mathbb{O})$ is the cost of a minimum cost perfect matching in graph G. From now on, we will call any set of k distinct facilities $\{f_1, \ldots, f_k\}$ as a *k-center-set*.

Note that we are considering the *hard* assignment version of the clustering problems. That is, we are not allowed to open more than one facility at any location in L. This version differs from the *soft* assignment version, where more than one facility can be opened at any location in L. Note that the total number of open facilities in both versions is at most k. The soft assignment version is easier than the hard assignment version since the soft assignment version can be reduced to the hard assignment version by creating k copies of every location in L. Moreover, it has been observed that the soft-assignment version allows us to obtain better approximation guarantees than the hard-assignment version [61, 111]. For our discussion, we will call a center-set a *soft center-set* if it contains a facility location multiple times. Otherwise, we call it a *hard center-set*. In fact, a soft center set is a multi-set. We will avoid using the term multi-set to keep our discussion simple.

In the past, the constrained versions of the clustering problems were studied separately as independent problems. More recently, in 2015, Ding and Xu [72] gave a unified framework

for these problems that they called the *constrained clustering* framework. They proposed this unified framework in the context of the k-median and k-means problems in the continuous Euclidean spaces where $L = \mathbb{R}^p$ and C is a finite subset of $\mathbb{R}^p$. In this work, we extend the framework to arbitrary discrete metric spaces.

3.1.1 Constrained k-service framework

We define the constrained k-service problem in discrete metric space as follows:

Definition 29 (Constrained k-service problem)**.** *Let $(\mathcal{X}, d)$ be a metric space, k be any positive integer, and z be any positive real number. Given a set $L \subseteq \mathcal{X}$ of feasible facility locations, a set $C \subseteq \mathcal{X}$ of clients, and a set $\mathbb{S}$ of feasible partitionings of C, find a partitioning $\mathbb{O} = \{O_1, O_2, \ldots, O_k\}$ in $\mathbb{S}$, that minimizes the cost function: $\Phi^*(\mathbb{O}) \equiv \min_{\text{k-center-set } F} \Phi(F, \mathbb{O})$.*

The key component of the above definition is the set of feasible clusterings $\mathbb{S}$. Using it, we can define any constrained version of the k-service problem. Note that $\mathbb{S}$ could contain an exponential number of distinct partitionings, i.e., $|\mathbb{S}| \geq \exp(n, k)$. However, for many constrained clustering problems, the set $\mathbb{S}$ can be defined concisely using a simple set of mathematical constraints. For example, the set $\mathbb{S}$ for the r-gather problem can be defined as $\mathbb{S} \coloneqq \{\mathbb{O} \mid \text{for every partition } O_i \in \mathbb{O}, |O_i| \geq r_i\}$, where $\mathbb{O} = \{O_1, O_2, \ldots, O_k\}$ is a partitioning of the client set. We study nine other constrained clustering problems: r-capacity, balanced, chromatic, fault-tolerant, ordered-weighted-average, strongly private, ℓ-diversity, fair, and uncertain k-service problems. The definitions of these problems are given in Table 3.1. Note that the problems: fault-tolerant, ordered-weighted-average, and uncertain k-service problems do not satisfy Definition 29 of the constrained k-service problem since their objective functions are different than $\Phi^*(\mathbb{O})$. However, each of these problems can be reduced to a different problem that satisfies Definition 29 of the constrained k-service problem. Ding and Xu [72] showed these reductions for the fault-tolerant and uncertain k-service problems. For the ordered-weighted-average k-service problem, Byrka *et al.* [36] showed a reduction to the

fault-tolerant k-service problem. Thus all three problems indirectly satisfy the definition of the constrained k-service problem.

#	**Problem**	**Description**
1.	r-Gather k-service problem	Given k positive integers: $r_1, \ldots, r_k$, find clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Phi^*(\mathbb{O})$ such that for all i, $\lvert O_i\rvert \geq r_i$
2.	r-Capacity k-service problem	Given k positive integers: $r_1, \ldots, r_k$, find clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Phi^*(\mathbb{O})$ such that for all i, $\lvert O_i\rvert \leq r_i$
3.	Balanced k-service problem	Given positive integers: $\ell_1, \ldots, \ell_k$, and $r_1, \ldots, r_k$, find clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Phi^*(\mathbb{O})$ such that for all i, $\ell_i \leq \lvert O_i\rvert \leq r_i$
4.	Chromatic k-service problem	Given that every client has an associated color, find a clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Phi^*(\mathbb{O})$ such that for all i, O_i should not have any two points with the same color.
5.	Fault-tolerant k-service problem	Given positive integer $\ell_x \leq k$ for every client $x \in C$, find a set F of k centers, such that the sum of service cost of the clients to ℓ_x of nearest centers out of $F = \{f_1, f_2, \ldots, f_k\}$, is minimised.
6.	Ordered-Weighted-Average k-service problem	Given a vector $(w_1, ..., w_k)$ of non-increasing weights, find a center set $\{f_1, ..., f_k\}$ such that $\sum_{x\in C}\sum_{j=1}^{k} w_j \cdot \big(\overline{d}_j(x)\big)^z$ is minimised. Here, $\big(\overline{d}_1(x), ..., \overline{d}_k(x)\big)$ is a non-decreasing ordering of $\big(d(x, f_1), ..., d(x, f_k)\big)$.
7.	Strongly private k-service problem	Given a partitioning $C_1, \ldots, C_\omega$ of the client set C, and a set of integers: $\{\ell_1, \ldots, \ell_\omega\}$, find a clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Phi^*(\mathbb{O})$ that satisfies $\lvert C_j \cap O_i\rvert \geq \ell_j$ for every $i \in [k]$ and $j \in [\omega]$.
8.	ℓ-Diversity k-service problem	Given a partitioning $C_1, \ldots, C_\omega$ of the client set C, a real number $\ell > 1$, find a clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Phi^*(\mathbb{O})$ such that the fraction of points belonging to the same partition inside O_i is $\leq 1/\ell$.
9.	Fair k-service problem	Given ω color classes $C_1, \ldots, C_\omega$ (not necessarily disjoint), such that every C_j is a subset of the client set C, and two fairness vectors $\alpha, \beta \in [0, 1]^\omega$, find a clustering $\mathbb{O} = \{O_1, \ldots, O_k\}$ with minimum $\Phi^*(\mathbb{O})$ such that it satisfies that $\beta_j \cdot \lvert O_i\rvert \leq \lvert O_i \cap C_j\rvert \leq \alpha_j \cdot \lvert O_i\rvert$ for every $i \in [k]$ and $j \in [\omega]$.
10.	Uncertain k-service problem	Given a discrete probability distribution for every client, i.e., for a client $x \in C$ there is a set $D_x = \{x_1, \ldots, x_h\}$ such that x takes the value x_i with probability t_x^i and $\sum_{i=1}^{h} t_x^i \leq 1$. Find a clustering $\mathbb{O} = \{O_1, ..., O_k\}$ and facility set $F = \{f_1, \ldots, f_k\}$ such that $\sum_{x\in C} \min_{f_j\in F} d(x, f_j)^z$ is minimized, where $d(x, f_j)^z = \sum_{i=1}^{h} t_x^i \cdot d(x_i, f_j)^z$.

Table 3.1: List of constrained k-service problems with FPT time partition algorithms (see Section 3.9).

It has been observed in the past works [72, 28] that any constrained version of k-median/means

can be solved using a partition algorithm and a solution to a very general "list" version of the clustering problem which we discuss next [1].

Definition 30 (List k-Service Problem)**.** *Let $\mathcal{I} = (L, C, k, d, z)$ be an arbitrary instance of the k-service problem, $\mathbb{O} = \{O_1, \ldots, O_k\}$ be an arbitrary clustering of the client set C, and $0 < \varepsilon \leq 1$ be an arbitrary constant. The goal of the problem is to find a list $\mathcal{L}$ of k-center-sets (i.e., each element of the list is a set of k elements from L) such that with probability at least $1 - 1/n$, the list $\mathcal{L}$ contains a k-center-set F such that $\Phi(F, \mathbb{O}) \leq \alpha \cdot \Phi^*(\mathbb{O})$ for $\alpha = 3^z + \varepsilon$ and $n = |C \cup L|$. For the special case when $C \subseteq L$, the approximation guarantee is $\alpha = 2^z + \varepsilon$.*

Definition 31 (Partition Algorithm)**.** *Let $\mathcal{I} = (L, C, k, d, z)$ be any instance of the k-service problem, and let $\mathbb{S} = \{\mathbb{O}_1, \mathbb{O}_2, \ldots, \mathbb{O}_t\}$ be a collection of clusterings of C. Given a center set $F \subseteq L$, a partition algorithm outputs a clustering in $\mathbb{S}$ that has the least clustering cost $\Phi(F, \mathbb{O})$ with respect to F.*

Note that the set $\mathbb{S}$ differs for different constrained clustering problems; therefore, the partition algorithm differs for various constrained clustering problems. However, the algorithm for the list k-service problem is the same for every constrained clustering problem. Suppose we have an algorithm for the list k-service problem and a partition algorithm for a particular constrained k-service problem; then, we can obtain an approximation algorithm for that constrained k-service problem. The following theorem proves this result:

Theorem 26. *Let $\mathcal{I} = (L, C, k, d, z, \mathbb{S})$ be any instance of the constrained k-service problem, and let $A_{\mathbb{S}}$ be a partition algorithm for $\mathbb{S}$ with running time T_A. Let B be any algorithm for the list k-service problem with running time T_B. Then, there is an algorithm that, with probability at least $1 - 1/n$, outputs a clustering $\mathbb{O} \in \mathbb{S}$ that is an α-approximate solution for the constrained k-service instance $\mathcal{I}$. Moreover, the running time of the algorithm is $O(T_B + |\mathcal{L}| \cdot T_A)$.*

[1] This notion of list version of the clustering problem was implicitly present in the work of Ding and Xu [72]. Bhattacharya et al. [28] formalized this as the *list k-means problem*.

Proof. The algorithm is simple. We first run algorithm B to obtain a list $\mathcal{L}$. For every k-center-set in the list, the algorithm runs the partition algorithm $A_{\mathbb{S}}$ on it. Then the algorithm outputs a center set that gives the minimum clustering cost. Let F' be this k-center-set and $\mathbb{O}'$ be the corresponding clustering. We claim that $(F', \mathbb{O}')$ is an α-approximation for the constrained k-service problem with probability at least $1 - 1/n$.

Let $\mathbb{O}^*$ be an optimal solution for the constrained k-service instance $(L, C, k, d, z, \mathbb{S})$ and F^* denote the corresponding k-center-set. By the definition of the list k-service problem, with probability at least $1 - 1/n$, there is a k-center-set F in the list $\mathcal{L}$, such that $\Phi(F, \mathbb{O}^*) \leq \alpha \cdot \Phi(F^*, \mathbb{O}^*)$. Let $\mathbb{O} = A_{\mathbb{S}}(F) \in \mathbb{S}$ be the optimal clustering corresponding to F. Thus, $\Phi(F, \mathbb{O}) \leq \alpha \cdot \Phi(F^*, \mathbb{O}^*)$. Since $(F', \mathbb{O}')$ gives the minimum cost clustering in the list, we have $\Phi(F', \mathbb{O}') \leq \Phi(F, \mathbb{O})$. Therefore, $\Phi(F', \mathbb{O}') \leq \alpha \cdot \Phi(F^*, \mathbb{O}^*)$.

The running time analysis is also simple. T_B is the time to obtain the list $\mathcal{L}$. Then, the algorithm runs the partition procedure $A_{\mathbb{S}}$ for every center set in the list; the running time of this step is $|\mathcal{L}| \cdot T_A$. Picking a minimum cost clustering from the list takes $O(|\mathcal{L}|)$ time. Hence the overall running time is $O(T_B + |\mathcal{L}| \cdot T_A)$. □

The goal is to design an algorithm for the list k-service problem and the partition algorithms for different constrained clustering problems. In Section 3.5, we design an algorithm for the list k-service problem with FPT running time. Formally, we state the result as follows:

Theorem 27. *Let $\mathcal{I} = (L, C, k, d, z, \mathbb{O}, \varepsilon)$ be any instance of the list k-service problem. There is an algorithm that outputs a list $\mathcal{L}$ of size at most $O\left((\log n) \cdot (k/\varepsilon)^{O(kz^2)}\right)$, where $n = |C \cup L|$. Moreover, the running time of the algorithm is $O\left(n \cdot (k/\varepsilon)^{O(kz^2)}\right)$, which is FPT in k.*

Using Theorems 26 and 27, we further obtain the following two corollaries:

Corollary 6 (Main Result: k-Median)**.** *For any constrained version of the k-median problem that has a partition algorithm with running time T, there exists a $(3 + \varepsilon)$-approximation algo-*

rithm that succeeds with probability at least $1-1/n$ *and has running time* $T\cdot(k/\varepsilon)^{O(k)}\cdot(\log n)+O\left(n\cdot(k/\varepsilon)^{O(k)}\right)$. *For the special case when* $C\subseteq L$, *the algorithm gives* $(2+\varepsilon)$*-approximation guarantee.*

Corollary 7 (Main Result: k-Means)**.** *For any constrained version of the* k*-means problem that has a partition algorithm with running time* T, *there exists a* $(9+\varepsilon)$*-approximation algorithm that succeeds with probability at least* $1-1/n$ *and has running time* $T\cdot(k/\varepsilon)^{O(k)}\cdot(\log n)+O\left(n\cdot(k/\varepsilon)^{O(k)}\right)$. *For the special case when* $C\subseteq L$, *the algorithm gives* $(4+\varepsilon)$*-approximation guarantee.*

The remaining task is to design partition algorithms for different constrained clustering problems. For all the problems mentioned in Table 3.1, we design their partition algorithms in Section 3.9. To summarize, all the problems in Table 3.1 except the ℓ-diversity and fair k-service problems, the running time of the partition algorithms is $k^k\cdot n^{O(1)}$. Most of these partition algorithms were first designed by Ding and Xu [72]. For the ℓ-diversity k-service problem, the running time of the partition algorithm is $(\omega k)^{\omega k}\cdot n^{O(1)}$ [21]. For the fair k-service problem, the running time of the partition algorithm is $(k\Gamma)^{O(k\Gamma)}\cdot n^{O(1)}$, where Γ is the number of distinct collection of color classes induced by the colors of clients [21]. We describe the notation Γ in more detail in Section 3.9.6. Using an algorithm for the list k-service problem and partition algorithms, we obtain FPT time $(3+\varepsilon)$ and $(9+\varepsilon)$ approximation guarantees for these problems with respect to k-median and k-means objectives, respectively. Formally, we state these results as follows:

Theorem 28. *There is an* FPT *time* $(3^z+\varepsilon)$*-approximation algorithm for the following constrained versions of the* k*-service problem:*

1. *r-gather k-service problem*
2. *r-capacity k-service problem*
3. *Balanced k-service problem*
4. *Chromatic k-service problem*
5. *Fault-tolerant k-service problem*
6. *Ordered-Weighted-Average k-service problem*
7. *Strongly private k-service problem*
8. *Uncertain k-service problem*

The running time of the algorithm is $(k/\varepsilon)^{O(k\,z^2)} \cdot n^{O(1)}$. For the special case when $C \subseteq L$, the algorithm gives $(2^z + \varepsilon)$-approximation guarantee.

Theorem 29. *There is an* FPT *time $(3^z + \varepsilon)$-approximation algorithm for the ℓ-diversity k-service problem with running time $(k\omega/\varepsilon)^{O\left(k\cdot\left(\omega+z^2\right)\right)} \cdot n^{O(1)}$. For the special case when $C \subseteq L$, the algorithm gives $(2^z + \varepsilon)$-approximation guarantee.*

Theorem 30. *There is an* FPT *time $(3^z + \varepsilon)$-approximation algorithm for the fair k-service problem with running time $(k\Gamma/\varepsilon)^{O\left(k\cdot\left(\Gamma+z^2\right)\right)} \cdot n^{O(1)}$, where Γ denote the number of distinct collection of color classes induced by the colors of clients. Moreover, if the color classes are pair-wise disjoint, then $\Gamma = \omega$, and the running time of the algorithm is $(k\omega/\varepsilon)^{O\left(k\cdot\left(\omega+z^2\right)\right)} \cdot n^{O(1)}$. For the special case when $C \subseteq L$, the algorithm gives $(2^z+\varepsilon)$-approximation guarantee.*

We extend the constrained clustering framework to the outlier setting in the next subsection. The discussion will be analogous to the above discussion.

3.1.2 Constrained k-service framework with outliers

In practical scenarios, it often happens that a few clients are located at faraway locations from the majority of the clients. These clients are called *outliers*. Outliers force the algorithm to open the facilities close to the outliers. Due to this, the majority of the clients have to pay high assignment costs. This leads to poor clustering of the dataset. To overcome this issue, we cluster the dataset without the outliers. This gives rise to the *outlier k-service* problem. A mathematical formulation of the problem was first proposed by Charikar *et al.* [40]. The following is the definition of the outlier k-service problem:

Definition 32 (Outlier k-Service)**.** *Let $(\mathcal{X}, d)$ be a metric space. Let k and m be any positive integers, and z be any positive real number. Given a set $L \subseteq \mathcal{X}$ of feasible facility locations, and a set $C \subseteq \mathcal{X}$ of clients, find a subset $Z \subseteq C$ of size at most m clients and a set $F \subseteq L$*

of k facilities such that the k-service cost of $C' \coloneqq C \setminus Z$ is minimized: service-cost$(F, C') \equiv \sum_{x \in C'} \left\{ \min_{f \in F} \left\{ d(x,f)^z \right\} \right\}$

Charikar *et al.* [40] designed a bi-criteria approximation algorithm for the problem that gives an $O(1)$-approximation guarantee while violating the number of outliers by a factor of $(1+\varepsilon)$. Chen [43] gave the first constant factor approximation algorithm for the outlier k-*median* problem. Recently, Krishnaswamy *et al.* [106] designed the $(7.081+\varepsilon)$ and $(53.002+\varepsilon)$-approximation algorithms for the outlier k-median and k-means problems, respectively. Friggstad *et al.* [81] gave a bi-criteria approximation algorithm for the outlier k-means problem that gives a $(25+\varepsilon)$-approximation guarantee and opens $(1+\varepsilon)$ facilities. Recently, Feng *et al.* [80] gave a $(6+\varepsilon)$-approximation algorithm for the outlier k-means problem ($C \subseteq L = \mathcal{X}$), with FPT running time of $O\left(n \cdot \beta^k \left(\frac{k+m}{\varepsilon}\right)^k\right)$, for some constant $\beta > 0$. In this work, we improve this result by giving a $(4+\varepsilon)$-approximation algorithm for the outlier k-means problem and a $(2+\varepsilon)$-approximation algorithm for the outlier k-median problem when $C \subseteq L = \mathcal{X}$ with FPT running time $O\left(n \cdot \left(\frac{k+m}{\varepsilon}\right)^{O(k)}\right)$. For the general case (when C is not necessarily a subset of L), our algorithm gives a $(9+\varepsilon)$-approximation guarantee for the outlier k-means problem and $(3+\varepsilon)$-approximation guarantee for the outlier k-median problem with FPT running time $O\left(n \cdot \left(\frac{k+m}{\varepsilon}\right)^{O(k)}\right)$. Furthermore, we extend the algorithm to the constrained versions of the outlier k-service problem. The constrained outlier k-service problem is defined as follows:

Definition 33 (Constrained Outlier k-Service Problem)**.** *Let (L, C, k, d, z, m) be any instance of the outlier k-service problem and $\mathbb{S}$ be any collection of partitionings such that every partitioning $\mathbb{O} \in \mathbb{S}$ is a partitioning of some subset $C' \subseteq C$ of size at least $|C| - m$. Find a clustering $\mathbb{O} = \{O_1, O_2, \ldots, O_k\}$ in $\mathbb{S}$, that minimizes the objective function: $\Phi^*(\mathbb{O}) \equiv \min\limits_{k\text{-center-set } F} \Phi(F, \mathbb{O})$.*

Furthermore, we define the *list outlier k-service problem* and *outlier partition algorithm*, as follows:

Definition 34 (List Outlier k-Service Problem)**.** *Let $\mathcal{I} = (L, C, k, d, z, m)$ be an arbitrary*

instance of the outlier k-service problem, $\mathbb{O} = \{O_1, \ldots, O_k\}$ be an arbitrary partitioning of the some subset $C' \subseteq C$ of size at least $|C| - m$, and $0 < \varepsilon \leq 1$ be an arbitrary constant. The goal of the problem is to find a list $\mathcal{L}$ of k-center-sets (i.e., each element of the list is a set of k elements from L) such that with probability at least $1 - 1/n$, the list $\mathcal{L}$ contains a k-center-set F such that $\Phi(F, \mathbb{O}) \leq \alpha \cdot \Phi^*(\mathbb{O})$ for $\alpha = 3^z + \varepsilon$ and $n = |C \cup L|$. For the special case when $C \subseteq L$, the approximation guarantee $\alpha = 2^z + \varepsilon$.

Definition 35 (Outlier Partition Algorithm)**.** *Let $\mathcal{I} = (L, C, k, d, z, m)$ be an instance of the outlier k-service problem, and let $\mathbb{S} = \{\mathbb{O}_1, \mathbb{O}_2, \ldots, \mathbb{O}_t\}$ be a collection of clusterings such that each $\mathbb{O}_i$ is a clustering of of some subset of $C' \subseteq C$ of size at least $|C| - m$. Given a center set $F \subseteq L$, an outlier partition algorithm outputs a clustering in $\mathbb{S}$ that has the least clustering cost $\Phi(F, \mathbb{O})$ with respect to F.*

Suppose we have an algorithm for the list outlier k-service problem and a partition algorithm for a specific constrained outlier k-service problem. We can obtain an approximation algorithm for that constrained outlier k-service problem. The following theorem state this result and is analogous to Theorem 26 of the non-outlier version.

Theorem 31. *Let $\mathcal{I} = (L, C, k, d, z, m, \mathbb{S})$ be any instance of the constrained outlier k-service problem, and let $A_{\mathbb{S}}$ be an outlier partition algorithm for $\mathbb{S}$ with running time T_A. Let B be any algorithm for the list outlier k-service problem with running time T_B. Then, there is an algorithm that, with probability at least $1 - 1/n$, outputs a clustering $\mathbb{O} \in \mathbb{S}$ that is an α-approximate solution for the constrained outlier k-service instance $\mathcal{I}$. Moreover, the running time of the algorithm is $O(T_B + |\mathcal{L}| \cdot T_A)$.*

Proof. The proof is analogous to the proof of Theorem 26. □

The goal now becomes to design an algorithm for the list outlier k-service problem and outlier partition algorithms for different constrained versions of the outlier k-service problems. In

Section 3.5, we design an algorithm for the list outlier k-service problem with FPT running time parameterized by m and k. Formally, we state the result as follows:

Theorem 32. *Let $\mathcal{I} = (L, C, k, d, z, m, \mathbb{O}, \varepsilon)$ be any instance of the list outlier k-service problem. There is an algorithm that outputs a list $\mathcal{L}$ of size at most $O\left((\log n) \cdot ((k+m)/\varepsilon)^{O\left(k\,z^2\right)}\right)$. Moreover, the running time of the algorithm is $O\left(n \cdot ((k+m)/\varepsilon)^{O\left(k\,z^2\right)}\right)$, which is FPT in k and m.*

Using Theorems 31 and 32, we obtain the following two main results:

Corollary 8 (Main Result: Outlier k-Median)**.** *For any constrained version of the outlier k-median problem that has a partition algorithm with running time T, there exists a $(3+\varepsilon)$-approximation algorithm that succeeds with probability at least $1 - 1/n$ and has running time $T \cdot ((k+m)/\varepsilon)^{O(k)} \cdot (\log n) + O\left(n \cdot ((k+m)/\varepsilon)^{O(k)}\right)$. For the special case when $C \subseteq L$, the algorithm gives $(2+\varepsilon)$-approximation guarantee.*

Corollary 9 (Main Result: Outlier k-Means)**.** *For any constrained version of the outlier k-means problem that has a partition algorithm with running time T, there exists a $(9+\varepsilon)$-approximation algorithm that succeeds with probability at least $1 - 1/n$ and has running time $T \cdot ((k+m)/\varepsilon)^{O(k)} \cdot (\log n) + O\left(n \cdot ((k+m)/\varepsilon)^{O(k)}\right)$. For the special case when $C \subseteq L$, the algorithm gives $(4+\varepsilon)$-approximation guarantee.*

We consider the outlier versions of all the problems described in Table 3.1. In Section 3.9, we design FPT time partition algorithms for the outlier versions of all these problems. Thus, we get FPT time $(3+\varepsilon)$ and $(9+\varepsilon)$ approximation algorithms for the outlier versions of these problems with respect to the k-median and k-means objectives, respectively. Formally, we state the results as follows:

Theorem 33. *There is an FPT time $(3^z + \varepsilon)$-approximation algorithm for the outlier versions of the following constrained k-service problems:*

1. r-gather k-service problem *2. r-capacity k-service problem*

3. Balanced k-service problem *4. Chromatic k-service problem*

5. Fault-tolerant k-service problem *6. Ordered-Weighted-Average k-service problem*

7. Strongly private k-service problem *8. Uncertain k-service problem*

The running time of the algorithm is $((k+m)/\varepsilon)^{O(k\,z^2)} \cdot n^{O(1)}$. For the special case when $C \subseteq L$, the algorithm gives $(2^z+\varepsilon)$-approximation guarantee.

Theorem 34. *There is an* FPT *time $(3^z+\varepsilon)$-approximation algorithm for the outlier version of the ℓ-diversity k-service problem with running time $((k+m)\omega/\varepsilon)^{O\left(k\cdot\left(\omega+z^2\right)\right)} \cdot n^{O(1)}$. For the special case when $C \subseteq L$, the algorithm gives $(2^z+\varepsilon)$-approximation guarantee.*

Theorem 35. *There is an* FPT *time $(3^z+\varepsilon)$-approximation algorithm for the outlier version of the fair k-service problem with running time $((k+m)\Gamma/\varepsilon)^{O\left(k\cdot\left(\Gamma+z^2\right)\right)} \cdot n^{O(1)}$, where Γ denote the number of distinct collection of color classes induced by the colors of clients. Moreover, if the color classes are pair-wise disjoint, then $\Gamma=\omega$, and the running time of the algorithm is $((k+m)\omega/\varepsilon)^{O\left(k\cdot\left(\omega+z^2\right)\right)} \cdot n^{O(1)}$. For the special case when $C \subseteq L$, the algorithm gives $(2^z+\varepsilon)$-approximation guarantee.*

This completes the discussion on the constrained outlier k-service problem. Next, we convert our algorithm to a streaming algorithm. We require a streaming version of the list outlier k-service algorithm and a streaming version of the partition algorithm. In Section 3.8, we design a constant-pass log-space streaming algorithm for the list outlier k-service problem. In Section 3.9, we design streaming partition algorithms for the outlier versions of some of the problems given in Table 3.1. Although the *single*-pass streaming algorithms are considered useful, it is interesting to know that there are constant-pass streaming algorithms for many outlier versions of the constrained k-service problem.

3.2 Related Work

A unified framework for the constrained k-means/k-median problems was introduced by Ding and Xu [72]. Using this framework, they designed $(1+\varepsilon)$-approximation algorithms for various constrained clustering problems with FPT running time parameterized by k. However, their study was limited to the Euclidean space where $L = \mathbb{R}^p$ and C is a finite subset of $\mathbb{R}^p$. They obtained the results using an algorithm for the list version of the k-means problem (even though it was not formally defined in their work). The running time of their algorithm was $O(np \cdot (\log n)^k \cdot 2^{poly(k/\varepsilon)})$ and the list size was $(\log n)^k \cdot 2^{poly(k/\varepsilon)}$. Bhattacharya *et al.* [28] formally defined the list k-service problem. They obtained a faster algorithm for the list k-service problem with running time of $O(np \cdot (k/\varepsilon)^{O(\log(k/\varepsilon))})$ and list size of $(k/\varepsilon)^{O(\log(k/\varepsilon))}$. We use a sampling-based approach similar to the algorithm of Bhattacharya *et al.* [28]. Our work differs from the previous works in the following ways:

1. Working in a metric space instead of the Euclidean space poses challenges as some of the main tools used for analysis in the Euclidean setting cannot be used in metric spaces. We carefully devise and prove new sampling lemmas that makes the high-level analysis of Bhattacharya *et al.* [28] go through.

2. Bhattacharya *et al.* [28] gave an algorithm for the list-k-means problem with list size $|\mathcal{L}| = (\frac{k}{\varepsilon})^{O(\frac{k}{\varepsilon})}$ and running time $O(np|\mathcal{L}|)$. Their algorithm explores a rooted tree of size $(\frac{k}{\varepsilon})^{O(\frac{k}{\varepsilon})}$ and depth k where the degree of every non-leaf vertex is $(\frac{k}{\varepsilon})^{O(\frac{1}{\varepsilon})}$. Every node in this tree has an associated center and the path from root to a leaf node gives one of the k-center-sets for the output list. The algorithm has an unavoidable iteration of depth k since their analysis works only when the centers are picked *one-by-one* in k iterations. We circumvent this inherent restriction by using a constant factor approximate solution F to the unconstrained k-means/median problem for the given dataset (C, L). That is, service-cost$(F, C) \leq \alpha \cdot$ OPT, where OPT denotes the optimal k-means/median cost.

Then the sampling algorithm runs in a *single* iteration where poly$(\frac{k}{\varepsilon})$ points from C are D^z-sampled with respect to F. Thus, we obtain the list $\mathcal{L}$ in a single shot. This technique helps us in designing streaming algorithm for the problem.

3. We study the hard-assignment version of the constrained k-service problem which is harder than the soft-assignment version of the problem. The hard and soft assignment versions are equivalent in the continuous Euclidean space where $L = \mathbb{R}^p$. The reason is that if two facilities are opened at the same facility location, then one of the facilities can be moved by an infinitely small distance to convert a soft-assignment to a hard-assignment. Therefore, the previous works on the constrained clustering problem did not consider the distinction between soft and hard assignment.

4. We extend the constrained k-median/means framework to the outlier setting. We design $((k+m)/\varepsilon)^{O(k)} \cdot n^{O(1)}$ time algorithm for the outlier version of the constrained k-median/means problems in general metric spaces, where $n = |C \cup L|$ and m is the number of outliers.

Ours is the first algorithm for many problems that achieves a constant approximation in FPT running time. Please see Table 3.2 for the known results. Independent of our work, Bandyopadhyay *et al.* [21] designed similar algorithms as ours using coreset techniques. However, they did not study the problem in hard-assignment and outlier settings.

We want to point out that the r-gather, r-capacity, and balanced k-service problems that we study in this paper impose cluster-wise constraints. However, the alternate definitions of the r-gather, r-capacity, and balanced k-service problems impose constraints on individual facility locations. Formally, the balanced k-service problem with location-wise constraints is defined as follows:

Definition 36 (Balanced k-service Problem with Location-Wise Constraints)**.** *Given an instance $\mathcal{I} = (L, C, k, d, z)$ of the k-service problem, a lower bound function $g\colon L \to \mathbb{Z}_+$, and an*

upper bound function $h\colon L \to \mathbb{Z}_+$, find a set $F \subseteq L$ of k facility and assignment $\phi\colon C \to L$ that minimizes the assignment cost $\sum_{j\in C} \min_{i\in F} d(j,i)^z$ and satisfies that $g(f) \leq |\phi^{-1}(f)| \leq h(f)$ for every facility location $f \in F$.

The above definition also encapsulates the r-gather and r-capacity k-service problems with location-wise constraints. For the r-gather problem, $h(f) = |C|$ for every facility location $f \in L$, and for the r-capacity problem, $g(f) = 0$ for every facility location $f \in L$. Moreover, when every facility location has the same values of $g(f)$ and $h(f)$, then the problems are known as the *uniform r-gather, r-capacity, and balanced k-service problems*. It is easy to see that for the uniform version, the problem with location-wise constraints is equivalent to the problem with cluster-wise constraints. In other words, Definition 36 is the same as the definition given in Table 3.1 for the uniform case. To the best of our knowledge, the non-uniform variant of the problem with cluster-wise constraints has not been studied before. Furthermore, we believe that it is non-trivial to obtain any polynomial time reduction between the problems with cluster-wise constraints and location-wise constraints.

As we mentioned earlier, the unconstrained metric k-median problem is hard to approximate within a factor strictly smaller than $(1 + 2/e)$, and the metric k-means problem is hard to approximate within a factor strictly smaller than $(1 + 8/e)$. Surprisingly this lower bound persists even if we allow an FPT running time [53]. The problem also has a matching upper bound algorithm with an FPT running time [53]. Therefore, the unconstrained k-means and k-median problems in the metric setting are relatively well understood. On the other hand, our understanding of most constrained versions of the problem is still far from complete. We believe that our work is important in understanding constrained problems in general metric spaces.

#	Problem	k-Median Objective		k-Means Objective	
		Without Outliers	With Outliers	Without Outliers	With Outliers
1.	r-Gather Clustering (non-uniform version with cluster-wise constraint)	-	-	-	-
	r-Gather Clustering (uniform version)	**7.2** (for C = L) (FPT time) [69]	-	**86.9** (for C = L) (FPT time) [69]	-
2.	r-Capacity Clustering (non-uniform version with cluster-wise constraint)	-	-	-	-
	r-Capacity Clustering (uniform version)	$(\mathbf{3}+\varepsilon)$ [52] (FPT time)	-	$(\mathbf{9}+\varepsilon)$ [52] (FPT time)	-
3.	Balanced Clustering (non-uniform version with cluster-wise constraint)	-	-	-	-
	Balanced Clustering (uniform version)	**7.2** (for C = L) (FPT time) [69]	-	**86.9** (for C = L) (FPT time) [69]	-
4.	Chromatic Clustering	-	-	-	-
5.	Fault Tolerant Clustering (non-uniform version)	**93** [92] (polynomial time)	-	$\boldsymbol{O(1)}$ [95] (polynomial time)	$\boldsymbol{O(\ell)}$ [95] (polynomial time)
	Fault Tolerant Clustering (uniform version)	**4** [130] (polynomial time)	-	-	-
6.	Ordered-Weighted-Average Clustering	**93** [36] (polynomial time)	-	-	-
7.	Strongly Private Clustering	-	-	-	-
8.	ℓ-Diversity Clustering [2]	**(4.675, 1)** [27] (polynomial time)	-	**(62.856, 1)** [27] (polynomial time)	-
9.	Fair Clustering (disjoint color classes)	**(4.675, 1)** [27] (polynomial time)	-	**(62.856, 1)** [27] (polynomial time)	-
	Fair Clustering (overlapping color classes)	$(\mathbf{4.675}, \mathbf{4\Delta+3})$ (polynomial time) [26]	-	$(\boldsymbol{O(1)}, \mathbf{4\Delta+3})$ (polynomial time [26]	-
10.	Uncertain Clustering (assigned version)	$(\mathbf{6.35}+\varepsilon)$ (for $C \subseteq L$) [60] (polynomial time)	-	$(\mathbf{74}+\varepsilon)$ (for $C \subseteq L$) [60] (polynomial time)	-

Table 3.2: The known results for the constrained k-median/means problems with and without outliers. We only mention the results for the *soft assignment* version of the problems; however, some of these results also hold for the *hard assignment* version that we did not mention explicitly for the sake of simplicity. No polynomial time constant factor approximation algorithm is known for the ℓ-diversity and fair clustering problems. The algorithms of [26] and [27] give $O(1)$ approximation guarantees corresponding to the k-median and k-means objective respectively; however, they violate the fairness constraint by an additive factor of $4\Delta + 3$, where Δ denotes the maximum number of color classes a client can be part of. When the color classes are disjoint, the algorithms violate the fairness constraint by an additive factor of 1.

We discuss the known results for each of the problems mentioned in Table 3.1 in more detail:

1. **r-gather k-service problem:** For the *uniform* r-gather *k-median* problem, the algorithms of [129] and [9] can be adapted to obtain an $O(1)$-approximation guarantee [10]. Under the assumption of $C = L$, Hu Ding [69] gave FPT time $(3\lambda+2)$-approximation algorithm for the uniform r-gather *k-median* problem and $18\lambda + 16$ approximation algorithm for the uniform r-gather *k-means* problem. Here, λ denotes the approximation guarantee of any *unconstrained* k-median or k-means algorithm. We can use the FPT algorithm of Addad *et al.* [53], which has $\lambda = 1 + 2/e$ for the unconstrained k-median problem and $\lambda = 1 + 8/e$ for the unconstrained k-means problem. Moreover, these bounds are tight, assuming Gap-ETH. Thus, the algorithm of Hu Ding [69] gives at best 7.2 and 86.9-approximation for the r-gather k-median and k-means problems, respectively.

2. **r-capacity k-service problem:** For the capacitated k-service problem, no constant-factor approximation is known yet, even in the uniform setting. However, various bi-criteria approximation algorithms are known for the problem [112, 111, 34, 67], which either violate the capacity or cardinality constraint (the constraint on the number of open facilities) by a constant factor. The problem has been also studied in FPT time parameterized by k [139, 2, 52]. The best-known algorithm is due to Addad *et al.* [52]; the authors designed the FPT time 3 and 9 approximation algorithms for the non-uniform capacitated k-median and k-means problems, respectively. However, their algorithm holds when the non-uniform capacities on imposed on the locations. On the other hand, our algorithm holds when the non-uniform capacities are imposed on the clusters. Our algorithm gives the same bounds as that of [52] for the uniform capacitated case. Moreover, we extend our algorithm to the outlier and streaming settings.

3. **Balanced k-service problem:** Under the assumption of $C = L$, Hu Ding [69] gave FPT

[2]Bera *et al.* [27] did not explicitly mention the results for the ℓ-diversity clustering problem. However, the results follow from [27] since the ℓ-diversity problem is a special case of fair clustering problem with disjoint color classes as noted by Bandyapadhyay *et al.* [21].

time $(3\lambda + 2)$-approximation algorithm for the uniform balanced k-*median* problem and $18\lambda + 16$ approximation algorithm for the uniform balanced k-*means* problem. Here, λ denotes the approximation guarantee of any *unconstrained* k-median or k-means algorithm. We can use the FPT algorithm of Addad *et al.* [53], which has $\lambda = 1 + 2/e$ for the unconstrained k-median problem and $\lambda = 1 + 8/e$ for the unconstrained k-means problem. Moreover, these bounds are tight, assuming Gap-ETH. Thus, the algorithm of Hu Ding [69] gives at best 7.2 and 86.9-approximation for the balanced k-median and k-means problems, respectively.

4. **Chromatic k-service problem:** The problem was formulated by Ding and Xu [71] and it has certain applications in cell biology [70]. Ding and Xu [72, 71] gave a PTAS for the chromatic k-median and k-means problems in the Euclidean space (i.e., $C \subseteq L = \mathbb{R}^p$) and Bhattacharya *et al.* [28] improved the running time of the algorithm. No constant factor approximation algorithm is known for the problem in the general metric spaces.

5. **Fault-tolerant k-service problem:** In the fault-tolerant k-service problem, given a facility set $F \subseteq L$, the cost of a client $x \in C$ is the sum of the distances to its ℓ_x closest facility locations in F. If ℓ_x is the same for every x in C, then we call the problem the *uniform fault-tolerant k-service problem*. On the other hand, if ℓ_x is not the same for every x, then we call the problem the non-uniform fault-tolerant k-service problem. For the non-uniform fault-tolerant k-*median* problem, the first approximation algorithm was given by Anthony *et al.* [15]. The algorithm had a $O(\log n)$-approximation guarantee. Hajiaghayi *et al.* [92] gave an improved 93-approximation algorithm for the problem. For the *uniform* fault-tolerant k-median problem, a better approximation guarantee of 4 is known due to Swamy and Shmoys [130]. For the *outlier* fault-tolerant k-means problem, Inamdar and Varadarajan [95] gave $O(\ell)$-approximation algorithm.

6. **Ordered-weighted-average (OWA) k-service problem:** The problem was first proposed by Byrka *et al.* [36]. The authors reduced the OWA k-service problem to the

fault-tolerant k-service problem with client multiplicities. The client multiplicity means that each client $j \in C$ could have multiple copies in the instance. This reduction gave a 93-approximation for the OWA k-median problem, using the 93-approximation algorithm of the fault-tolerant k-median problem [92]. However, since the multiplicity could be exponential in $|C|$, a straightforward implementation of the algorithm of Hajiaghayi *et al.* [92] does not run in polynomial time. However, the authors showed that the algorithm of Hajiaghayi *et al.* [92] could be adapted to run in polynomial time for the fault-tolerant k-median problem with client multiplicities.

7. **Strongly private k-service problem:** This problem has recently been proposed by Rösner and Schmidt [127] in the context of the k-center objective. The authors gave 5 and 4 approximation algorithms for the k-supplier and k-center versions, respectively, without outliers. However, no constant factor approximation algorithm is known for the problem for the k-service objective.

8. **ℓ-diversity k-service problem:** Bandyapadhyay *et al.* [21] noted that the ℓ-diversity clustering problem is a special case of the fair clustering problem when the color classes are disjoint, and $\alpha_j = 1/\ell$ and $\beta_j = 0$ for every color class $C_j \in \{C_1, \ldots, C_\omega\}$. Therefore, the problem admit 4.675 and 20.443 approximation algorithms for the k-median and k-means versions, respectively, without outliers [27]; however, the algorithms violate the constraint by an additive factor of 3, i.e., $0 \leq |O_i \cap C_j| \leq |O_i|/\ell + 3$ for every cluster $O_i \in \{O_1, \ldots, O_k\}$ and color class $C_j \in \{C_1, \ldots, C_\omega\}$.

   Another variant of the ℓ-diversity clustering problem was proposed by Li *et al.* [110]. Given an integer constant $\ell \geq 0$, the task is to find a clustering $\mathbb{O} = \{O_1, ..., O_t\}$ of the client set C with minimum $\Phi^*(\mathbb{O})$ such that for every cluster O_i, $|O_i| \geq \ell$ and O_i should not have any two clients with the same color. Note that, unlike other clustering problems, here, we do not have any restriction on the number of open centers.

9. **Fair k-service problem:** In this problem, we are given ω color classes: $C_1, \ldots, C_\omega$

that are subsets of the client set C and two fairness vectors $\alpha, \beta \in [0,1]^{\omega}$. No true constant factor approximation algorithm is known for the problem yet. The existing algorithms give $O(1)$ approximation guarantee for the k-median and k-means objectives, respectively; however, they violate the fairness constraint by an additive factor of $4\Delta + 3$, where Δ denote the maximum number of groups a client can be part of [26]. For the case of disjoint color classes, the fairness constraint is violated by an additive factor of 1 [27].

10. **Unfair k-service problem (Probabilistic Clustering):** The problem was proposed by Cormode and McGregor [60]. The authors defined the input instance in the following manner: A client j in C is represented by a random variable X_j such that j is present at the location $x \in \mathcal{X}$ with probability t_x^j, i.e., $P[X_j = x] = t_x^j$. Certainly, we have $\sum_{x \in \mathcal{X}} t_x^j \leq 1$ for every client $j \in C$. Also, note that the probability could be less than one since it is possible that a client might not exist at all. The cost function is defined in two ways: *unassigned* and *assigned*, as follows.

    (a) *Unassigned Cost:* In this case, we output a center-set F that minimizes the following cost function:

$$k\text{-median:} \quad \sum_{(x_1,x_2,\ldots,x_n)\in\mathcal{X}^n} \left(\prod_{j=1}^{n} \mathbf{Pr}[X_j = x_j] \cdot \sum_{j=1}^{n} d(x_j, F) \right)$$

$$k\text{-means:} \quad \sum_{(x_1,x_2,\ldots,x_n)\in\mathcal{X}^n} \left(\prod_{j=1}^{n} \mathbf{Pr}[X_j = x_j] \cdot \sum_{j=1}^{n} d^2(x_j, F) \right)$$

    Here, $n \coloneqq |C|$, and $d(x, F) \coloneqq \min_{f \in F} \{d(x, f)\}$ denote the distance of x to the closest facility location.

    Let F^* be an optimal center-set corresponding to the above objective function. We assign a client j to a facility location in F based on its realized position. Suppose x_j is the realized position of the client j. Then we assign j to a facility location that is closest to x_j.

(b) *Assigned Cost:* In this case, we assign a client to a cluster center prior to its realization. Therefore, we assume that for a client j, all its realizations are assigned to the same center. The goal is to output a center set F and an assignment $\sigma : C \to F$ that minimizes the following cost function:

$$k\text{-median:} \quad \sum_{(x_1,x_2,\ldots,x_n)\in\mathcal{X}^n} \left(\prod_{j=1}^{n} \mathbf{Pr}[X_j = x_j] \cdot \sum_{j=1}^{n} d(x_j, \sigma(j)) \right)$$

$$k\text{-means:} \quad \sum_{(x_1,x_2,\ldots,x_n)\in\mathcal{X}^n} \left(\prod_{j=1}^{n} \mathbf{Pr}[X_j = x_j] \cdot \sum_{j=1}^{n} d^2(x_j, \sigma(j)) \right)$$

Cormode and McGregor [60] showed that using the linearity of expectation, the problem can be equivalently stated as finding a clustering $\mathbb{O} = \{O_1, ..., O_k\}$ and facility set $F = \{f_1, \ldots, f_k\}$ such that $\sum_{x\in C} \min_{f_j\in F} d(x, f_j)$ is minimized, where $d(x, f_j)^z = \sum_{i=1}^{h} t_x^i \cdot d(x_i, f_j)$.

The *unassigned* version of these problems is quite simple. In this case, both problems can be simply reduced to their weighted unconstrained counterparts by linearity of expectation (see Section 5 of [60]). Thus we get a $(2.675 + \varepsilon)$-approximation for the uncertain k-median problem [35] and $(9 + \varepsilon)$-approximation for the uncertain k-means problem [8]. Therefore, in this work, we only study these problems with respect to their *assigned* objectives.

For the assigned version, Lammersen and Schmidt [108] gave the first coreset construction for the assigned version of the uncertain k-median problem. Cormode and McGregor [60] reduced the assigned version of the uncertain k-service problem to its weighted unconstrained counterpart, with a certain loss in the approximation factor. In particular, they gave a $(2\alpha + 1)$-approximation algorithm for the uncertain k-median problem and a $(8\alpha + 2)$-approximation algorithm for the uncertain k-means problem [3]. Here,

[3]For the uncertain k-*means* objective, Cormode and McGregor [60] did not state the $(8\alpha + 2)$-approximation, explicitly. However, this result can be obtained using the same technique used to obtain the $(2\alpha+1)$-approximation for the uncertain k-median problem. The $(2\alpha + 1)$-approximation algorithm for the uncertain k-median problem

α is the approximation guarantee of any unconstrained k-median/k-means algorithm. The current best approximation guarantee for the unconstrained k-median problem is $(2.675 + \varepsilon)$ [35], and the unconstrained k-means problem is $(9 + \varepsilon)$ [8]. Substituting these α values, we obtain $(6.35 + \varepsilon)$-approximation for the uncertain k-median problem, and $(74 + \varepsilon)$-approximation for the uncertain k-means problem. All the above-stated approximation guarantees assume that $C \subseteq L = \mathcal{X}$.

3.3 Notations and Identities

This section defines the notations and identities that we frequently use in the paper. We define the unconstrained k-service cost of a client set S with respect to a center set F as service-cost$(F, S) \coloneqq \sum_{x \in S} \min_{f \in F} d(f, x)^z$. For a singleton set $\{f\}$, we denote service-cost$(\{f\}, S)$ shortly by service-cost(f, S). For an instance $\mathcal{I} = (L, C, k, d, z)$, we denote the optimal (unconstrained) k-service cost of the instance by OPT(L, C, k).

The following is the binomial approximation inequality that we use to simplify the terms with large exponents.

Fact 1 (Binomial Approximation)**.** *For $\varepsilon \cdot n \leq 1/2$, we have $(1 + \varepsilon)^n \leq (1 + 2\varepsilon n)$*

Proof. We prove using induction on n. For the base case, $n = 0$; therefore, the fact holds trivially. Then, we assume that the induction hypothesis holds for $n = k$. That is $(1 + \varepsilon)^k \leq 1 + 2\varepsilon k$ for $\varepsilon k \leq 1/2$. Then, we show that induction hypothesis holds for $n = k + 1$ for $\varepsilon(k + 1) \leq 1/2$. The proof follows from the following sequence of inequalities:

$$(1 + \varepsilon)^{k+1} \leq (1 + 2k\varepsilon)(1 + \varepsilon)$$

$$(\because \varepsilon k \leq \varepsilon(k + 1) \leq 1/2 \text{ and using induction hypothesis for } n = k)$$

is stated in Theorem 10 of [60]. In this theorem, if we replace the triangle-inequality with *approximate* triangle-inequality for the k-means objective, we would obtain an $(8\alpha + 2)$-approximation guarantee.

$$
\begin{aligned}
&= 1 + 2\varepsilon k + \varepsilon + 2\varepsilon^2 k \\
&\leq 1 + 2\varepsilon k + \varepsilon + \varepsilon, \qquad (\because \varepsilon k \leq 1/2) \\
&= 1 + 2\varepsilon(k+1).
\end{aligned}
$$

Hence proved. □

We use the following fact to carry out the trade-off between two values a and b. We will choose the value of δ according to our requirements.

Fact 2. *For any $\delta, z, a, b > 0$, we have $(a+b)^z \leq (1+\delta)^z \cdot b^z + \left(1 + \frac{1}{\delta}\right)^z \cdot a^z$.*

Proof. There are two possibilities: $a \leq \delta \cdot b$ or $a > \delta \cdot b$. For the first case, we have $(a+b)^z \leq (1+\delta)^z \cdot b^z$. For the second case, we have $(a+b)^z \leq \left(1 + \frac{1}{\delta}\right)^z \cdot a^z$. Hence we get the required result. □

Since we are working on the metric spaces, triangle inequality becomes a powerful tool to prove the required bounds. We need to generalize the triangle inequality since we deal with a general cost function d^z. The following inequality is the generalization of the triangle inequality and simply follows from the *power-mean* inequality.

Fact 3 (Approximate triangle inequality)**.** *For a set of points $\{a, b, c\} \in \mathcal{X}$, $d(a,b)^z \leq 2^{z-1} \cdot \left((d(a,c)^z + d(c,b)^z\right)$. Similarly for a set of four points $\{a, b, c, d\} \in \mathcal{X}$ we have $d(a,b)^z \leq 3^{z-1} \cdot \left((d(a,c)^z + d(c,d)^z + d(d,b)^z\right)$*

In the following lemma, we use *uniform sampling* to obtain a constant approximation for any arbitrary cluster S. A similar version of the lemma has been used in multiple other works to analyze sampling-based algorithms (for example, see Lemma 3.1 in [16]).

Lemma 7. *For a set $S \subseteq C$, let f^* be any center in L. If we uniformly sample a point x in S and open a facility at the closest location in L, then the following identity holds:*

$$\mathbb{E}[\text{service-cost}(t(x), S)] \leq 3^z \cdot \text{service-cost}(f^*, S),$$

where $t(x)$ is the closest facility location from x.

Proof. The proof follows from the following sequence of inequalities.

$$\begin{aligned}
&\mathbb{E}[\text{service-cost}(t(x), S)] \\
&= \frac{1}{|S|}\left(\sum_{x\in S} \text{service-cost}(t(x), S)\right) \\
&= \frac{1}{|S|}\left(\sum_{x\in S}\sum_{x'\in S} d(t(x), x')^z\right) \\
&\leq \frac{3^{z-1}}{|S|}\left(\sum_{x\in S}\sum_{x'\in S} (d(f^*, x')^z + d(x, f^*)^z + d(t(x), x)^z)\right), \quad \text{(using Fact 3)} \\
&\leq \frac{3^{z-1}}{|S|}\left(\sum_{x\in S}\sum_{x'\in S} (d(f^*, x')^z + d(x, f^*)^z + d(f^*, x)^z)\right), \quad \text{(by defn. of } t(x)) \\
&= \frac{3^{z-1}}{|S|}\left(\sum_{x\in S} \text{service-cost}(f^*, S) + \sum_{x'\in S} \text{service-cost}(f^*, S) + \sum_{x'\in S} \text{service-cost}(f^*, S)\right) \\
&= \frac{3^{z-1}}{|S|}\left(3|S| \cdot \text{service-cost}(f^*, S)\right) \\
&= 3^z \cdot \text{service-cost}(f^*, S)
\end{aligned}$$

□

Note that, in the above lemma, it is not necessary to open a facility at the closest location from x. Rather, we can open a facility at a location that is at least as close to x as f^*. In the next lemma, we show that if we are allowed to open a facility location at client locations as well, then it gives a better approximation guarantee.

Lemma 8. *For a set $S \subseteq C$, let f^* be any center in L. If we uniformly sample a point x in S and open a facility at x, then the following identity holds:*

$$\mathbb{E}[\text{service-cost}(x, S)] \leq 2^z \cdot \text{service-cost}(f^*, S)$$

Proof. The proof follows from the following inequalities.

$$\begin{aligned}
\mathbb{E}[\text{service-cost}(x, S)] &= \frac{1}{|S|}\left(\sum_{x\in S} \text{service-cost}(x, S)\right) \\
&= \frac{1}{|S|}\left(\sum_{x\in S}\sum_{x'\in S} d(x, x')^z\right) \\
&\leq \frac{2^{z-1}}{|S|}\left(\sum_{x\in S}\sum_{x'\in S} (d(f^*, x')^z + d(x, f^*)^z)\right), \quad \text{(using Fact 3)} \\
&= \frac{2^{z-1}}{|S|}\Big(2|S| \cdot \text{service-cost}(f^*, S)\Big) \\
&= 2^z \cdot \text{service-cost}(f^*, S)
\end{aligned}$$

□

3.4 A Simple List k-Service Algorithm

In this section, we design an algorithm for the soft version of the list k-service problem. That is, we can open more than one facility at a location in L while keeping the total number of open facilities to most k. The algorithm is simple; however, it can not be extended to the hard-assignment version. We use the techniques similar to Section 2.4; we convert the bi-criteria approximation algorithm for the unconstrained k-service problem to the list k-service algorithm. The bi-criteria approximation algorithm is defined as follows:

Definition 37 (Bi-criteria Approximation)**.** *Let $\mathcal{I} = (L, C, k, d, z)$ be any instance of the un-*

constrained k-service problem. An (α, β) bi-criteria approximation algorithm is an algorithm that outputs a set $F' \subseteq L$ of βk facilities such that the cost of the client set C with respect to F' is at most α times the optimal cost of the instance. That is,

$$\text{service-cost}(F', C) \leq \alpha \cdot \min_{\text{k-center-set } F} \Big\{\text{service-cost}(F, C)\Big\} = \alpha \cdot \text{OPT}(L, C, k)$$

We convert an (α, β) bi-criteria approximate solution to a list k-service solution using the following theorem:

Lemma 9. *Let $\mathcal{I} = (L, C, k, d, z)$ be any instance of the k-service problem. Let S be any (α, β) bi-criteria approximate solution of $\mathcal{I}$. Let $\mathcal{L}$ be the list of all k-sized soft-subsets of S. Then, for any arbitrary partitioning $\mathbb{O} = \{O_1, \ldots, O_k\}$ of C, there exists a k-sized set S' in the list $\mathcal{L}$ that gives $(3^{z-1} \cdot (\alpha + 2))$-approximation to the optimal assignment cost of $\mathbb{O}$. That is,*

$$\Phi(S', \mathbb{O}) \leq (3^{z-1} \cdot (\alpha + 2)) \cdot \Phi^*(\mathbb{O}).$$

Proof. Let $F_{\mathbb{O}} = \{f_1, \ldots, f_k\} \subseteq L$ be an optimal soft-center set of $\mathbb{O}$. That is, $\Phi(F_{\mathbb{O}}, \mathbb{O}) = \Phi^*(\mathbb{O})$. For any point $x \in C \cup F_{\mathbb{O}}$, let $g(x)$ denote a facility in S that is closest to x. That is, $g(x) \coloneqq \arg\min_{s \in S} \big\{d(s, x)\big\}$. We define a new soft-center set $S' \coloneqq \{g(f_1), \ldots, g(f_k)\} \subseteq S$. We show that S' is a $(3^{z-1} \cdot (\alpha + 2))$-approximate solution to $\mathbb{O}$. The proof follows from the following sequence of inequalities:

$$\begin{aligned}
\Phi(S', \mathbb{O}) &= \sum_{i=1}^{k} \sum_{x \in O_i} \{d(x, g(f_i))^z\} \\
&\leq \sum_{i=1}^{k} \sum_{x \in O_i} \Big(d(x, f_i) + d(f_i, g(f_i))\Big)^z, && \text{(using triangle inequality)} \\
&\leq \sum_{i=1}^{k} \sum_{x \in O_i} \Big(d(x, f_i) + d(f_i, g(x))\Big)^z, && \text{(from definition of } g(f_i))
\end{aligned}$$

$$\leq \sum_{i=1}^{k} \sum_{x \in O_i} \Big(d(x, f_i) + d(f_i, x) + d(x, g(x)) \Big)^z, \qquad \text{(using triangle inequality)}$$

$$\leq 3^{z-1} \cdot \sum_{i=1}^{k} \sum_{x \in O_i} \Big(d(x, f_i)^z + d(f_i, x)^z + d(x, g(x))^z \Big), \quad \text{(using Fact 3)}$$

$$= 3^{z-1} \cdot \sum_{i=1}^{k} \sum_{x \in O_i} \Big(2 \cdot d(x, f_i)^z + d(x, g(x))^z \Big),$$

$$= 3^{z-1} \cdot 2 \cdot \Phi^*(\mathbb{O}) + 3^{z-1} \cdot \sum_{i=1}^{k} \sum_{x \in O_i} d(x, g(x))^z,$$

$$= 3^{z-1} \cdot 2 \cdot \Phi^*(\mathbb{O}) + 3^{z-1} \cdot \sum_{x \in C} d(x, g(x))^z,$$

$$= 3^{z-1} \cdot 2 \cdot \Phi^*(\mathbb{O}) + 3^{z-1} \cdot \text{service-cost}(S, C),$$

$$= 3^{z-1} \cdot 2 \cdot \Phi^*(\mathbb{O}) + 3^{z-1} \cdot \alpha \cdot \text{OPT}(L, C, k), \qquad (\because S \text{ is an } \alpha\text{-approximation})$$

$$\leq 3^{z-1} \cdot (2 + \alpha) \cdot \Phi^*(\mathbb{O}),$$

($\because$ unconstrained cost is always smaller then the constrained cost)

This proves that S' is a $(3^{z-1} \cdot (\alpha + 2))$-approximate solution to $\mathbb{O}$. Note that $S' \subseteq S$ and $\mathcal{L}$ is the list of k-sized soft-subsets of S; therefore $S' \in \mathcal{L}$. This proves the lemma. □

To use the above lemma, we require a bi-criteria approximation algorithm for the unconstrained k-service problem. Interestingly, there exists a polynomial time randomized $\big(1 + \varepsilon, O(\ln(1/\varepsilon))\big)$ bi-criteria approximation algorithm due to Neal Young [140]. The algorithm outputs $O(k \ln(1/\varepsilon))$ centers such that the expected unconstrained k-service cost of the client set with respect to the center set is at most $(1 + \varepsilon)$ times the optimal. There also exists a polynomial time deterministic $\big(1 + \varepsilon, O(\ln(1/\varepsilon)/\varepsilon)\big)$ bi-criteria approximation algorithm due to Chandra Chekuri [41]. We substitute the algorithm of Chandra Chekuri [41] in the previous

lemma; therefore, $\alpha = 1 + \varepsilon$ and $\beta = O(\ln(1/\varepsilon)/\varepsilon)$. We set $\varepsilon = \varepsilon'/3^{z-1}$ for some constant $\varepsilon' \leq 1$. Then, the above lemma gives the following result for the list k-service problem:

Corollary 10 (Main Result)**.** *Let $\mathcal{I} = (L, C, k, d, z, \mathbb{O}, \varepsilon)$ be any instance of the list k-service problem. Then, there is an algorithm that outputs a list $\mathcal{L}$ of size $(k \ln(1/\varepsilon)/\varepsilon)^{O(k)}$ such that it contains a k-sized soft-center-set that is $(3^z + \varepsilon')$-approximation to the optimal clustering cost of $\mathbb{O}$. The running time of the algorithm is $(k \ln(1/\varepsilon)/\varepsilon)^{O(k)} \cdot n^{O(1)}$, which is* FPT *in k.*

The above algorithm is simple; however, it has the following main limitations:

1. The algorithm does not hold for the hard assignment version of the list k-service problem. The set S' obtained using $F_{\mathbb{O}}$ could have duplicate facilities since it is possible that two distinct facilities $f_i, f_j \in F_{\mathbb{O}}$, their closest facilities in $g(f_i)$ and $g(f_j)$ in S could be the same.

2. The algorithm does not give improved approximation guarantee for the special case when $C \subseteq L$.

3. We do not know how to extend the algorithm to the outlier setting while achieving the same approximation guarantee of $(3^z + \varepsilon)$.

4. We do not know how to convert the algorithm to the streaming algorithm.

In the next section, we use a different approach to solve the problem, and we do not face these limitations.

3.5 Algorithm for List Outlier k-Service Problem

In this section, we design an FPT (in m and k) time algorithm for the list outlier k-service problem with running time $(k + m)^{O(kz^2)} \cdot n^{O(1)}$. It implies a $k^{O(kz^2)} \cdot n^{O(1)}$ time algorithm for the list k-service problem without outliers. Moreover, our algorithm works in the outlier

setting. Our algorithm is based on D^z-sampling technique: given a center set F and client set C, sampling a point from the client set C w.r.t. center set F using the distribution where the sampling probability of a client $x \in C$ is $\frac{\text{service-cost}(F,\{x\})}{\text{service-cost}(F,C)} = \frac{\min_{f\in F} d(f,x)^z}{\sum_{y\in C}\min_{f\in F} d(f,y)^z}$. If F is empty, then D^z-sampling is the same as uniform sampling. Please see Algorithm 3.1 for the list outlier k-service problem.

`List-Outlier-k-Service` $(L, C, k, d, z, \varepsilon, m)$

Inputs: Outlier k-service instance (L, C, k, d, z, m) and accuracy ε
Output: A list $\mathcal{L}$, each element in $\mathcal{L}$ being a k-center set
Constants: $\beta = 4^{z-1}\cdot\left(\frac{z^z\cdot 3^{z^2+4z+3}}{\varepsilon^{z+1}}+1\right)$; $\gamma = \frac{z^z\cdot 3^{z^2+5z+1}}{\varepsilon^z}$; $\eta = \frac{\alpha\,\beta\,\gamma\,k\cdot 3^{z+2}}{\varepsilon^2}$

(1) Run any α-approximation algorithm with $\alpha = poly(k+m)$ for the *unconstrained* $(k+m)$-service instance $(C, C, k+m, d, z)$ and let F be the obtained center-set. (*k-means++ [16] is one such algorithm.*)
(2) $\mathcal{L} \leftarrow \emptyset$
(3) Repeat $(\log n)\cdot 2^k$ times:
(4) Sample a multi-set M of ηk points from C using D^z-sampling w.r.t. center set F
(5) $M \leftarrow M \cup F$
(6) $T \leftarrow \emptyset$
(7) For every point x in M:
(8) $T \leftarrow T \cup \{k$ facilities in L that are closest to $x\}$
(9) For all subsets S of T of size k:
(10) $\mathcal{L} \leftarrow \mathcal{L} \cup \{S\}$
(11) return($\mathcal{L}$)

Algorithm 3.1: Algorithm for the list outlier k-service problem

Let us discuss some of the main ideas of the algorithm and its analysis. First, note that as per the algorithm description, the list size is $2^k \cdot \binom{(\eta+1)k^2+mk}{k} \cdot (\log n)$, which is bounded by $O\left((\log n)\cdot((k+m)/\varepsilon)^{O(kz^2)}\right)$ for the parameters given. This is because in step (9), the algorithm considers all possible k sized subsets of (multi)set T of size $(\eta+1)k^2+mk$. We now discuss the approximation guarantee. Note that in the first step, we obtain a $(m+k)$-sized center-set $F \subseteq C$ which is an α-approximation for the *unconstrained* $(k+m)$-service instance $(C, C, k+m, d, z)$. Any α polynomial in k and m suffices for our analysis. That

is, service-cost$(F, C) \leq \alpha \cdot$ OPT$(C, C, k+m)$. One such algorithm is the k-means++ algorithm [16] that gives an $O(4^z \cdot \log(k+m))$-approximation guarantee and a running time $O(n(k+m))$, where $k+m$ are the number of centers.

Now, let us see how the center-set F can help us. Let us focus on any cluster O_i of a target clustering $\mathbb{O} = \{O_1, \ldots, O_k\}$. We note that the closest facility to a uniformly sampled client from any client set O_i provides a constant approximation to the optimal 1-median/means cost for O_i in expectation (see Lemma 7). Unfortunately, we cannot uniformly sample from O_i directly since O_i is not known to us. Given this, our main objective should be to use F to try to sample uniformly from O_i so that we could achieve a constant approximation for O_i. Let us do a case analysis based on the distance of points in O_i from the nearest point in F. Consider the following two possibilities: The first possibility is that the points in O_i are close to F. If this is the case, we can uniformly sample a point from F instead of O_i. This would incur some extra cost. However, the cost is small and can be bounded. To cover this first possibility, the algorithm adds the entire set F to the set of sampled points M (see line (5) of the algorithm). The second possibility is that the points in O_i are far away from F. In this case, we can D^z-sample the points from C. Since the points in O_i are far away, the sampled set would contain a good portion of points from O_i, and the points will be *almost* uniformly distributed. We will show that almost uniform sampling is sufficient to apply Lemma 7 on O_i. However, we would have to sample many points to boost the success probability. This requirement is taken care of by line (4) of the algorithm. Note that we may need to use a hybrid approach for analysis since the real case may be a combination of the first and second possibilities. Most of the ingenuity of this work lies in formulating and proving appropriate sampling lemmas to make this hybrid analysis work.

To apply Lemma 7, we need to fulfill one more condition. We need the closest facility location from a sampled point. This requirement is handled by lines (7) and (8) of the algorithm. However, note that the algorithm picks k-closest facility locations instead of just one facility

location. We will show that this step is crucial to obtain a *hard-assignment* solution for the problem. Finally, the algorithm adds all the potential center sets to a list $\mathcal{L}$ (see lines (9) and (10) of the algorithm). The algorithm repeats this procedure $(\log n) \cdot 2^k$ times to boost the success probability (see line (3) of the algorithm). We prove the following result (restatement of Theorem 32):

Theorem 36. *Let $0 < \varepsilon \leq 1$ and $z \geq 1$. Let (L, C, k, d, z, m) be any outlier-k-service instance. Let $\mathbb{O} = \{O_1, O_2, \ldots, O_k\}$ be any arbitrary clustering of some arbitrary subset $C' \subseteq C$ of size at least $|C| - m$. The algorithm* `List-Outlier-k-Service`*, with probability at least $1 - 1/n$, outputs a list $\mathcal{L}$ of size $O\left((\log n) \cdot ((k+m)/\varepsilon)^{O(k\,z^2)}\right)$ such that there is a k-center-set $S \in \mathcal{L}$ in the list such that $\Phi(S, \mathbb{O}) \leq (3^z + \varepsilon) \cdot \Phi^*(\mathbb{O})$. Moreover, the running time of the algorithm is $O\left(n \cdot ((k+m)/\varepsilon)^{O(k\,z^2)}\right)$. For the special case when $C \subseteq L$, the approximation guarantee is $(2^z + \varepsilon)$.*

Firstly, we analyse the case when C is not necessarily a subset of L. The analysis of the special case $C \subseteq L$ is similar; however there are subtle differences. We analyse the special case in Section 3.6.

Let $Z \subseteq C$ be (unknown) target set of at most m outliers, $\mathbb{O} = \{O_1, O_2, \ldots, O_k\}$ be the (unknown) target clustering of $C \setminus Z$, and $F^* = \{f_1^*, f_2^*, \ldots, f_k^*\}$ be the optimal center set of $\mathbb{O}$. Suppose O_i is assigned to f_i^*, and $\Delta(O_i)$ denote its assignment cost, i.e., $\Delta(O_i) =$ service-cost(f_i^*, O_i). We classify the clusters into two categories: *low-cost* cluster set W and *high-cost* cluster set H.

$$W \coloneqq \{O_i \mid \text{service-cost}(F, O_i) \leq \frac{\varepsilon}{\alpha\,\gamma\,k} \cdot \text{service-cost}(F, C), \text{ for } 1 \leq i \leq k\}$$

$$H \coloneqq \{O_i \mid \text{service-cost}(F, O_i) > \frac{\varepsilon}{\alpha\,\gamma\,k} \cdot \text{service-cost}(F, C), \text{ for } 1 \leq i \leq k\}$$

In other words, the set W contains the clusters with low-cost and H contains the clusters with

high-cost with respect to F. The set M that is obtained in lines (4) and (5) of the algorithm has the following property:

> **Property-I**: For any cluster $O_i \in \{O_1, O_2, \ldots, O_k\}$, with probability at least $1/2$, there is a point s_i in M such that the following holds:
>
> $$\text{service-cost}(t(s_i), O_i) \leq \begin{cases} \left(3^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i) + \frac{\varepsilon}{2^{z+1}k} \cdot \text{OPT}(C, C, k+m), & \text{if } O_i \in W \\ \left(3^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i), & \text{if } O_i \in H \end{cases}$$
>
> where $t(s_i)$ denotes any facility location that is closer to s_i than f_i^*, i.e., $d(s_i, t(s_i)) \leq d(s_i, f_i^*)$.

We prove the above property in Sections 3.5.1 and 3.5.2. Next, we note the following lemma:

Lemma 10. *Let $Z \subseteq C$ be any arbitrary set of at most m outliers, $\mathbb{O} = \{O_1, \ldots, O_k\}$ be any clustering defined over the client set $C \setminus Z$, and $F^* = \{f_1^*, f_2^*, \ldots, f_k^*\} \subseteq L$ be an optimal center set of $\mathbb{O}$. Then* $\text{OPT}(C, C, k+m) \leq 2^z \cdot \sum_{i=1}^{k} \Delta(O_i)$*, where* $\Delta(O_i) = \text{service-cost}(f_i^*, O_i)$.

Proof. Let $F_c \coloneqq \{g(f_1^*), g(f_2^*), \ldots, g(f_k^*)\}$ be the set such that $g(f_i^*)$ denote the client in O_i that is closest from f_i^*. Note that $\Phi(F_c, \mathbb{O}) \leq \sum_{i=1}^{k} 2^z \cdot \Delta(O_i)$. The proof follows from the following sequence of inequalities:

$$\begin{aligned} \Phi(F_c, \mathbb{O}) &= \sum_{i=1}^{k} \sum_{x \in O_i} d(x, g(f_i^*))^z \\ &\leq \sum_{i=1}^{k} \sum_{x \in O_i} \left(d(x, f_i^*)^z + d(f_i^*, g(f_i^*))^z\right) && \text{(using triangle inequality)} \\ &\leq \sum_{i=1}^{k} \sum_{x \in O_i} \left(d(x, f_i^*) + d(f_i^*, x)\right)^z && \text{(using the definition. of } g(f_i^*)) \\ &= 2^z \cdot \sum_{i=1}^{k} \sum_{x \in O_i} d(x, f_i^*)^z \end{aligned}$$

$$= 2^z \cdot \sum_{i=1}^{k} \Delta(O_i)$$

Now, consider each point in $Z = \{z_1, \ldots, z_m\}$, as a cluster of its own. Let us denote these clusters by $\{O_{k+1}, O_{k+2}, \ldots, O_{k+m}\}$, i.e., $O_{k+i} = \{z_i\}$. Based on it, let us define a new clustering $\mathbb{O}' \coloneqq \{O_1, O_2, \ldots, O_{k+m}\}$ having $k+m$ clusters. Let $F' \coloneqq F_c \cup Z$ be a center-set for $\mathbb{O}'$ such that for $1 \leq i \leq k$, a cluster O_i is assigned to f_i^*; and for $k+1 \leq i \leq k+m$, a cluster O_{k+i} is assigned to z_i. Thus we get $\Phi(F', \mathbb{O}') = \Phi(F_c, \mathbb{O}) \leq \sum_{i=1}^{k} 2^z \cdot \Delta(O_i)$. Moreover, note that F' is a feasible center set and $\mathbb{O}'$ is a feasible clustering for the unconstrained $(k+m)$-service instance $(C, C, k+m, d, z)$. Thus OPT$(C, C, k+m) \leq \Phi(F', \mathbb{O}')$. Hence we get OPT$(C, C, k+m) \leq 2^z \cdot \sum_{i=1}^{k} \Delta(O_i)$. This proves the lemma.

□

Now we use the above lemma and property to prove Theorem 36. The lemma and property together implies that $T_s \coloneqq \{t(s_1), t(s_2), \ldots, t(s_k)\}$ is a $\left(3^z + \varepsilon\right)$-approximation for $\mathbb{O}$, with probability at least $1/2^k$. [4] Now note that the facility locations that are closest to s_i satisfy the definition of $t(s_i)$. Moreover, the algorithm adds one such facility location to set T (see line (8) of the algorithm). Therefore there is a center-set T_s in the list that gives $(3^z + \varepsilon)$-approximation for $\mathbb{O}$. To boost the success probability to $1 - 1/n$, the algorithm repeats the process $(\log n) \cdot 2^k$ times (see line (3) of the algorithm). Based on these arguments, it looks like we got the desired result. However, there is one issue that we need to take care of. Remember, we are looking for a hard assignment for the problem, and the set T_s could be a soft center-set, since the closest facility locations might be the same for two s_i's. In other words, $t(s_i)$ could be the same as $t(s_j)$ for some $i \neq j$. However, we show that there is another hard center-set in the list $\mathcal{L}$, that gives the required approximation for the problem. To obtain a hard center-set, we use of line (8) of the algorithm. In line (8), the algorithm picks the k closest facility locations from L instead of

[4] Note that the probabilities can be multiplied since M can be partitioned into k groups and we actually show that the good point s_i for O_i is either in F or is in any group with probability at least $1/2$.

just one. Note that it is not necessary to open a facility at a closest location in L. Rather we can open a facility at any location f in L that is at least as close to s_i as f_i^*, i.e., $d(s_i, f) \leq d(s_i, f_i^*)$ (see Property-I).

Let $T(s_i)$ denote a set of k closest facility location for s_i. We show that there is a hard center-set $T_h \coloneqq \{f_1, \ldots, f_k\} \subset \bigcup_i T(s_i)$, such that $d(s_i, f_i) \leq d(s_i, f_i^*)$ for every $1 \leq i \leq k$. We define T_h using the following simple subroutine:

```
FindFacilities
(1)  T_h ← ∅
(2)  for i ∈ {1, ..., k}:
(3)      if f_i^* ∈ T(s_i):
(4)          T_h ← T_h ∪ {f_i^*}
(5)      else
(6)          let f be any facility in T(s_i) that is not in T_h
(7)          T_h ← T_h ∪ {f}
```

Lemma 11. *$T_h = \{f_1, f_2, \ldots, f_k\}$ contains exactly k different facilities such that for every $1 \leq i \leq k$, we have $d(s_i, f_i) \leq d(s_i, f_i^*)$.*

Proof. Firstly, we show that all facilities in T_h are different. Since f_i^* is different for different clusters, the *if statement* always adds different facilities to T_h. And, the *else statement* only adds a facility to T_h that is not present in T_h. Therefore, all the facilities in T_h are different.

Now we prove the second property, i.e., $d(s_i, f_i) \leq d(s_i, f_i^*)$ for every $1 \leq i \leq k$. The property is trivially true for the facilities that are added in the *if statement*. For the facilities added in the *else statement*, we know that $T(s_i)$ does not contain f_i^*. Since $T(s_i)$ is a set of k-closest facility locations, for any facility location f in $T(s_i)$, $d(s_i, f) \leq d(s_i, f_i^*)$. Thus any facility added in the *else statement* has $d(s_i, f) \leq d(s_i, f_i^*)$. This completes the proof. □

Thus $T_h \in \mathcal{L}$ is a hard center-set, which gives $(3^z+\varepsilon)$-approximation to the problem. We prove Property-I and Lemma 10 in the following subsections.

Now, we analyse Theorem 36 for the case when C is a subset of L. In other words, we are allowed to open a facility at a client location. For this case, the algorithm can open the facilities at the locations $\{s_1, s_2, \ldots, s_k\}$ instead of $t(s_i)$'s, and it would not need to execute lines (7) and (8). Furthermore, it can be shown that $\{s_1, s_2, \ldots, s_k\}$ gives $(2^z+\varepsilon)$-approximation to the optimal clustering cost of $\mathbb{O}$. However, note that $\{s_1, s_2, \ldots, s_k\}$ could be a soft center-set. To obtain a hard center-set, the algorithm must execute lines (7) and (8). In that case, the analysis differs slightly. We discuss these details in Section 3.6.

3.5.1 Proof of Property-I: Low-Cost Clusters

Let O_i be a low-cost cluster, i.e., service-cost$(F, O_i) \leq \frac{\varepsilon}{\alpha\,\gamma\,k} \cdot$ service-cost(F, C). For a point $x \in \mathcal{X}$, let $c(x)$ denote the closest location in F, i.e., $c(x) = \arg\min_{f\in F}\{d(x,f)\}$. Based on the definition of $c(x)$, consider a multi-set $M_i \coloneqq \{c(x) \mid x \in O_i\}$. Since O_i has a low cost with respect to F, the points in O_i are close to points from F. Consider uniformly sampling a point from M_i. In the next lemma, we show that a uniformly sampled point from M_i is a good enough center for O_i.

Lemma 12. *Let p be a point sampled uniformly at random from M_i. Then the following bound holds:*

$$\mathbb{E}[\text{service-cost}(t(p), O_i)] \leq \left(3^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i) + \frac{\varepsilon}{2^{z+1}\,k} \cdot \text{OPT}(C, C, k+m).$$

Proof. The proof follows from the following sequence of inequalities.

$\mathbb{E}[\text{service-cost}(t(p), O_i)]$

$$= \frac{1}{|O_i|} \cdot \left(\sum_{p \in M_i} \text{service-cost}(t(p), O_i) \right)$$

$$= \frac{1}{|O_i|} \cdot \left(\sum_{p \in M_i} \sum_{x \in O_i} d(x, t(p))^z \right)$$

$$= \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} \sum_{x \in O_i} d\big(x, t(c(x'))\big)^z \right)$$

$$\leq \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} \sum_{x \in O_i} (d(x, x') + d(x', c(x')) + d(c(x'), t(c(x'))))^z \right),$$

(using triangle inequality)

$$\leq \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} \sum_{x \in O_i} (d(x, x') + d(x', c(x')) + d(c(x'), f_i^*))^z \right),$$

(by the defn. of $t(x)$)

$$\leq \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} \sum_{x \in O_i} (d(x, x') + d(x', c(x')) + d(x', c(x')) + d(x', f_i^*))^z \right),$$

(using triangle inequality)

Let us use Fact 2, by setting $a = 2 \cdot d(x', c(x'))$ and $b = d(x, x') + d(x', f_i^*)$. We get the following expression, for any $\delta > 0$:

$$\leq \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} \sum_{x \in O_i} \left(\left(1 + \frac{1}{\delta}\right)^z \cdot (d(x, x') + d(x', f_i^*))^z + (1+\delta)^z \cdot 2^z \cdot d(x', c(x'))^z \right) \right)$$

$$= \left(1 + \frac{1}{\delta}\right)^z \cdot \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} \sum_{x \in O_i} (d(x, x') + d(x', f_i^*))^z \right) +$$

$$(1+\delta)^z \cdot \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} |O_i| \cdot 2^z \cdot d(x', c(x'))^z \right)$$

By lemma 7, we have

$\mathbb{E}[\text{service-cost}(t(x), O_i)] \leq \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} \sum_{x \in O_i} (d(x, x') + d(x', f_i^*))^z \right) \leq 3^z \cdot \Delta(O_i)$. Thus,

we get:

$\mathbb{E}[\text{service-cost}(t(p), O_i)]$

$$\leq \left(1+\frac{1}{\delta}\right)^z \cdot 3^z \cdot \Delta(O_i) + (1+\delta)^z \cdot \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} |O_i| \cdot 2^z \cdot d(x', c(x'))^z\right)$$

$$= \left(1+\frac{1}{\delta}\right)^z \cdot 3^z \cdot \Delta(O_i) + 2^z(1+\delta)^z \cdot \text{service-cost}(F, O_i)$$

$$= \left(1+\frac{\varepsilon}{z \cdot 3^{z+2}}\right)^z \cdot 3^z \cdot \Delta(O_i) + 2^z(1+\delta)^z \cdot \text{service-cost}(F, O_i),$$

$$\text{(by substituting } \delta = \frac{z \cdot 3^{z+2}}{\varepsilon} \text{)}$$

$$\leq \left(1+2z \cdot \frac{\varepsilon}{z \cdot 3^{z+2}}\right) \cdot 3^z \cdot \Delta(O_i) + 2^z(1+\delta)^z \cdot \text{service-cost}(F, O_i),$$

(using Fact 1)

$$\leq \left(3^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i) + 2^z(1+\delta)^z \cdot \text{service-cost}(F, O_i)$$

$$\leq \left(3^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i) + 2^z(2\delta)^z \cdot \text{service-cost}(F, O_i), \qquad (\because 1 \leq \delta, \text{ for } \varepsilon \leq 1)$$

$$\leq \left(3^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i) + \frac{4^z \cdot \delta^z \cdot \varepsilon}{\alpha\,\gamma\,k} \cdot \text{service-cost}(F, C),$$

$$(\because \text{service-cost}(F, O_i) \leq \frac{\varepsilon}{\alpha\,\gamma\,k} \cdot \text{service-cost}(F, C))$$

$$\leq \left(3^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i) + \frac{\varepsilon}{2^{z+1}\,\alpha\,k} \cdot \text{service-cost}(F, C), \qquad (\because \gamma = \frac{z^z \cdot 3^{z^2+5z+1}}{\varepsilon^z})$$

$$\leq \left(3^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i) + \frac{\varepsilon}{2^{z+1}\,k} \cdot \text{OPT}(C, C, k+m),$$

$$(\because \text{service-cost}(F, C) \leq \alpha \cdot \text{OPT}(C, C, k+m))$$

This completes the proof of the lemma. □

Since the above lemma estimates the average cost corresponding to a sampled point, there has to be a point p in M_i such that service-cost$(t(p), O_i) \leq \left(3^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i) + \frac{\varepsilon}{2^{z+1}\, k} \cdot$ OPT$(C, C, k+m)$. Since M_i is only composed of the points from F, and we keep the entire set F in M (see line (5) of the algorithm), therefore Property-I is satisfied for every low-cost cluster $O_i \in W$.

3.5.2 Proof of Property-I: High-Cost Clusters

Let O_i be a high-cost cluster, i.e., service-cost$(F, O_i) > \frac{\varepsilon}{\alpha\, \gamma\, k} \cdot$ service-cost(F, C). Since the cost of the cluster is high, some points of O_i are far away from the center set F. We partition O_i into two sets: O_i^n and O_i^f, as follows.

$$O_i^n \coloneqq \{x \mid d(c(x), x)^z \leq R^z, \text{ for } x \in O_i\}, \quad \text{where } R^z = \frac{1}{\beta} \cdot \frac{\text{service-cost}(F, O_i)}{|O_i|}$$

$$O_i^f \coloneqq \{x \mid d(c(x), x)^z > R^z, \text{ for } x \in O_i\}, \quad \text{where } R^z = \frac{1}{\beta} \cdot \frac{\text{service-cost}(F, O_i)}{|O_i|}$$

In other words, O_i^n represents the set of points that are *near* to the center set F and O_i^f represents the set of points that are *far* from the center set F. Recall that our prime objective is to obtain a uniform sample from O_i so we can apply lemma 7. To achieve that we consider sampling from O_i^n and O_i^f separately. The idea is as follows. To sample a point from O_i^f, we use the D^z-sampling technique and show that it gives an almost uniform sample from O_i^f. For O_i^n, we use F as its proxy and sample a point from F instead. However, doing so would incur an extra cost. We show that the extra cost is proportional to service-cost(F, O_i^n), which can be bounded

easily. To bound the extra cost, we use the following lemma.

Lemma 13. *For* $R^z = \frac{1}{\beta} \cdot \frac{\text{service-cost}(F, O_i)}{|O_i|}$*, we have* $\text{service-cost}(F, O_i^n) \leq \frac{\varepsilon^{z+1}}{z^z \cdot 3^{z^2+5z+2}} \cdot \Delta(O_i)$.

Proof. We have,

$$\begin{aligned}
\Delta(O_i) &\geq \text{service-cost}(f_i^*, O_i^n) \\
&= \sum_{x \in O_i^n} d(f_i^*, x)^z \\
&\geq \sum_{x \in O_i^n} \left(\frac{d(c(x), f_i^*)^z}{2^{z-1}} - d(x, c(x))^z \right), \quad \text{(using Fact 3)} \\
&= \sum_{x \in O_i^n} \left(\frac{d(c(x), f_i^*)^z}{2^{z-1}} \right) - \text{service-cost}(F, O_i^n) \\
&\geq \sum_{x \in O_i^n} \left(\frac{d(c(f_i^*), f_i^*)^z}{2^{z-1}} \right) - \text{service-cost}(F, O_i^n).
\end{aligned}$$

Using Fact 3, we get $\text{service-cost}(c(f_i^*), O_i) \leq 2^{z-1} \cdot (\Delta(O_i) + |O_i| \cdot d(c(f_i^*), f_i^*)^z)$. Since $\text{service-cost}(F, O_i) \leq \text{service-cost}(c(f_i^*), O_i)$, we get $d(c(f_i^*), f_i^*)^z \geq \frac{\text{service-cost}(F, O_i) - 2^{z-1} \cdot \Delta(O_i)}{2^{z-1}|O_i|}$. Using this, the previous expression simplifies to:

$$\begin{aligned}
\Delta(O_i) &\geq |O_i^n| \left(\frac{\text{service-cost}(F, O_i) - 2^{z-1} \cdot \Delta(O_i)}{4^{z-1} \cdot |O_i|} \right) - \text{service-cost}(F, O_i^n) \\
&= |O_i^n| \cdot \frac{\beta R^z}{4^{z-1}} - |O_i^n| \cdot \frac{\Delta(O_i)}{2^{z-1} \cdot |O_i|} - \text{service-cost}(F, O_i^n),
\end{aligned}$$

$$(\because R^z = \frac{1}{\beta} \cdot \frac{\text{service-cost}(F, O_i)}{|O_i|})$$

$$\geq \text{service-cost}(F, O_i^n) \cdot \frac{\beta}{4^{z-1}} - |O_i^n| \cdot \frac{\Delta(O_i)}{2^{z-1} \cdot |O_i|} - \text{service-cost}(F, O_i^n),$$

$$(\because \text{service-cost}(F, O_i^n) \leq |O_i^n| \cdot R^z)$$

$$\geq \frac{(\beta - 4^{z-1})}{4^{z-1}} \text{service-cost}(F, O_i^n) - \Delta(O_i), \qquad (\because |O_i^n| \leq |O_i| \leq 2^{z-1} \cdot |O_i|)$$

On rearranging the terms of the expression, we get

$$\text{service-cost}(F, O_i^n) \leq \frac{2 \cdot 4^{z-1}}{\beta - 4^{z-1}} \cdot \Delta(O_i)$$

$$\leq \frac{\varepsilon^{z+1}}{z^z \cdot 3^{z^2+5z+2}} \cdot \Delta(O_i) \qquad \because \beta = 4^{z-1} \cdot \left(\frac{z^z \cdot 3^{z^2+4z+3}}{\varepsilon^{z+1}} + 1 \right)$$

Hence proved. □

Since we are using F as a proxy for O_i^n, we define a multi-set $M_i^n := \{c(x) \mid x \in O_i^n\}$, where $c(x)$ denote the location in F that is closest to x, i.e., $c(x) = \arg\min_{f \in F}\{d(x, f)\}$. Furthermore, we define another multi-set $M_i := O_i^f \cup M_i^n$. In the following lemma, we show that there is a point in M_i that is a good center for O_i. The lemma is similar to lemma 12 of the low-cost clusters.

Lemma 14. *Let p be a point sampled uniformly at random from M_i. Then the following bound holds:*

$$\mathbb{E}[\text{service-cost}(t(p), O_i)] \leq \left(3^z + \frac{\varepsilon}{4}\right) \cdot \Delta(O_i)$$

Proof. $\mathbb{E}[\text{service-cost}(t(p), O_i)]$

$$= \frac{1}{|O_i|} \cdot \left(\sum_{p \in M_i} \text{service-cost}(t(p), O_i) \right)$$
$$= \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i^n} \text{service-cost}(t(c(x')), O_i) + \sum_{x' \in O_i^f} \text{service-cost}(t(x'), O_i) \right).$$

We evaluate these two terms separately.

1. The first term:

$$\sum_{x' \in O_i^n} \text{service-cost}(t(c(x')), O_i)$$
$$= \sum_{x' \in O_i^n} \sum_{x \in O_i} d(x, t(c(x')))^z$$
$$\leq \sum_{x' \in O_i^n} \sum_{x \in O_i} \left(d(x, x') + d(x', c(x')) + d(c(x'), t(c(x'))) \right)^z,$$

(using triangle-inequality)

$$\leq \sum_{x' \in O_i^n} \sum_{x \in O_i} \left(d(x, x') + d(x', c(x')) + d(c(x'), f_i^*) \right)^z,$$

(by the defn. of $t(c(x'))$)

$$\leq \sum_{x' \in O_i^n} \sum_{x \in O_i} \left(d(x, x') + d(x', c(x')) + d(x', c(x')) + d(x', f_i^*) \right)^z,$$

(using triangle-inequality)

$$\leq \sum_{x' \in O_i^n} \sum_{x \in O_i} \left(\left(1 + \frac{1}{\delta} \right)^z \cdot (d(x, x') + d(x', f_i^*))^z + 2^z (1 + \delta)^z \cdot d(x', c(x'))^z \right),$$

(using Fact 2).

2. The second term:

$$\sum_{x' \in O_i^f} \text{service-cost}(t(x'), O_i) \leq \sum_{x' \in O_i^f} \sum_{x \in O_i} (d(x, x') + d(x', t(x')))^z,$$

(using triangle-inequality)

$$\leq \sum_{x' \in O_i^f} \sum_{x \in O_i} (d(x, x') + d(x', f_i^*))^z,$$

(by the defn. of $t(x')$).

On combining the two terms, we get the following expression:

$$\mathbb{E}[\text{service-cost}(t(p), O_i)] \leq \left(1 + \frac{1}{\delta}\right)^z \cdot \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} \sum_{x \in O_i} (d(x, x') + d(x', f_i^*))^z\right) + (1+\delta)^z \cdot \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i^n} |O_i| \cdot 2^z d(x', c(x'))^z\right)$$

Using Lemma 7, we have $\mathbb{E}[\text{service-cost}(t(x), O_i)] \leq \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} \sum_{x \in O_i} (d(x, x') + d(x', f_i^*))^z\right) \leq 3^z \cdot \Delta(O_i)$. Thus, we get the following:

$\mathbb{E}[\text{service-cost}(t(p), O_i)]$

$$\leq \left(1 + \frac{1}{\delta}\right)^z \cdot 3^z \cdot \Delta(O_i) + (1+\delta)^z \cdot \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i^n} |O_i| \cdot 2^z \cdot d(x', c(x'))^z\right)$$

$$= \left(1 + \frac{1}{\delta}\right)^z \cdot 3^z \cdot \Delta(O_i) + 2^z (1+\delta)^z \cdot \text{service-cost}(F, O_i^n)$$

$$= \left(1 + \frac{\varepsilon}{z \cdot 3^{z+3}}\right)^z \cdot 3^z \cdot \Delta(O_i) + 2^z(1+\delta)^z \cdot \text{service-cost}(F, O_i^n),$$

$$\text{(by substituting } \delta = \frac{z \cdot 3^{z+3}}{\varepsilon}\text{)}$$

$$\leq \left(1 + 2z \cdot \frac{\varepsilon}{z \cdot 3^{z+3}}\right) \cdot 3^z \cdot \Delta(O_i) + 2^z(1+\delta)^z \cdot \text{service-cost}(F, O_i^n),$$

(using Fact 1)

$$= \left(3^z + \frac{\varepsilon}{8}\right) \cdot \Delta(O_i) + 2^z(1+\delta)^z \cdot \text{service-cost}(F, O_i^n),$$

$$= \left(3^z + \frac{\varepsilon}{8}\right) \cdot \Delta(O_i) + 2^z \cdot (2\delta)^z \cdot \text{service-cost}(F, O_i^n), \qquad \because 1 \leq \delta, \text{ for } \varepsilon \leq 1$$

$$\leq \left(3^z + \frac{\varepsilon}{8}\right) \cdot \Delta(O_i) + \frac{\varepsilon}{8} \cdot \Delta(O_i), \qquad \text{(using lemma 13)}$$

$$= \left(3^z + \frac{\varepsilon}{4}\right) \cdot \Delta(O_i)$$

This completes the proof of the lemma. □

We obtain the following corollary by applying Markov's inequality to the previous lemma.

Corollary 11. *For any $0 < \varepsilon \leq 1$ and point p sampled uniformly at random from M_i, we have:*

$$\textbf{Pr}\left[\text{service-cost}(t(p), O_i) \leq \left(3^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i)\right] > \frac{\varepsilon}{3^{z+2}}.$$

Let us call a point p, a *good* point if $t(p)$ gives $\left(3^z + \frac{\varepsilon}{2}\right)$-approximation for O_i. We obtain the following corollary from the previous corollary.

Corollary 12. *There are at least* $\left\lceil \frac{\varepsilon \cdot |O_i|}{3^{z+2}} \right\rceil$ *good points in* M_i.

Now, our goal is to obtain one such good point from M_i. We are done if F contains any good point since the algorithm adds the entire set F to M. As a result, Property I is satisfied for high-cost cluster O_i. On the other hand, if F does not contain any good point, then there is no good point in M_i^n as well. It simply means that all good points are present in O_i^f. We use the D^z-sampling technique to sample these good points. Let $G \subseteq O_i^f$ denote the set of good points. Then we have $|G| \geq \left\lceil \frac{\varepsilon \cdot |O_i|}{3^{z+2}} \right\rceil$. Using this fact, we prove the following lemma.

Lemma 15. *For any point* $p \in O_i^f$ *and any* D^z*-sampled point* $x \in C$*, we have:* $\mathbf{Pr}[x = p] \geq \frac{\varepsilon}{\alpha\,\beta\,\gamma\,k\,|O_i|} = \tau$ *and* $\mathbf{Pr}\left[\text{service-cost}(t(x), O_i) \leq \left(3^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i)\right] \geq \frac{\varepsilon^2}{\alpha\,\beta\,\gamma\,k \cdot 3^{z+2}}$

Proof. For any point $p \in O_i^f$, $\mathbf{Pr}[x = p] =$

$$\frac{d^z(p, c(p))}{\text{service-cost}(F, C)} \geq \frac{R^z}{\text{service-cost}(F, C)} = \frac{1}{\beta\,|O_i|} \cdot \frac{\text{service-cost}(F, O_i)}{\text{service-cost}(F, C)} \geq \frac{\varepsilon}{\alpha\,\beta\,\gamma\,k\,|O_i|}$$

Let Z denote an indicator random variable, such that $Z = 1$ if service-cost$(t(x), O_i) \leq \left(3^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i)$ and 0 otherwise.

$$\begin{aligned}
\mathbf{Pr}[\,Z = 1\,] &\geq \sum_{p \in G} \mathbf{Pr}[\,x = p\,] \cdot \mathbf{Pr}[\,Z = 1 \mid x = p\,] \\
&\geq \sum_{p \in G} \frac{\varepsilon}{\alpha\,\beta\,\gamma\,k\,|O_i|} \cdot 1 \\
&= |G| \cdot \frac{\varepsilon}{\alpha\,\beta\,\gamma\,k\,|O_i|} \geq \frac{\varepsilon^2}{\alpha\,\beta\,\gamma\,k \cdot 3^{z+2}}
\end{aligned}$$

This completes the proof of the lemma. □

The above lemma states that every point in C_i^f has a sampling probability of at least τ. Moreover, a sampled point gives $\left(3^z + \frac{\varepsilon}{2}\right)$-approximation for O_i with probability at least $\frac{\varepsilon^2}{\left(\alpha\,\beta\,\gamma\,k \cdot 3^{z+2}\right)}$.

To boost this probability, we sample $\eta \coloneqq \frac{\alpha\,\beta\,\gamma\,k \cdot 3^{z+2}}{\varepsilon^2}$ points independently from C using D^z-sampling (see line (4) of the algorithm). It follows that, with probability at least $1/2$, there is a point in the sampled set that gives $\left(3^z + \frac{\varepsilon}{2}\right)$-approximation for O_i. Hence, Property-I is satisfied for high-cost cluster O_i. Also, in line(4) of the algorithm, we sample $\eta \cdot k$ points, i.e., η points corresponding to each cluster. Hence, Property-I holds for every cluster in H.

3.6 Analysis of List Outlier k-Service Algorithm: *Special Case* $C \subseteq L$

Since we are dealing with a special case, all the previous lemmas (i.e., Lemma 12, 13, 14, and 15) are valid here as well. We only update Lemmas 12 and 14 with approximation guarantee of 2^z instead of 3^z. We show that M has the following property when $C \subseteq L$:

> **Property-II**: When $C \subseteq L$, for any cluster $O_i \in \{O_1, O_2, \ldots, O_k\}$, there is a point s_i in M such that with probability at least $1/2$, the following holds:
>
> $$\text{service-cost}(u_i(s_i), O_i) \leq \begin{cases} \left(2^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i) + \frac{\varepsilon}{2^{z+1}\,k} \cdot \text{OPT}(C, C, k+m), & \text{if } O_i \in W \\ \left(2^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i), & \text{if } O_i \in H. \end{cases}$$
>
> Here, for any point $x \in C \cup L$, $u_i(x)$ denotes a location that is as close to x as its *closest* point in O_i. In other words, if $p = \arg\min_{y \in O_i}\{d(y, x)\}$ is the closest location in O_i, then $d(x, u_i(x)) \leq d(x, p)$.

We prove Property-II for the low-cost and high-cost clusters in 3.6.1 and 3.6.2, respectively. Us-

ing Property-II and Lemma 10, the set $\{u_1(s_1), u_2(s_2), \ldots, u_k(s_k)\}$ is $\left(2^z + \frac{\varepsilon}{2}\right)$-approximation for $\mathbb{O}$ with probability at least $1/2^k$. To boost the success probability to $1 - 1/n$, the algorithm repeats the procedure $(\log n) \cdot 2^k$ times (see line (3) of Algorithm 3.1). Note that s_i is always a possible candidate for $u(s_i)$, and it is added to the set T in line (8) of the algorithm. Therefore, $\mathcal{L}$ contains a set that is $\left(2^z + \frac{\varepsilon}{2}\right)$-approximation for $\mathbb{O}$. However, the set $\{s_1, s_2, \ldots, s_k\}$ could be a soft center-set. Now we show that there is a hard center-set $T_h = \{f_1, f_2, \ldots, f_k\}$ in the list, which gives $\left(2^z + \frac{\varepsilon}{2}\right)$-approximation. Our argument is based on the fact that we can open a facility at any location $u_i(s_i)$, which is at least as close to s_i as the closest location in O_i (see Property-II). Let p_i denote a location in O_i that is closest to s_i i.e. $p_i = \arg\min_{x \in O_i} d(x, s_i)$. We show that $d(s_i, f_i) \leq d(s_i, p_i)$ for every $1 \leq i \leq k$. The following analysis is very similar to the analysis that we did in Section 3.5 for the general case. The only difference is that f_i^* is replaced with p_i.

Let $T(s_i)$ denote a set of k closest facility locations for a sampled point $s_i \in M$ (see line(8) of the algorithm). We define $T_h = \{f_1, f_2, \ldots, f_k\}$ using the following simple subroutine:

```
FindFacilities
(1)  T_h ← ∅
(2)  for i ∈ {1, ..., k}:
(3)      if p_i ∈ T(s_i):
(4)          T_h ← T_h ∪ {p_i}
(5)      else
(6)          let f be any facility in T(s_i) that is not in T_h
(7)          T_h ← T_h ∪ {f}
```

Since size of each $T(s_i)$ is exactly k, T_h will contain k facilities at the end of the for-loop.

Lemma 16. *The subroutine picks a set $T_h = \{f_1, f_2, \ldots, f_k\}$ of k distinct facilities such that*

for every $1 \leq i \leq k$, $d(s_i, f_i) \leq d(s_i, p_i)$.

Proof. Firstly, we show that all facilities in T_h are different. Since, p_i is different for different clusters, the *if statement* adds facilities to T_h that are different. And, the *else statement* only adds a facility to T_h that is not present in T_h. Therefore, all the facilities in T_h are different.

Now we prove the second property, i.e., $d(s_i, f_i) \leq d(s_i, p_i)$ for every $1 \leq i \leq k$. The property is trivially true for the facilities added in the *if statement*. For the facilities added in the *else statement*, we know that $T(s_i)$ does not contain p_i. Since $T(s_i)$ is a set of k-closest facility locations, for any facility location f in $T(s_i)$, $d(s_i, f) \leq d(s_i, p_i)$. Thus any facility added in the *else statement* has $d(s_i, f) \leq d(s_i, p_i)$. This completes the proof. □

Thus $T_h \in \mathcal{L}$ is a hard center-set, which gives $(2^z + \varepsilon)$-approximation for the problem. Now we prove Property-II for the low-cost and high-cost clusters.

3.6.1 Proof of property-II: *Low-cost clusters*

Let O_i be a low-cost cluster, i.e., service-cost$(F, O_i) \leq \frac{\varepsilon}{\alpha\,\gamma\,k} \cdot$ service-cost(F, C). For a point $x \in \mathcal{X}$, let $c(x)$ denote the closest location in F, i.e., $c(x) = \arg\min_{f \in F}\{d(x, f)\}$. Based on the definition of $c(x)$, consider a multi-set $M_i \coloneqq \{c(x) \mid x \in O_i\}$. Since O_i has a low cost with respect to F, the points in O_i are close to points from F. Consider uniformly sampling a point from M_i. In the next lemma, we show that a uniformly sampled point from M_i is a good enough center for O_i.

Lemma 17. *Let* p *be a point sampled uniformly at random from* M_i*. Then the following bound holds:*

$$\mathbb{E}[\text{service-cost}(u_i(p), O_i)] \leq \left(2^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i) + \frac{\varepsilon}{2^{z+1}\,k} \cdot \text{OPT}(C, C, k + m)$$

Proof. The proof follows from the following sequence of inequalities.

$$\begin{aligned}
\mathbb{E}[\text{service-cost}(u_i(p), O_i)] &= \frac{1}{|O_i|} \cdot \left(\sum_{p \in M_i} \text{service-cost}(u_i(p), O_i) \right) \\
&= \frac{1}{|O_i|} \cdot \left(\sum_{p \in M_i} \sum_{x \in O_i} d(x, u_i(p))^z \right) \\
&= \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} \sum_{x \in O_i} d(x, u_i(c(x')))^z \right) \\
&\leq \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} \sum_{x \in O_i} \left(\, d(x, x') + d(x', c(x')) + d(c(x'), u_i(c(x'))) \, \right)^z \right), \\
&\qquad\qquad \text{(using triangle inequality)} \\
&\leq \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} \sum_{x \in O_i} \left(d(x, x') + d(x', c(x')) + d(c(x'), x') \right)^z \right), \\
&\qquad\qquad \text{(by the defn. of } u_i(c(x'))) \\
&= \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} \sum_{x \in O_i} \left(d(x, x') + 2d(x', c(x')) \right)^z \right)
\end{aligned}$$

Let us use Fact 2 by setting $b = d(x, x')$ and $a = 2 \cdot d(x', c(x'))$. We obtain the following expression, for any $\delta > 0$:

$$\begin{aligned}
&\leq \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} \sum_{x \in O_i} \left(\left(1 + \frac{1}{\delta} \right)^z \cdot d(x, x')^z + (1 + \delta)^z \cdot 2^z d(x', c(x'))^z \right) \right) \\
&= \left(1 + \frac{1}{\delta} \right)^z \cdot \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} \sum_{x \in O_i} d(x, x')^z \right) \\
&\qquad\qquad + (1 + \delta)^z \cdot \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} |O_i| \cdot 2^z d(x', c(x'))^z \right) \\
&= \left(1 + \frac{1}{\delta} \right)^z \cdot \mathbb{E}[\text{service-cost}(x, O_i)] + 2^z (1 + \delta)^z \cdot \text{service-cost}(F, O_i) \\
&\leq \left(1 + \frac{1}{\delta} \right)^z \cdot 2^z \cdot \Delta(O_i) + 2^z (1 + \delta)^z \cdot \text{service-cost}(F, O_i), \qquad \text{(using lemma 8)}
\end{aligned}$$

$$= \left(1 + \frac{\varepsilon}{z \cdot 3^{z+2}}\right)^z \cdot 2^z \cdot \Delta(O_i) + 2^z(1+\delta)^z \cdot \text{service-cost}(F, O_i),$$

$$\text{(by substituting } \delta = \frac{z \cdot 3^{z+2}}{\varepsilon}\text{)}$$

$$\leq \left(1 + 2z \cdot \frac{\varepsilon}{z \cdot 3^{z+2}}\right) \cdot 2^z \cdot \Delta(O_i) + 2^z(1+\delta)^z \cdot \text{service-cost}(F, O_i), \quad \text{(using Fact 1)}$$

$$= \left(2^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i) + 2^z(1+\delta)^z \cdot \text{service-cost}(F, O_i),$$

$$\leq \left(2^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i) + 2^z \cdot (2\delta)^z \cdot \text{service-cost}(F, O_i), \qquad \because 1 \leq \delta, \text{ for } \varepsilon \leq 1$$

$$\leq \left(2^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i) + \frac{4^z \cdot \delta^z \cdot \varepsilon}{\alpha\, \gamma\, k} \cdot \text{service-cost}(F, C),$$

$$(\because \text{service-cost}(F, O_i) \leq \frac{\varepsilon}{\alpha\, \gamma\, k} \cdot \text{service-cost}(F, C))$$

$$\leq \left(2^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i) + \frac{\varepsilon}{2^{z+1}\, \alpha\, k} \cdot \text{service-cost}(F, C), \qquad \because \gamma = \frac{z^z \cdot 3^{z^2+5z+1}}{\varepsilon^z}$$

$$\leq \left(2^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i) + \frac{\varepsilon}{2^{z+1}\, k} \cdot \text{OPT}(C, C, k),$$

$$(\because \text{service-cost}(F, C) \leq \alpha \cdot \text{OPT}(C, C, k+m))$$

This completes the proof of the lemma. □

By the above lemma, we can claim that there is a point p in M_i such that service-cost$(u_i(p), O_i) \leq \left(2^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i) + \frac{\varepsilon}{2^{z+1}\, k} \cdot \text{OPT}(C, C, k)$. Since M_i is only composed of the points from F and F is contained in M, Property- II is satisfied for every low-cost cluster $O_i \in W$. Next we

analyze the high-cost clusters.

3.6.2 Proof of property-II: *High-cost clusters*

Let O_i be a high-cost cluster, i.e., service-cost$(F, O_i) > \frac{\varepsilon}{\alpha\,\gamma\,k} \cdot$ service-cost(F, C). Since the cost of the cluster is high, some points of O_i are far away from the center set F. We partition O_i into two sets: O_i^n and O_i^f, as follows.

$$O_i^n \coloneqq \{x \mid d(c(x), x)^z \leq R^z, \text{ for } x \in O_i\}, \quad \text{where } R^z = \frac{1}{\beta} \cdot \frac{\text{service-cost}(F, O_i)}{|O_i|}$$

$$O_i^f \coloneqq \{x \mid d(c(x), x)^z > R^z, \text{ for } x \in O_i\}, \quad \text{where } R^z = \frac{1}{\beta} \cdot \frac{\text{service-cost}(F, O_i)}{|O_i|}$$

In other words, O_i^n represents the set of points that are *near* to the center set F and O_i^f represents the set of points that are *far* from the center set F. Recall that our prime objective is to obtain a uniform sample from O_i so we can apply lemma 7. To achieve that we consider sampling from O_i^n and O_i^f separately. The idea is as follows. To sample a point from O_i^f, we use the D^z-sampling technique and show that it gives an almost uniform sample from O_i^f. For O_i^n, we use F as its proxy and sample a point from F instead. However, doing so would incur an extra cost. We bound the extra cost using the Lemma 13 from the previous section. Since we are using F as a proxy for O_i^n, we define a multi-set $M_i^n \coloneqq \{c(x) \mid x \in O_i^n\}$, where $c(x)$ denote the location in F that is closest to x, i.e., $c(x) = \arg\min_{f \in F}\{d(x, f)\}$. Furthermore, we define another multi-set $M_i \coloneqq O_i^f \cup M_i^n$. In the following lemma, we show that there is a point in M_i that is a good center for O_i.

Lemma 18. *Let p be a point sampled uniformly at random from M_i. Then the following bound holds:*

$$\mathbb{E}[\text{service-cost}(u_i(p), O_i)] \leq \left(2^z + \frac{\varepsilon}{4}\right) \cdot \Delta(O_i)$$

Proof. $\mathbb{E}[\text{service-cost}(u_i(p), O_i)]$

$$= \frac{1}{|O_i|} \cdot \left(\sum_{p \in M_i} \text{service-cost}(u_i(p), O_i) \right)$$
$$= \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i^n} \text{service-cost}(u_i(c(x')), O_i) + \sum_{x' \in O_i^f} \text{service-cost}(u_i(x'), O_i) \right)$$

We evaluate these two terms separately.

1. The first term:

   $\sum_{x' \in O_i^n} \text{service-cost}(u_i(c(x')), O_i)$

   $$= \sum_{x' \in O_i^n} \sum_{x \in O_i} d(x, u_i(c(x')))^z$$
   $$\leq \sum_{x' \in O_i^n} \sum_{x \in O_i} \big(d(x, x') + d(x', c(x')) + d(c(x'), u_i(c(x'))) \big)^z,$$

   (using triangle-inequality)

   $$\leq \sum_{x' \in O_i^n} \sum_{x \in O_i} \big(d(x, x') + d(x', c(x')) + d(c(x'), x') \big)^z,$$

   (by the defn. of $u_i(c(x'))$)

   $$= \sum_{x' \in O_i^n} \sum_{x \in O_i} \big(d(x, x') + 2d(x', c(x')) \big)^z,$$
   $$\leq \sum_{x' \in O_i^n} \sum_{x \in O_i} \left(\left(1 + \frac{1}{\delta} \right)^z \cdot d(x, x')^z + 2^z (1 + \delta)^z \cdot d(x', c(x'))^z \right)$$

   (using Fact 2)

2. The second term:

$\sum_{x' \in O_i^f}$ service-cost$(u_i(x'), O_i)$

$$\leq \sum_{x' \in O_i^f} \sum_{x \in O_i} (d(x, x') + d(x', u_i(x')))^z, \quad \text{(using triangle-inequality)}$$

$$= \sum_{x' \in O_i^f} \sum_{x \in O_i} d(x, x')^z, \quad \text{(by the defn. of } u_i(x'), \text{ for } x' \in O_i^f, d(x', u_i(x')) = 0)$$

Now we combine the two terms; we get the following expression:

$\mathbb{E}[$service-cost$(u_i(p), O_i)]$

$$\leq \left(1 + \frac{1}{\delta}\right)^z \cdot \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i} \sum_{x \in O_i} d(x, x')^z\right) + (1+\delta)^z \cdot \frac{1}{|O_i|} \cdot \left(\sum_{x' \in O_i^n} |O_i| \cdot 2^z \cdot d(x', c(x'))^z\right)$$

$$= \left(1 + \frac{1}{\delta}\right)^z \cdot \mathbb{E}[\text{service-cost}(x, O_i)] + 2^z (1+\delta)^z \cdot \text{service-cost}(F, O_i^n)$$

$$\leq \left(1 + \frac{1}{\delta}\right)^z \cdot 2^z \cdot \Delta(O_i) + 2^z (1+\delta)^z \cdot \text{service-cost}(F, O_i^n), \quad \text{(using lemma 7)}$$

$$= \left(1 + \frac{\varepsilon}{z \cdot 3^{z+3}}\right)^z \cdot 2^z \cdot \Delta(O_i) + 2^z (1+\delta)^z \cdot \text{service-cost}(F, O_i^n),$$

$$\text{(by substituting } \delta = \frac{z \cdot 3^{z+3}}{\varepsilon})$$

$$\leq \left(1 + 2z \cdot \frac{\varepsilon}{z \cdot 3^{z+3}}\right) \cdot 2^z \cdot \Delta(O_i) + 2^z (1+\delta)^z \cdot \text{service-cost}(F, O_i^n), \quad \text{(using Fact 1)}$$

$$= \left(2^z + \frac{\varepsilon}{8}\right) \cdot \Delta(O_i) + 2^z (1+\delta)^z \cdot \text{service-cost}(F, O_i^n),$$

$$= \left(2^z + \frac{\varepsilon}{8}\right) \cdot \Delta(O_i) + 2^z \cdot (2\delta)^z \cdot \text{service-cost}(F, O_i^n), \qquad \because 1 \leq \delta, \text{ for } \varepsilon \leq 1$$

$$\leq \left(2^z + \frac{\varepsilon}{8}\right) \cdot \Delta(O_i) + \frac{\varepsilon}{8} \cdot \Delta(O_i), \quad \text{(using lemma 13)}$$

$$= \left(2^z + \frac{\varepsilon}{4}\right) \cdot \Delta(O_i)$$

This completes the proof of the lemma. □

We obtain the following corollary by applying Markov's inequality to the previous lemma.

Corollary 13. *If we sample a point $p \in M_i$, uniformly at random, then for $\varepsilon \leq 1$:*

$$\textbf{Pr}\left[\text{service-cost}(u_i(p), O_i) \leq \left(2^z + \frac{\varepsilon}{2}\right) \cdot \Delta(O_i)\right] \geq \frac{\varepsilon}{2^{z+2}} > \frac{\varepsilon}{3^{z+2}}$$

Since Corollary 11 is similar to Corollary 13, all the further arguments made in Section 3.5.2 are valid here as well. Thus with probability at least $1/2$, there is a point x in the sampled set M such that $u_i(x)$ gives $\left(2^z + \frac{\varepsilon}{2}\right)$-approximation for O_i. In other words, Property-II holds for every high-cost cluster in H.

3.7 A Matching Lower Bound on Approximation

In this section, we show that the algorithm `List-Outlier-k-Service` does not provide better than $(3^z - \delta')$ approximation guarantee for arbitrarily small $\delta' > 0$ (and $2^z - \delta'$ for the special case $C \subseteq L$). In other words, the analysis of our algorithm is tight. To show this, we create a *bad instance* for the problem in the following manner. Let $G = (V, E)$ be an undirected weighted graph with vertex set $V \coloneqq C \cup L$. The shortest weighted path between two vertices defines the distance metric $d(.,.)$. The set C is partitioned into subsets: $C_1, C_2, \ldots, C_k$, and L is partitioned into subsets: $L_1, L_2, \ldots, L_k$. The subgraphs over $C_1 \cup L_1, C_2 \cup L_2, \ldots,$ and $C_k \cup L_k$ are identical to each other. A subgraph over $C_i \cup L_i$ is defined as follows: all the clients are connected to a common facility location f_i^* with a unit weight edge. Each client is connected to a distinct set of k facility locations with an edge of weight $(1 - \delta)$. We denote this set by $T(x)$ for a client $x \in C_i$. Figure 3.1 shows this subgraph. Lastly, all pairs of subgraphs

$C_i \cup L_i$ and $C_j \cup L_j$ are connected with an edge (f_i^*, f_j^*) of weight $\Delta \gg |C|$. This completes the construction of the bad instance.

Consider the unconstrained outlier k-service problem with $m = 0$ (no outliers allowed). It is easy to see that $\mathbb{O} = \{C_1, C_2, \ldots, C_k\}$ is the optimal clustering for this instance. The optimal cost of a cluster C_i is service-cost$(f_i^*, C_i) = |C_i|$, and the optimal cost of the entire instance is OPT$(L, C, k) = \sum_i |C_i| = |C|$.

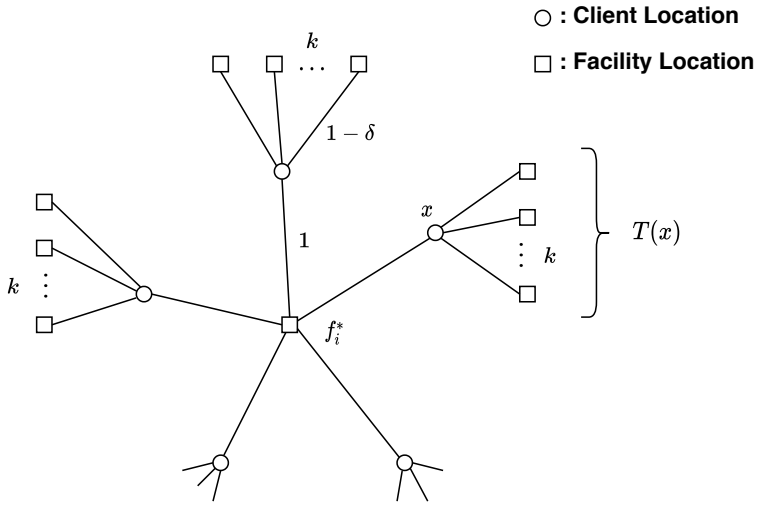

Figure 3.1: An undirected weighted subgraph on $C_i \cup L_i$.

Now, we show that the list $\mathcal{L}$ produced by the algorithm `List-Outlier-k-Service` does not contain a center-set that could provide better than $(3^z - \delta')$-approximation to $\mathbb{O}$. To show this, we examine every center-set in the list $\mathcal{L}$ produced by `List-Outlier-k-Service`. Note that the set T obtained in line (8) of the algorithm does not contain any optimal facility location f_i^* because f_i^* does not belong to $T(x)$. Therefore, no center set in the list contains any of the optimal facility locations $\{f_1^*, ..., f_k^*\}$. Now we evaluate the clustering cost corresponding to every center set in the list. Let $F = \{f_1, f_2, \ldots, f_k\}$ be a center-set in the list. We have two possibilities for the facilities in F. The first possibility is that at least two facilities in F belong to the same subgraph $C_i \cup L_i$. In this case, the cost of the clustering would be service-cost$(F, \mathbb{O}) > \Delta \gg$ OPT(L, C, k). So, in this case, F gives an unbounded clustering

cost. Let us consider the second possibility that all the facilities in F belong to different sub-graphs. Without loss of generality, we assume that $f_i \in L_i$. Since f_i can not be the optimal facility location, we can further assume that $f_i \in T(x)$ for some $x \in C_i$. The cost of a cluster in this case is service-cost$(f_i, C_i) = (3-\delta)^z(|C_i|-1)+(1-\delta)^z > (3-\delta)^z(|C_i|-1)$. Hence, the overall cost of the instance is $\Phi(F, \mathbb{O}) > (3-\delta)^z \cdot (|C|-k) \geq (3-\delta)^z \cdot |C| - 3^z\, k \geq (3^z - \delta') \cdot |C|$, for $\delta' = 3^{z-1} \cdot z\, \delta + \frac{3^z k}{|C|}$. Therefore, the list does not contain any center set that can provide better than $(3^z - \delta')$ approximation guarantee to the optimal clustering $\mathbb{O}$.

Theorem 37. *For any $0 < \delta' \leq 1$, there are instances of the k-service problem for which the algorithm* `List-Outlier-k-Service` *does not provide better than $(3^z - \delta')$ approximation guarantee.*

Now, let us examine the same bad instance when we have the flexibility to open a facility at a client location. In this case, we have a third possibility that $F = \{f_1, f_2, \ldots, f_k\}$ such that f_i is some client location in C_i. The cost of a cluster in this case is service-cost$(f_i, C_i) = 2^z \cdot (|C_i|-1)$ and the overall cost the instance is $\Phi(F, \mathbb{O}) = 2^z \cdot |C| - 2^z \cdot k = (2^z - \delta') \cdot |C|$, for $\delta' = 2^z \cdot k/|C|$. So for the special case $C \subseteq L$, we obtain the following theorem.

Theorem 38. *For any $0 < \delta' \leq 1$, there are instances of the k-service problem (with $C \subseteq L$), for which the algorithm* `List-Outlier-k-Service` *does not provide better than $(2^z - \delta')$ approximation guarantee.*

3.8 Streaming Algorithms

In this section, we convert the algorithm of the constrained outlier k-service problem to a constant-pass log-space streaming algorithm. The offline algorithm for the constrained outlier k-service problem has two main components: the list outlier k-service algorithm and the outlier partition algorithm. The list outlier k-service procedure is common to all constrained versions

of the problem. However, the outlier partition algorithm differs for different constrained versions. We convert the algorithm `List-Outlier-k-Service` to a streaming algorithm in the following manner:

Conversion of `List-Outlier-k-Service` to Streaming Algorithm:

1. In the first pass, we run a streaming α-approximation algorithm for the instance $(C, C, k + m, d, z)$. For this, we use the streaming algorithm of Braverman *et al.* [32]. The algorithm gives a constant-approximation with the space complexity of $O((k + m) \cdot \log n)$.
2. In the second pass, we perform the D^z-sampling step using the *reservoir sampling* technique [135].
3. In the third pass, we find the k-closest facility locations for every point in M.

This gives us the following result:

Lemma 19. *There is a 3-pass streaming algorithm for the list outlier k-service problem. The algorithm outputs a list of size $f(k, m, \varepsilon, z) \cdot \log n$. Moreover, the running time of the algorithm is $O(n \cdot f(k, m, \varepsilon))$ and space complexity is $f(k, m, \varepsilon, z) \cdot \log n$, where $f(k, m, \varepsilon, z) = ((k + m)/\varepsilon)^{O(kz^2)}$.*

Now, we need to design the partition algorithms for the constrained clustering problems in the streaming setting. For the chromatic k-service problems, there does not exist any deterministic log-space streaming algorithm (see Section 4.5 of [87]). In Section 3.9.4 we show that this impossibility result also holds for the strongly-private k-service problem. For the ℓ-diversity and fair k-service problems, we neither know any log-space streaming algorithm nor any impossibility result. For the remaining constrained clustering problems mentioned in Table 3.1,

we design the streaming partition algorithms for their outlier and non-outlier versions in Section 3.9. Together with Lemma 19, we obtain the following results for the constrained clustering problems:

Theorem 39. *There is a 6-pass streaming algorithm for the outlier version of the r-gather, r-capacity, and balanced k-service problems that gives a $(3^z + \varepsilon)$-approximation guarantee. The algorithm has the space complexity of $O(f(k, m, \varepsilon, z) \cdot \log n)$ and the running time of $O(f(k, m, \varepsilon, z) \cdot n^{O(1)})$, where $f(k, m, \varepsilon, z) = ((k + m)/\varepsilon)^{O(kz^2)}$. For the special case when $C \subseteq L$, the algorithm gives $(2^z + \varepsilon)$-approximation guarantee.*

Corollary 14. *There is a 6-pass streaming algorithm for the r-gather, r-capacity, and balanced k-service problems that gives a $(3^z + \varepsilon)$-approximation guarantee. The algorithm has the space complexity of $O(f(k, \varepsilon, z) \cdot \log n)$ and the running time of $O(f(k, \varepsilon, z) \cdot n^{O(1)})$, where $f(k, \varepsilon, z) = (k/\varepsilon)^{O(kz^2)}$. For the special case when $C \subseteq L$, the algorithm gives $(2^z + \varepsilon)$-approximation guarantee.*

Theorem 40. *There is a 4-pass streaming algorithm for the fault-tolerant, ordered-weighted-average, and uncertain k-service problems that gives a $(3^z + \varepsilon)$-approximation guarantee. The algorithm has the space complexity of $O(f(k, \varepsilon, z) \cdot \log n)$ and the running time of $O(f(k, \varepsilon, z) \cdot n)$, where $f(k, \varepsilon, z) = (k/\varepsilon)^{O(kz^2)}$. For the special case when $C \subseteq L$, the algorithm gives $(2^z + \varepsilon)$-approximation guarantee.*

Theorem 41. *There is a 5-pass streaming algorithm for the outlier version of the fault-tolerant, ordered-weighted-average, and uncertain k-service problems that gives a $(3^z + \varepsilon)$-approximation guarantee. The algorithm has the space complexity of $O(f(k, m, \varepsilon, z) \cdot \log n)$ and the running time of $O(f(k, m, \varepsilon, z) \cdot n)$, where $f(k, m, \varepsilon, z) = ((k+m)/\varepsilon)^{O(kz^2)}$. For the special case when $C \subseteq L$, the algorithm gives $(2^z + \varepsilon)$-approximation guarantee.*

3.9 Partition Algorithms

In this section, we design partition algorithms for the outlier and non-outlier versions of the constrained k-service problems mentioned in Table 3.1. We also convert each of these algorithms to log-space streaming algorithms or show that such an algorithm does not exist. In the streaming setting, the clients in C arrive as a stream of points, and the partition algorithm has to assign each client to a facility location in given set F such that the clustering constraints are satisfied.

3.9.1 r-gather/r-capacity/balanced k-service problem

We design a partition algorithm for the balanced k-service problem. The same partition algorithm is also used for the r-gather and r-capacity k-service problems since they are the special versions of the balanced k-service problem. Let $\mathcal{I} = (L, C, k, d, z, m, \ell_1, \ldots, \ell_k, r_1, \ldots, r_k)$ be any arbitrary instance of the balanced outlier k-service problem. Let $F = \{f_1, \ldots, f_k\}$ be a given set of facility locations. Let Z be an optimal set of outliers and $\mathbb{O} = \{O_1, \ldots, O_k\}$ be an optimal partitioning of $C \setminus Z$ with respect to F that satisfies the balanced clustering constraints. Since we do not know which cluster is assigned to which facility location in F, we exhaustively consider all k^k possibilities. Without loss of generality, we assume that cluster O_i is assigned to facility location f_i. Now we reduce the partitioning problem to a min-cost circulation problem on a flow network $G = (V, E)$. The vertex set V is composed of the following vertices:

1. A source vertex s, sink vertex t, and vertex o.

2. A vertex set V_C corresponding to the client set C. In other words, for every client $x \in C$ there is a vertex v_x in V_C.

3. A vertex set V_F that corresponds to the facility set F. In other words, for every facility $f_i \in F$ there is a vertex $v_{f_i} \in V_F$.

Furthermore, the edge set E is composed of the following edges, each with lower and upper bound flow constraints:

1. There is a directed edge (s, o) with a lower bound flow of $|C| - m$ and an upper bound flow of $|C|$. This edge ensures that at least $|C| - m$ clients are assigned to F. The cost of the edge is 0.

2. For every vertex $v_x \in V_C$, there is a directed edge (o, v_x) with upper bound flow of 1 and lower bound flow of 0. These edges ensure that a client is assigned to at most one facility in F. The cost of each edge is 0.

3. For every vertex $v_x \in V_C$ and vertex $v_{f_i} \in V_F$, there is a directed edge (v_x, v_{f_i}) with upper bound flow of 1 and lower bound flow of 0. The cost of each edge (v_x, v_{f_i}) is proportional to the distance $d(x, f_i)$.

4. For every $v_{f_i} \in V_F$, there is a directed edge (v_{f_i}, t) with lower bound flow ℓ_i and upper bound flow r_i. These edges ensures that each partition $O_i \in \{O_1 \ldots, O_k\}$ satisfies the constraint: $\ell_i \leq |O_i| \leq r_i$. The cost of each edge is 0.

It is easy to see that a feasible integral flow through the network G corresponds to an assignment of at least $|C| - m$ clients in C to F that satisfies the clustering constraints. The minimum cost circulation problem on G can be solved in polynomial time using the algorithms in [76, 132, 23]. Since we run this algorithm for k^k possible guesses of assigning O_i's to f_i's, the overall running time of the partition algorithm is $k^k \cdot n^{O(1)}$, where $n = |C \cup L|$. Formally, we state the result as follows:

Lemma 20. *There is a $k^{O(k)} \cdot n^{O(1)}$ time partition algorithm for the outlier version of the balanced k-service problem, where $n = |C \cup L|$.*

To convert the partition algorithm to a streaming algorithm, we use the following result:

Definition 38. *Let z be any positive real number. Let $G = (V_F, V_C, E)$ be an edge-weighted bipartite graph with partitions V_F and V_C. Further, we associate a number n_v with each vertex $v \in V_C$. We say that G* approximately represents *the pair (F, C), where F is a set of k facility locations and C is the client set if the following conditions are satisfied:*

- *The set $V_F = F$. Each client $x \in C$ is mapped to a unique vertex v in V_C – call this vertex $\phi(x)$. Further n_v is equal to $|\phi^{-1}(v)|$.*
- *For each client $x \in C$ and facility $f \in F$, the weight of the edge $(\phi(x), f)$ in G is within $(1 \pm \varepsilon)$ of $d(x, f)^z$.*

Theorem 42. *Given a pair (F, C), there is a single pass streaming algorithm that builds a bipartite graph G that approximately represents this pair. The space used by this algorithm (which includes the size of G) is $O\left(k^2 \cdot 2^k \cdot (k + \log n + \log \Delta) \cdot \left(k \cdot 6^k \cdot \log n + k^k \cdot \log^k(\frac{1}{\varepsilon})\right)\right)$, where Δ is the aspect ratio defined as $\Delta = \frac{\max_{x \in C, f \in F} d(x,f)}{\min_{x \in C \setminus F, f \in F} d(x,f)}$, $n = |C \cup F|$, and $k = |F|$. Further, the dependence on $\log \Delta$ can be removed by adding one more pass to the algorithm.*

Before proving Theorem 42, let us see how it gives streaming partition algorithm. We again formulate the flow formulation in an analogous manner – the only change is that the subgraph on the vertex set V_C and V_F is replaced with the approximate graph representation using Theorem 42 and the upper bound capacity on edge (o, v) is changed to n_v for every vertex $v \in V_C$. It is easy to see that an optimal cost flow in the flow network corresponds to $(1+\varepsilon)$-approximation to the cost of optimal partitioning. After finding an optimal flow, the algorithm must assign the clients to the facility locations in F. For this, the algorithm makes one more pass through the client set and assigns a client x to that facility location f to which a non-zero flow is going via edge $(\phi(x), v_f)$. After assigning a client to a facility location, the algorithm reduces the flow by one unit on appropriate edges. In total, the algorithm makes three passes through the client set — two passes to create an approximate graph representation of (F, C) and one pass to assign clients to the facility locations in F. Formally, we state the result as follows:

Lemma 21. *There is a 3 pass streaming partition algorithm for the outlier version of the balanced k-service problem. The algorithm outputs a $(1+\varepsilon)$-approximate clustering with respect to the optimal partitioning. The space used by the algorithm is $O(f(k,\varepsilon)\cdot \log n)$ where $f(k,\varepsilon) = k^{O(k)}\cdot \log^k(1/\varepsilon)$ and $n = |C \cup L|$.*

Now, we prove Theorem 42.

Proof of Theorem 42

For each center $f \in F$, we define a set of buckets B_f as follows: the bucket $b(f,i)$ corresponds to the values $[(1+\varepsilon)^i, (1+\varepsilon)^{i+1})$. Let $d_{\min}$ and $d_{\max}$ denote the minimum and the maximum between a pair of points in $C \setminus F$ and F, respectively (so $\Delta = d_{\max}/d_{\min}$). We define the buckets $b(f,i)$ for $i = \log_{1+\varepsilon} d_{\min}, \ldots, \log_{1+\varepsilon} d_{\max}$. As above, B_f denotes the collection of buckets corresponding to f. Clearly, $|B_f| = O(\frac{\log \Delta}{\epsilon})$.

Now we consider the set $B_{f_1} \times B_{f_2} \ldots \times B_{f_k}$ – an element of this set is called a *hyperbucket.* In other words, a hyperbucket is a $k-$tuple $(b(f_1,i_1),\ldots,b(f_k,i_k))$. Each point $p \in C$ can be mapped to a hyperbucket in the natural manner – define $\phi(p)$ to be the hyperbucket $(b(f_1,i_1),\ldots,b(f_k,i_k))$, where i_j is such that $d(p,f_j) \in [(1+\varepsilon)^{i_j}, (1+\varepsilon)^{i_j+1})$ for $j = 1,\ldots,k$.

Call a hyper-bucket $\mathbf{b}$ to be empty if $\phi^{-1}(\mathbf{b})$ is empty. We now count the number of non-empty hyper-buckets. For a center f and index i, call the bucket $b(f,i)$ **interesting** if there is another center f' such that $[(1+\varepsilon)^i, (1+\varepsilon)^{i+1}) \cap [\varepsilon \cdot ||f-f'||, ||f-f'||/\varepsilon]$ is non-empty. Call a hyperbucket $(b(f_1,i_1),\ldots,b(f_k,i_k))$ interesting if *all* the buckets $b(f_j,i_j)$ in it are interesting. We first count the number of interesting hyperbuckets:

Claim 1. *The number of interesting hyperbuckets is $O\left(k^k \cdot \log^k(\frac{1}{\varepsilon})\right)$.*

Proof. For a fixed center f, the number of interesting buckets $b(f,i)$ is $O(k\log(\frac{1}{\varepsilon}))$. Since all

the buckets in a hyperbucket needs to be interesting, the result follows. □

We now count the number of non-interesting hyperbuckets.

Claim 2. *Let $b(f,i)$ be a non-interesting bucket for some center f and index i. Then the number of non-empty non-interesting hyperbuckets containing $b(f,i)$ is $O(6^k)$.*

Proof. Consider such a hyperbucket $\mathbf{b}$ containing $b(f,i)$. Let p be a point such that $\phi(p)$ is $\mathbf{b}$. Let f' be a center other than f. Two cases arise:

- $(1+\varepsilon)^{i+1} \leq \varepsilon d(f,f')$: In this case,

$$d(p,f') \in d(f,f') \pm d(f,p) \in (1 \pm \varepsilon) \cdot d(f,f').$$

  Therefore, there are at most 3 choices for the index i' such that $b(f',i')$ is one of the coordinates of $\mathbf{p}$.

- $d(f,f') \leq \varepsilon(1+\varepsilon)^i$: Here,

$$d(p,f') \in d(f,f') \pm d(f,p) \in (1 \pm \varepsilon) \cdot d(f,p).$$

  Again, there are at most 3 choices for i' as above.

From the above argument, it is clear that the number of non-empty non-interesting hyperbuckets $\mathbf{b}$ containing $b(f,i)$ is $O(6^k)$. This proves the claim. □ □

We can now count the number of non-empty non-interesting hyperbuckets. Consider such a bucket $\mathbf{b}$. There must be a coordinate $b(f,i)$ in it which is non-interesting – there are $O(k \log \Delta)$ choices for $b(f,i)$. For each such choice, the above claim shows that there are $O(6^k)$ choices for

the remaining coordinates of $\mathbf{b}$. Thus, we see that the total number of non-empty hyperbuckets is $O(k \cdot 6^k \log \Delta + k^k \cdot \log^k(\frac{1}{\varepsilon}))$.

We can describe the streaming algorithm. The bipartite graph will have all the non-empty hyperbuckets on the right side and the k centers F on the left side. The length of an edge between a hyperbucket $(b(f_1, i_1), \ldots, b(f_k, i_k))$ and a center f_j will be $(1+\varepsilon)^{z \cdot i_j}$. Initially, the right side of the bipartite graph will be empty (because all hyperbuckets are empty). Whenever a new point p is seen, the algorithm computes $\phi(p)$. If this hyperbucket is not present in the right side of the bipartite graph, we add a new vertex corresponding to it. The algorithm can also maintain the cardinality of $\phi^{-1}(\mathbf{b})$ for every non-empty bucket $\mathbf{b}$ (this will be stored in the variable n_v, where v is a vertex in the right side of the graph). The theorem now follows from the fact that the number of edges is equal to k times the number of non-empty hyperbuckets, and the space required to maintain n_v values is $\log n$ times the number of non-empty hyperbuckets. This proves that there is a single pass streaming algorithm that builds graph G. The space used by this algorithm (which includes the size of G) is $O\left((k + \log n) \cdot \left(k \cdot 6^k \cdot \log \Delta + k^k \cdot \log^k(\frac{1}{\varepsilon})\right)\right)$.

Note that the space required depends on $\log \Delta$, whereas we would really like it to depend on $\log n$ instead. We discuss this next.

Removing the dependence on Δ: Let D denote the set of all pair-wise distances between the centers (so $|D| \leq k^2$). The following is the key observation:

Lemma 22. *Let p be a point in C, and f be the closest center to it. For any center $f' \in F$, $d(p, f')$ lies in the range $[u/4, 4u]$ for some $u \in D \cup \{d(p, f)\}$.*

Proof. Assume $d(p, f') \geq 4d(p, f)$, otherwise the lemma is already true. Then

$$d(f, f') \geq d(p, f') - d(p, f) \geq 3d(p, f).$$

Therefore,

$$d(p, f') \leq d(f, f') + d(p, f) \leq \frac{4d(f, f')}{3},$$

and

$$d(p, f') \geq d(f, f') - d(p, f) \geq \frac{2d(f, f')}{3}.$$

□ □

We fix a solution to the partition problem (of course, the algorithm does not know it, but it will help in the algorithm description). For a point x, let d_x denote the distance from x to the closest center, Let $d^\star$ be the maximum over all points x of d_x. We do know $d^\star$, we can find it by performing a pass over the data.

Now suppose we guess the maximum $u \in D$ such that any point $p \in C$ which is not assigned to a center in F within distance $4d_x$ is assigned to a center f' such that $d(p, f')$ lies in the range $[u/4, 4u]$ (the lemma above ensures that such a value u exists). [5] If $d^\star$ happens to be larger than u, we update u to $d^\star$. We know that the cost of the solution is at least $\Omega(u^z)$.

Now, we contract all distances in the metric which are smaller than u/n^2 – again we cannot do this directly in the streaming manner, but whenever a point p arrives, we will view all distances to centers which are less than u/n^2 to be 0. Since the optimal cost is $\Omega(u^z)$, this distance contraction affects the optimal value by at most a factor of $(1 + 1/n)$. So now, all non-zero distances between a point p and a center $f \in F$ such that p can be potentially assigned to f lie in the range $[u/n^2, 4u]$. Thus Δ becomes polynomially bounded. However one issue remains – each point p can only be assigned to a center (other than it's nearest center) which is at most $4u$ distance away. We need to incorporate this fact in the graph structure as well. For each point p, let $F(p)$ be the set of centers to which it can be assigned (these are the centers which are at most $4u$ distance away, or the center closest to p). Note that there are 2^k choices for $F(p)$.

[5] In the algorithm implementation, we will run this for all possible values of $u \in D$. This will increase the space complexity by a factor of k^2.

We modify the construction of G. Recall that the right side of G had one vertex for every hyperbucket $\mathbf{b}$. Now we will have one vertex for every pair $(\mathbf{b}, S)$, where S is a subset of F. If $\phi(p)$ is the hyperbucket $\mathbf{b}$, then we assign p to the pair $(\mathbf{b}, F(p))$. The result of this construction is that the $\log \Delta$ can now be replaced with $\log n$ but at the cost of multiplying the overall space requirement by a factor of $k^2 \cdot 2^k$ and adding one more pass. This proves the Theorem 42.

3.9.2 Fault-tolerant k-service problem

The fault-tolerant k-service problem does not satisfy Definition 29 of the constrained k-service problem. Therefore, the techniques we developed for the constrained clustering problems may not work for the fault-tolerant k-service problem. However, Ding and Xu [72] showed that the fault-tolerant k-service problem can be reduced to the chromatic k-service problem. Since the chromatic k-service problem satisfy Definition 29 of the constrained k-service problem, we can solve the fault-tolerant k-service problem using the same set of techniques. In the fault-tolerant k-service problem, we are given a positive integer $\ell_x \leq k$ for every client $x \in C$ (see definition 5 in Table 3.1). The reduction to the chromatic k-service problem goes as follows: we create ℓ_x copies for each client $x \in C$ and assign each copy the same color. For two clients $x, y \in C$ if $x \neq y$, their copies have different colors. Ding and Xu [72] showed that an α-approximation for the reduced chromatic k-service problem is also an α-approximation to the fault-tolerant k-service problem, and vice-versa for any $\alpha \geq 1$. However, the partition algorithm for the fault-tolerant k-service problem is simple and does not require a reduction to the chromatic k-service problem. Given a facility set F and a client $x \in C$, the algorithm simply assigns x to ℓ_x closest facility locations in F. Computing k closest facility location from x is a trivial step. Formally, we state the result as follows:

Lemma 23. *There is a single pass streaming partition algorithm for the fault-tolerant k-service problem with $O(k \cdot |C|)$ running time, where C is the client set, and k is the size of the center set. The algorithm only stores $O(k)$ points in the memory.*

The extension to the outlier setting is also simple: the algorithm identifies the m clients as outliers that have the highest assignment cost. For this, the algorithm requires additional $\Omega(m)$ space and $O(m \log m)$ time (assuming the max-heap implementation). Note that in the streaming setting, the algorithm can not assign a client to a facility until it identifies it as a non-outlier. Therefore, in the streaming setting, the set of m outliers are identified in the first pass, and non-outliers are assigned to facility locations in the second pass. Formally, we state the result as follows:

Lemma 24. *There is a two-pass streaming partition algorithm for the outlier version of the fault-tolerant k-service problem with running time $O(k \cdot |C| + m \log m)$. Moreover, the algorithm stores only $O(k + m)$ points in the memory.*

Note that the streaming version of the list k-service algorithm is not affected due to the reduction of the fault-tolerant k-service problem to the chromatic k-service problem. The reason is that the algorithm creates ℓ_x copies for a client x when it arrives in the stream. Therefore, the streaming algorithm for the list outlier k-service problem stays the same.

3.9.3 Ordered-weighted-average (OWA) k-service problem

OWA k-service problem does not satisfy Definition 29 of the constrained k-service problem. Therefore, the techniques developed for the constrained clustering problems do not directly work for the OWA k-service problem. However, Byrka *et al.* [36] gave a reduction from the OWA k-service problem to the fault-tolerant k-service problem with client multiplicities. The client multiplicity means that each client $x \in C$ has multiple finite copies in the instance. In OWA k-service problem, we are given a vector $(w_1, \ldots, w_k)$ of non-increasing weights (see definition 6 in Table 3.1). The reduction to the fault-tolerant k-service problem goes as follows: each client $x \in C$ is replaced with k clients: $x_1, \ldots, x_k$ such that $\ell_{x_i} = i$ and multiplicity of each x_i is some function of $w_1, \ldots, w_k$. Byrka *et al.* [36] showed that an α-approximation for the reduced fault-tolerant k-service instance is also an α-approximation to the OWA k-

service instance, and vice-versa for any $\alpha \geq 1$. However, the partition algorithm for the OWA k-service problem is simple and does not require a reduction to the fault-tolerant k-service problem. Given a facility set F and a client $x \in C$, the algorithm computes the assignment cost for x: $\sum_{x \in C} \sum_{j=1}^{k} w_j \cdot \left(\overline{d}_j(x)\right)^z$ such that $\left(\overline{d}_1(x), ..., \overline{d}_k(x)\right)$ is a non-decreasing ordering of $\big(d(x, f_1), ..., d(x, f_k)\big)$. Formally, we state the result as follows:

Lemma 25. *There is a single pass streaming partition algorithm for the ordered-weighted-average k-service problem with $O(k \log k \cdot |C|)$ running time, where C is the client set, and k is the size of the center set. The algorithm stores only $O(k)$ points in the memory.*

The extension to the outlier setting is also simple: the algorithm identifies the m clients as outliers that have the highest assignment cost. For this, the algorithm requires additional $\Omega(m)$ space and $O(m \log m)$ time (assuming the max-heap implementation). Note that in the streaming setting, the algorithm can not assign a client to a facility until it identifies it as a non-outlier. Therefore, in the streaming setting, the set of m outliers are identified in the first pass, and non-outliers are assigned to facility locations in the second pass. Formally, we state the result as follows:

Lemma 26. *There is a two-pass streaming partition algorithm for the outlier version of the ordered-weighted-average k-service problem with running time $O(k \log k \cdot |C| + m \log m)$. Moreover, the algorithm stores only $O(k + m)$ points in the memory.*

Note that the streaming version of the list k-service algorithm is not affected due to the reduction of the OWA k-service problem to the fault-tolerant k-service problem. The reason is that the algorithm creates k copies for a client x when it arrives in the stream. Therefore, the streaming algorithm for the list outlier k-service problem stays the same. However, it is possible that after reduction, the multiplicity of a client is exponential in $|C|$. In that case, in Algorithm 3.1, we must replace the D^z sampling with weighted D^z sampling, i.e., a point with weight w_x is sampled with the probability: $\dfrac{\min_{f \in F} \left\{d(f, x)^z \cdot w_x\right\}}{\sum_{y \in C} \min_{f \in F} \left\{d(f, y)^z \cdot w_y\right\}}$. A trivial but important observation

is that this new algorithm is equivalent to the old algorithm with w_x copies created for each client in C. Therefore the same analysis follows for the new list k-service algorithm, and it has a polynomial running time.

3.9.4 Chromatic and strongly private k-service problems

We combine the discussion of the chromatic and strongly private k-service problems by defining the *hybrid k-service* problem that encapsulates both problems.

Definition 39 (Hybrid k-Service Problem)**.** *Given a partitioning $C_1, \ldots, C_\omega$ of the client set C, and a set of integers: $\{\alpha_1, \ldots, \alpha_\omega, \beta_1, \ldots, \beta_\omega\}$, find a clustering $\mathbb{O} = \{O_1, ..., O_k\}$ with minimum $\Phi^*(\mathbb{O})$ that satisfies the constraint: $\beta_j \leq |C_j \cap O_i| \leq \alpha_j$ for every $i \in [k]$ and $j \in [\omega]$.*

The hybrid k-service problem encapsulates the chromatic and strongly-private problems as described below:

- For $\alpha_j = |C|$, the hybrid k-service problem is equivalent to the strongly private k-service problem.
- For $\beta_j = 0$ and $\alpha_j = 1$, the hybrid k-service problem is equivalent to the chromatic k-service problem.

Let $F = \{f_1, \ldots, f_k\}$ be a given set of facility locations. A partition algorithm for the outlier version of hybrid k-service problem finds an outlier set Z of size at most m and partitioning $\mathbb{O} = \{O_1, \ldots, O_k\}$ of $C \setminus Z$ that minimizes the objective function $\Phi(F, \mathbb{O})$ and satisfies the constraint: $\beta_j \leq |C_j \cap O_i| \leq \alpha_j$ for every $i \in [k]$ and $j \in [\omega]$. Now we reduce the partitioning problem to a min-cost circulation problem on a flow network $G = (V, E)$. The flow network is shown in Figure 3.2.

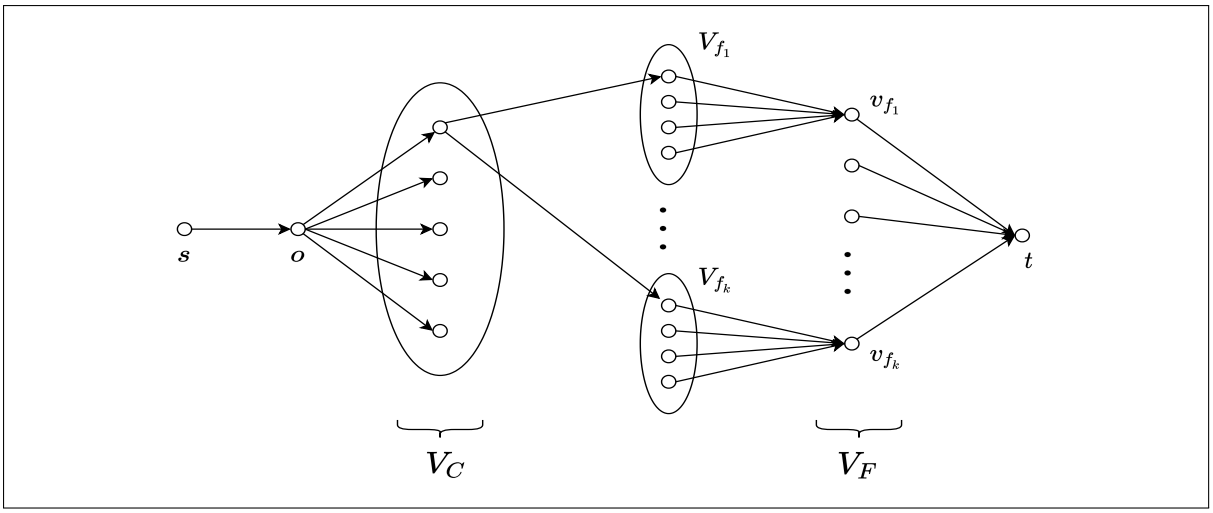

Figure 3.2: The flow network $G = (V, E)$ that is used in the partition algorithm of the hybrid k-service problem.

The vertex set V has the following composition:

1. A source vertex s, sink vertex t, and vertex o.

2. A vertex set V_C corresponding to the client set C. In other words, for every client $x \in C$ there is a vertex v_x in V_C.

3. A vertex set V_F that corresponds to the facility set F. In other words, for every facility $f_i \in F$ there is a vertex $v_{f_i} \in V_F$.

4. The vertex sets: $V_{f_1}, \ldots, V_{f_k}$. A vertex set V_{f_i} is defined as $\{v_{f_i,1}, \ldots, v_{f_i,\omega}\}$ such that a vertex $v_{f_i,j}$ corresponds to a facility $f_i \in F$ and a color class $j \in [\omega]$.

Furthermore, the edge set E is defined with the lower and upper bound flow constraints as follows:

1. There is a directed edge (s, o) with a lower bound flow of $|C| - m$ and an upper bound flow of $|C|$. This edge ensures that at least $|C| - m$ clients are assigned to F. The cost of the edge is 0.

2. For every vertex $v_x \in V_C$, there is a directed edge (o, v_x) with upper bound flow of 1 and lower bound flow of 0. These edges ensure that a client is assigned at most one facility in F. The cost of each edge is 0.

3. For every vertex $v_x \in V_C$ and vertex $v_{f_i,j} \in V_{f_i}$, there is a directed edge $(v_x, v_{f_i,j})$ if and only if the client x belongs to color class C_j. These edges have the upper bound flow of 1 and the lower bound flow of 0. The cost of each edge is $d(x, f_i)^z$.

4. For every $i \in [k]$ and $j \in [\omega]$, there is a directed edge $(v_{f_i,j}, v_{f_i})$ with lower bound flow of β_j and upper bound flow of α_j. These edges ensures that for every color class $j \in [\omega]$ and every facility $f_i \in F$, the constraint $\beta_j \leq |O_i \cap C_j| \leq \alpha_j$ is satisfied. The cost of each edge is 0.

5. For every $i \in [k]$, there is a directed edge (v_{f_i}, t) with lower bound flow 0 and upper bound flow $|C|$. The cost of each edge is 0.

It is easy to see that a feasible integral flow through the network G corresponds to an assignment of at least $|C| - m$ clients in C to F that satisfies the constraints of the hybrid k-service problem. The minimum cost circulation problem on G can be solved in polynomial time using the algorithms in [76, 132, 23]. Thus we get a polynomial time partition algorithm for the hybrid k-service problem. Formally, we state the result as follows:

Lemma 27. *There is a polynomial time partition algorithm for the outlier versions of chromatic and strongly private k-service problems.*

In the streaming setting, Khanna and Assadi (see Section 4.5 of [87]) showed that there does not exist any $o(n)$-space streaming algorithm for the chromatic k-service problem. The authors proved this result for a bad instance of the chromatic k-service problem with $m = n/2$ colors and two clusters. All clients in the instance are co-located, and there are precisely two clients

with the same color (assuming that n is even). Note that for this instance the chromatic k-service problem is equivalent to the strongly private k-service problem with $\beta_j = 1$ and $\alpha_j = |C|$. Therefore, the impossibility result also holds for the strongly-private k-service problem. Formally, we state the result as follows:

Lemma 28. *There does not exist any $o(n)$-space streaming algorithm for the outlier or non-outlier version of chromatic and strongly-private k-service problems.*

3.9.5 Uncertain k-service problem

The uncertain k-service problem does not satisfy Definition 29 of the constrained k-service problem. Therefore, the techniques developed for the constrained clustering problems may not work for the uncertain k-service problem. However, Ding and Xu [72] showed that the uncertain k-service problem can be equivalently stated as the constrained k-service problem on the weighted point set $\bigcup_{x \in C} D_x$ with the constraint that all points in D_x to be clustered in the same cluster. For every point $x_i \in D_x$, the weight of the point is t_x^i. Now the partition algorithm for the problem is simple: given a facility set F and a client $x \in C$, the algorithm assigns x to a facility f that minimizes the assignment cost $d(x, f)^z = \sum_{i=1}^{h} t_x^i \cdot d(x_i, f)^z$. In the streaming setting, we assume that for each client $x \in C$, the points in set D_x arrive as a contiguous stream of points. Then it is trivial to design a partition algorithm that makes a single pass and has running time $O(h \cdot k \cdot |C|)$.

Lemma 29. *There is a single pass streaming partition algorithm for the uncertain k-service problem with running time $O(h \cdot k \cdot |C|)$. The algorithm stores only $O(k)$ points in the memory.*

In the outlier setting, the algorithm identifies the m clients as outliers that have the highest assignment cost. For this, the algorithm requires additional $\Omega(m)$ space and $O(m \log m)$ running time (assuming max-heap implementation). Note that in the streaming setting, the algorithm can not assign a client to a facility until it is declared as a non-outlier. Therefore, in the streaming setting, the set of m outliers are identified in the first pass, and the non-outlier clients are

assigned to the facility locations in the second pass. Formally, we state the result as follows:

Lemma 30. *There is a two-pass streaming partition algorithm for the outlier version of the uncertain k-service problem with running time $O(h \cdot k \cdot |C| + m \log m)$. The algorithm stores only $O(k + m)$ points in the memory.*

3.9.6 ℓ-diversity and fair k-service problems

Bandyapadhyay *et al.* [21] noted that the ℓ-diversity clustering problem is a special case of the fair clustering problem. In other words, if the fair clustering problem has disjoint color classes, and $\alpha_j = 1/\ell$ and $\beta_j = 0$ for every color class $C_j \in \{C_1, \ldots, C_\omega\}$, then the problem is equivalent to the ℓ-diversity clustering problem. Therefore, designing a partition algorithm for the fair k-service problem is sufficient.

For the fair k-*service* problem (without outliers), Bandyapadhyay *et al.* [21] designed an FPT time partition algorithm. It is simple to extend their algorithm to the outlier setting. For the sake of completeness, we describe the complete algorithm. Let $\mathcal{I} = (L, C, k, d, z, m, C_1, \ldots, C_\omega, \alpha, \beta)$ be any instance of the fair outlier k-service problem. Recall that the sets $C_1, \ldots, C_\omega$ are the color classes, each of which is a subset of the client set C. Moreover, any two color classes can overlap with each other. In other words, a client in C might belong to a different color class. To simplify the problem, we partition the client set into Γ disjoint groups: $P_1, \ldots, P_\Gamma$ such that the points belonging to the same group P_i belong to the same set of colored classes. In other words, if clients x and y belong to the same group P_i, then for every color class $C_t \in \{C_1, \ldots, C_\omega\}$, $x \in C_t$ if and only if $y \in C_t$. Also, if x and y belong to different groups P_i and P_j, respectively, then there exists a color class $C_t \in \{C_1, \ldots, C_\omega\}$ such that $x \in C_t$ and $y \notin C_t$, or $x \notin C_t$ and $y \in C_t$. Note that if the color classes are pair-wise disjoint, then Γ equals the number of color classes, i.e., $\Gamma = \omega$. Now, we design FPT time partition algorithm for the fair outlier k-service problem with running time $(k\Gamma)^{k\Gamma} \cdot n^{O(1)}$, where Γ is the number of distinct collection of color classes induced by the colors of clients. Formally, we state the result as follows:

Lemma 31. *For the fair outlier k-service problem, there is a $(k\Gamma)^{k\Gamma} \cdot n^{O(1)}$ time partition algorithm, where Γ is the number of distinct collections of color classes induced by the colors of clients.*

Proof. Let $F = \{f_1, \ldots, f_k\}$ be the given set of facility locations for which we want an optimal partitioning satisfying the fair outlier constraints. Let Z denote the optimal set of outliers and OPT denote the optimal k-service cost of assigning $C \setminus Z$ to F while satisfying the fair constraints. The assignment problem can be modeled as an integer linear program; however, solving an integer linear program is NP-hard in general. Therefore, we model the problem as a mixed integer linear program, which can be solved in FPT time parameterized by the number of integer variables. The following is a formal statement for the same:

Proposition 2 (Proposition 8.1 of [21])**.** *Given a real-valued matrix $A \in \mathbb{R}^{m \times d}$, vector $b \in \mathbb{R}^m$, vector $c \in \mathbb{R}^d$, and a positive integer $p \leq d$. There is an FPT time algorithm that finds a vector $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ that minimizes $c \cdot x$, and satisfies that $A \cdot x \leq b$ and $x_1, \ldots, x_p \in \mathbb{Z}$. The running time of the algorithm is $O(p^{2.5p+o(p)} d^4 B)$, and the space complexity is polynomial in B, where B is the bit size of the given instance.*

Now, we model the assignment problem as a mixed integer linear program (MILP), as follows:

Minimize	$\sum_{x \in C} \sum_{f \in F} g(x, f) \cdot d(x, f)^z$	
Constraints:	$\sum_{f \in F} g_{x,f} \leq 1$	for every client $x \in C$
$\sum_{x \in P_i} g_{x,f} = h_{f,i}$		for every group $P_i \in \{P_1, \ldots, P_\Gamma\}$ and facility $f \in F$
$\sum_{f \in F, x \in C} g_{x,f} \geq \lvert C \rvert - m$		for client set C and facility set F

$\sum_{x \in C_j} g_{x,f} \leq \alpha_j \cdot \sum_{x \in C} g_{x,f}$	for every facility $f \in F$ and color class $C_j \in \{C_1, \ldots, C_\omega\}$
$\sum_{x \in C_j} g_{x,f} \geq \beta_j \cdot \sum_{x \in C} g_{x,f}$	for every facility $f \in F$ and color class $C_j \in \{C_1, \ldots, C_\omega\}$
$0 \leq g_{x,f} \leq 1$	for every client $x \in C$ and facility $f \in F$
$h_{f,i} \in \mathbb{Z}_{\geq 0}$	for every group $P_i \in \{P_1, \ldots, P_\Gamma\}$ and facility $f \in F$

In the above MILP, for every client $x \in C$ and facility $f \in F$, there is a fractional variable $g_{x,f} \leq 1$ that denotes the fraction of client x assigned to facility f. Moreover, for every group $P_i \in \{P_1, \ldots, P_\Gamma\}$ and facility $f \in F$, there is an integer variable $h_{f,i}$ that denotes the total fraction of clients in P_i that is assigned to facility f. In other words, $h_{f,i} = \sum_{x \in P_i} g_{x,f}$. The third constraint of the MILP corresponds to the number of outliers being at most m. Lastly, the fourth and fifth constraints of MILP correspond to the fairness constraints.

We solve the above mixed integer linear program in time $O(k\Gamma)^{k\Gamma} \cdot n^{O(1)}$ as per Proposition 2. However, the obtained optimal solution contains fractional $g_{x,f}$ values. Next, we show that there always exists a solution with integral $g_{x,f}$ values that can be obtained in polynomial time. We reduce the problem to the minimum cost *circulation* problem on directed graphs. We construct a flow network $G = (V, E)$ with upper and lower bound flow requirements on every edge. The construction is shown in Figure 3.3.

The vertex set V contains a source vertex s, a sink vertex t, and an *outlier regulating vertex* o. We will describe the functioning of o shortly. The rest of the vertex set is partitioned into two sets: V_C and $V_{F,P}$. The vertex set V_C corresponds to the client set C. In other words, for every client $x \in C$ there is a vertex $v_x \in C$. The vertex set $V_{F,P}$ corresponds to facility set F and groups $P_1, \ldots, P_\Gamma$. In other words, for every facility $f \in F$ and group $P_i \in \{P_1, \ldots, P_\Gamma\}$, there is a vertex $v_{f,i}$ in $V_{F,P}$. Furthermore, the edge set E is composed as follows. There is

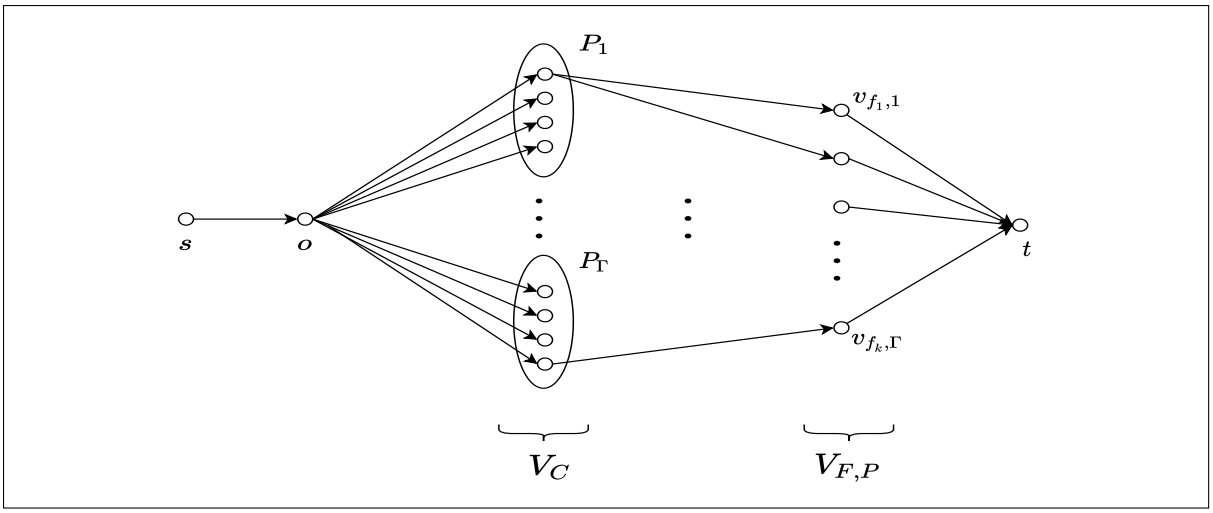

Figure 3.3: The flow network $G = (V, E)$ that is used by the partition algorithm of the fair outlier k-service problem.

an edge (s, o) with lower and upper bound flow requirement of exactly $\sum_{x\in C, f\in F} g_{x,f}$. Note that $\sum_{x\in C, f\in F} g_{x,f}$ is an integer since $\sum_{x\in C, f\in F} g_{x,f} = \sum_{i=1}^{\Gamma} \sum_{f\in F} h_{i,f}$ and $h_{i,f}$'s are integers. Also note that $\sum_{x\in C, f\in F} g_{x,f} \geq |C| - m$ as per Constraint 2 of the MILP. Therefore, it ensures that at least $|C| - m$ clients are assigned to F, and that's why we call the vertex o the outlier regulating vertex. Furthermore, the edge set E contains for every vertex $v_x \in C$, an edge (o, v_x) with lower bound flow requirement 0 and upper bound flow requirement 1. These edges ensure that a client is assigned to at most one facility in F. Furthermore, for every vertex $v_x \in V_C$ and $v_{f,i} \in V_{F,P}$, there is an edge $(v_x, v_{f,i})$ if and only if $x \in P_i$. Each edge has a lower bound flow requirement of 0 and an upper bound flow requirement of 1. Lastly, for every vertex $v_{f,i} \in V_{F,P}$, there is an edge $(v_{f,i}, t)$ with lower and upper bound flow requirement of exactly $h_{f,i}$. These edges ensure that exactly $h_{f,i}$ clients of P_i are assigned to any facility $f \in F$. It is easy to see that the constructed flow network G admit a feasible flow if we send a flow of $g_{x,f}$ through every edge $(v_x, v_{f,i})$. Moreover, this is the maximum flow that we can send through the network since the capacity of edge (s, o) is $\sum_{x\in C, f\in F} g_{x,f}$. Since the lower and upper bound requirements on every edge is an integer, a feasible integral flow exists through the network with the same cost. We find that flow in polynomial time using the algorithms in [132, 76, 23]. Let the new integral flow through $(v_x, v_{f,i})$ is $g'_{x,f}$. Then, we show that the values $g'_{x,f}$'s also satisfy the MILP

constraints. The constraints (1)-(3) of the MILP are trivially satisfied from the flow network. Also note that $\sum_{x\in P_i} g'_{x,f} = h_{f,i}$ due to the flow network. Therefore, the fair constraints (4) and (5) are satisfied since $\sum_{x\in C} g_{x,f} = \sum_{x\in C} g'_{x,f}$ and $\sum_{x\in C_j} g_{x,f} = \sum_{x\in C_j} g'_{x,f}$ for every facility $f \in F$ and every color class $C_j \in \{C_1, \ldots, C_\omega\}$. These two equalities follow from the following two sequences of equalities:

1.
$$\sum_{x\in C} g_{x,f} = \sum_{i=1}^{\Gamma} \sum_{x\in P_i} g_{x,f} = \sum_{i=1}^{\Gamma} h_{f,i} = \sum_{x\in C} g'_{x,f}$$

2.
$$\sum_{x\in C_j} g_{x,f} = \sum_{i:\ P_i\subseteq C_j} \sum_{x\in P_i} g_{x,f} = \sum_{i:\ P_i\subseteq C_j} h_{f,i} = \sum_{x\in C_j} g'_{x,f}$$

This completes the proof of the lemma. □

In the streaming setting, we do not know any log-space partition algorithm or hardness result for the ℓ-diversity and fair k-service problems.

3.10 Conclusion and Open Problems

In this paper, we worked within the unified framework of Ding and Xu [72] to obtain simple sampling-based algorithms for a range of constrained k-median/means problems in general metric spaces. Surprisingly, even within this high-level framework, we obtained better (or matched) approximation guarantees of known results designed specifically for the constrained problem. On the one hand, this shows the versatility of the unified approach along with the sampling method. On the other hand, it encourages us to try to design algorithms with better approximation guarantees for these constrained problems. Our matching approximation lower bound for the sampling algorithm suggests that further improvement may not be possible through sampling-based ideas. On the lower bound side, it may be useful to obtain results

similar to that for the unconstrained setting where approximation lower bounds of $(1 + 2/e)$ and $(1 + 8/e)$ are known for k-median and k-means, respectively [53]. Another direction is to find other constrained problems that can fit into the unified framework and can benefit from the results of this work.

Chapter 4

FPT Approximation for Constrained k-Median/Means: Euclidean & Outlier Setting

In this chapter, we study the k-median and k-means problems in the continuous Euclidean space. The continuous Euclidean k-median problem is defined as follows: given a set L of feasible facility locations in p dimensional Euclidean space and a finite client set $C \subset \mathbb{R}^p$, find a set $F \subset \mathbb{R}^p$ of k points (called centers) such that the following cost function is minimised:

$$\text{kmedian-cost}(F, C) \equiv \sum_{x \in C} \min_{f \in F} ||x - f||.$$

.

The continuous Euclidean k-means problem is defined similarly using squared Euclidean distances (i.e., $\|x - f\|^2$ instead of $\|x - f\|$). For the constrained versions of these problems, there already exists $(1 + \varepsilon)$-approximation algorithms with running time $O\left(np \cdot (k/\varepsilon)^{poly(k/\varepsilon)}\right)$ [28,

71]. In this chapter, we extend these algorithms to the outlier and streaming setting. We design $(1+\varepsilon)$-approximation algorithm for the outlier version of the constrained k-means and k-means with running time $O\left(np \cdot ((k+m)/\varepsilon)^{O(k/\varepsilon^{O(1)})}\right)$. Furthermore, we convert the algorithm to a constant-pass log-space streaming algorithm. The high level ideas of the algorithm are the same as previous chapter. However, some parts of the proof slightly differ since the sampling lemmas in Euclidean space are different from the general metric spaces. We also extend the algorithm to the general distance function $\|.\|^z$ that is not studied in the previous works. Note that here, the algorithm gives $(1+\varepsilon)$-approximation guarantee, which is independent of z unlike the problems in the general metric spaces where the approximation guarantee was $3^z+\varepsilon$, which is dependent on z.

4.1 Overview

The constrained k-median/means framework for the continuous Euclidean space is similar to the framework for the discrete metric spaces that we discussed in the previous chapter (see Section 3.1.2). It is just that the discrete set of facility locations L is replaced with $\mathbb{R}^p$, which has an infinite size. For the sake of completeness, we define the framework again. We directly define the framework for the outlier version of the problem since it trivially encapsulates the non-outlier version. We combine the discussion of the k-median and k-means problems by defining the k-service problem that encapsulates both these problems.

Definition 40 (Outlier Euclidean k-Service Problem)**.** *Let k, m, and p be any positive integers, and z be any positive real number. Given a set C of clients in p-dimensional Euclidean space $\mathbb{R}^p$, find a subset $Z \subseteq C$ of size at most m clients and a set $F \subseteq \mathbb{R}^p$ of k facilities such that the k-service cost of $C' := C \setminus Z$ is minimized:* euclid-cost$(F, C') \equiv \sum_{x \in C'} \left\{ \min_{f \in F} \left\{ \|x-f\|^z \right\} \right\}$

The outlier version of the constrained k-service problem in the Euclidean space is defined as follows:

Definition 41 (Constrained Outlier Euclidean k-Service Problem)**.** *Let (C, k, p, z, m) be any instance of the outlier Euclidean k-service problem and $\mathbb{S}$ be any collection of partitionings such that every partitioning $\mathbb{O} \in \mathbb{S}$ is a partitioning of some subset $C' \subseteq C$ of size at least $|C| - m$. Find a clustering $\mathbb{O} = \{O_1, O_2, \ldots, O_k\}$ in $\mathbb{S}$, that minimizes the objective function: $\Phi^*(\mathbb{O}) \equiv \min_{k\text{-center-set } F} \Phi(F, \mathbb{O})$, where $\Phi(F, \mathbb{O}) \equiv \min_{\text{permutation } \pi} \left\{ \sum_{i=1}^{k} \sum_{x \in O_i} \|x - f_{\pi(i)}\|^z \right\}$ and $F = \{f_1, \ldots, f_k\}$.*

Any constrained version of the outlier k-service problem can be solved using the *list k-service problem* and a *partition algorithm*. The outlier version of the list k-service problem in the Euclidean space is defined as follows:

Definition 42 (List Outlier Euclidean k-Service Problem)**.** *Let $\mathcal{I} = (C, k, p, z, m)$ be an arbitrary instance of the outlier Euclidean k-service problem, $\mathbb{O} = \{O_1, \ldots, O_k\}$ be an arbitrary partitioning of the some subset $C' \subseteq C$ of size at least $|C| - m$, and $0 < \varepsilon \leq 1$ be an arbitrary constant. The goal of the problem is to find a list $\mathcal{L}$ of k-center-sets (i.e., each element of the list is a set of k elements from L) such that with probability at least $1 - 1/n$, the list $\mathcal{L}$ contains a k-center-set F such that $\Phi(F, \mathbb{O}) \leq (1 + \varepsilon) \cdot \Phi^*(\mathbb{O})$ for $n = |C|$.*

A partition algorithm for an outlier version of a constrained k-service problem is defined as follows:

Definition 43 (Outlier Partition Algorithm)**.** *Let $\mathcal{I} = (C, k, p, z, m)$ be an instance of the outlier Euclidean k-service problem, and let $\mathbb{S} = \{\mathbb{O}_1, \mathbb{O}_2, \ldots, \mathbb{O}_t\}$ be a collection of clusterings such that each $\mathbb{O}_i$ is a clustering of of some subset of $C' \subseteq C$ of size at least $|C| - m$. Given a center set $F \subseteq L$, an outlier partition algorithm outputs a clustering in $\mathbb{S}$ that has the least clustering cost $\Phi(F, \mathbb{O})$ with respect to F.*

Suppose that we have an algorithm for the list k-service problem and a partition algorithm for a specific constrained k-service problem in the Euclidean space. We can obtain an approximation

algorithm for that constrained k-service problem. The following theorem state this result and is analogous to Theorem 31 for general metric spaces.

Theorem 43. *Let $\mathcal{I} = (C, k, p, z, m, \mathbb{S})$ be any instance of the constrained outlier Euclidean k-service problem, and let $A_{\mathbb{S}}$ be an outlier partition algorithm for $\mathbb{S}$ with running time T_A. Let B be any algorithm for the list outlier Euclidean k-service problem with running time T_B. Then, there is an algorithm that, with probability at least $1 - 1/n$, outputs a clustering $\mathbb{O} \in \mathbb{S}$ that is an $(1+\varepsilon)$-approximate solution for the constrained outlier Euclidean k-service instance $\mathcal{I}$. Moreover, the running time of the algorithm is $O(T_B + |\mathcal{L}| \cdot T_A)$.*

Proof. The proof is analogous to the proof of Theorem 26. □

The goal now becomes to design an algorithm for the list outlier Euclidean k-service problem and outlier partition algorithms for different constrained versions of k-service problems. In Section 4.4, we design an algorithm for the list outlier k-service problem with FPT running time parameterized by m and k. Formally, we state the result as follows:

Theorem 44. *Let $\mathcal{I} = (C, k, p, z, m, \mathbb{O}, \varepsilon)$ be any instance of the list outlier k-service problem. There is an algorithm that outputs a list $\mathcal{L}$ of size at most $O\left((\log n) \cdot ((k+m)z/\varepsilon)^{O\left(k\, z/\varepsilon^{O(z)}\right)}\right)$. Moreover, the running time of the algorithm is $O\left(np \cdot ((k+m)z/\varepsilon)^{O\left(k\, z/\varepsilon^{O(z)}\right)}\right)$, which is FPT in k and m.*

Using Theorems 43 and 44, we obtain the following two main results:

Corollary 15 (Main Result: Outlier Euclidean k-Median)**.** *For any constrained version of the outlier Euclidean k-median problem that has a partition algorithm with running time T, there exists a $(1+\varepsilon)$-approximation algorithm that succeeds with probability at least $1 - 1/n$ and has running time $T \cdot ((k+m)/\varepsilon)^{O\left(k/\varepsilon^{O(1)}\right)} \cdot (\log n) + O\left(np \cdot ((k+m)/\varepsilon)^{O\left(k/\varepsilon^{O(1)}\right)}\right)$.*

Corollary 16 (Main Result: Outlier Euclidean k-Means)**.** *For any constrained version of the outlier Euclidean k-means problem that has a partition algorithm with running time T, there*

exists a $(1+\varepsilon)$-approximation algorithm that succeeds with probability at least $1-1/n$ and has running time $T \cdot ((k+m)/\varepsilon)^{O\left(k/\varepsilon^{O(1)}\right)} \cdot (\log n) + O\left(np \cdot ((k+m)/\varepsilon)^{O\left(k/\varepsilon^{O(1)}\right)}\right)$.

We consider the outlier versions of all the constrained k-service problems mentioned in Table 3.1. In Section 3.9, we designed the FPT time partition algorithms for the outlier versions of all these problems in general metric spaces. Therefore, these algorithms naturally hold in the Euclidean space. Thus, we get FPT time $(1+\varepsilon)$-approximation algorithms for the outlier versions of these problems. Formally, we state the results as follows:

Theorem 45. *There is an* FPT *time $(1+\varepsilon)$-approximation algorithm for the outlier versions of the following constrained k-service problems:*

1. *r-gather k-service problem*
2. *r-capacity k-service problem*
3. *Balanced k-service problem*
4. *Chromatic k-service problem*
5. *Fault-tolerant k-service problem*
6. *Ordered-Weighted-Average k-service problem*
7. *Strongly private k-service problem*
8. *Uncertain k-service problem*

The running time of the algorithm is $((k+m)z/\varepsilon)^{O(kz/\varepsilon^{O(z)})} \cdot n^{O(1)} \cdot p$.

Theorem 46. *There is an* FPT *time $(1+\varepsilon)$-approximation algorithm for the outlier version of the ℓ-diversity k-service problem with running time $((k+m)z\,\omega/\varepsilon)^{O\left(k\cdot\left(\omega+z/\varepsilon^{O(z)}\right)\right)} \cdot n^{O(1)} \cdot p$.*

Theorem 47. *There is an* FPT *time $(1+\varepsilon)$-approximation algorithm for the outlier version of the fair k-service problem with running time $((k+m)z\,\Gamma/\varepsilon)^{O\left(k\cdot\left(\Gamma+z/\varepsilon^{O(z)}\right)\right)} \cdot n^{O(1)} \cdot p$, where Γ denote the number of distinct collection of color classes induced by the colors of clients. Moreover, if the color classes are pair-wise disjoint, then $\Gamma = \omega$, and the running time of the algorithm is $((k+m)z\,\omega/\varepsilon)^{O\left(k\cdot\left(\omega+z/\varepsilon^{O(z)}\right)\right)} \cdot n^{O(1)} \cdot p$.*

This completes the discussion on the constrained outlier Euclidean k-service problem. Next, we convert our algorithm to a streaming algorithm. We require a streaming version of the list outlier k-service algorithm and a streaming version of the partition algorithm. In Section 4.5, we design a constant-pass log-space streaming algorithm for the list outlier Euclidean k-service

problem. In Section 3.9, we designed the streaming partition algorithms for the outlier versions of some of the problems given in Table 3.1 in general metric space. Therefore, the algorithm naturally holds for the Euclidean k-service problem as well.

4.2 Related Work

A unified framework for the constrained Euclidean k-means/k-median problems was introduced by Ding and Xu [72]. Using this framework, they designed $(1+\varepsilon)$-approximation algorithms for various constrained clustering problems with FPT running time parameterized by k. They obtained the results using an algorithm for the list version of the k-means problem (even though it was not formally defined in their work). The running time of their algorithm was $O(nd \cdot (\log n)^k \cdot 2^{poly(k/\varepsilon)})$ and the list size was $(\log n)^k \cdot 2^{poly(k/\varepsilon)}$. Bhattacharya *et al.* [28] formally defined the list k-service problem. They obtained a faster algorithm for the list k-service problem with running time of $O(nd \cdot (k/\varepsilon)^{O(\log(k/\varepsilon))})$ and list size of $(k/\varepsilon)^{O(\log(k/\varepsilon))}$. We use a sampling-based approach similar to the algorithm of Bhattacharya *et al.* [28]. Our work differs from the previous works in the following ways:

1. Bhattacharya *et al.* [28] gave an algorithm for the list-k-means problem with list size $|\mathcal{L}| = (\frac{k}{\varepsilon})^{O(\frac{k}{\varepsilon})}$ and running time $O(nd|\mathcal{L}|)$. Their algorithm explores a rooted tree of size $(\frac{k}{\varepsilon})^{O(\frac{k}{\varepsilon})}$ and depth k where the degree of every non-leaf vertex is $(\frac{k}{\varepsilon})^{O(\frac{1}{\varepsilon})}$. Every node in this tree has an associated center and the path from root to a leaf node gives one of the k-center-sets for the output list. The algorithm has an unavoidable iteration of depth k since their analysis works only when the centers are picked *one-by-one* in k iterations. We circumvent this inherent restriction by using a constant factor approximate solution F to the unconstrained k-means/median problem for the given dataset C. That is, service-cost$(F, C) \leq \alpha \cdot$ OPT, where OPT denotes the optimal k-means/median cost. Then the sampling algorithm runs in a *single* iteration where $O((k/\varepsilon)^{O(k/\varepsilon)})$ points from C are D^z-sampled with respect to F. We show that all the good k centers for k clusters

can simultaneously be found from the sampled points and points in the set F. Thus, we obtain the list $\mathcal{L}$ in a single shot. This technique helps us in designing streaming algorithm for the problem.

2. For the unconstrained outlier Euclidean k-means problem there exists a $(1+\varepsilon)$ approximation with running time $O\left(np \cdot ((k+m)/\varepsilon)^{(k/\varepsilon)^{O(1)}}\right)$ [80]. We design a general algorithm for the cost function $\|.\|^z$ and outlier setting. We design $(1+\varepsilon)$-approximation algorithm for the outlier versions of a range of constrained k-service problems in the Euclidean space with FPT running time, parameterized by k and m.

Note that streaming *coreset* constructions provide another approach to designing streaming algorithm for the k-means problem. An (ε, k) coreset of a dataset $C \subset \mathbb{R}^p$ is a weighted set $S \subset \mathbb{R}^p$ along with a weight function $w : S \to \mathbb{R}^+$ such that for any k-center-set F, we have: $|\sum_{s\in S} \min_{f\in F} w(s) \cdot \|s-f\|^2 - \sum_{x\in C} \min_{f\in F} \|x-f\|^2| \leq \varepsilon \cdot \sum_{x\in C} \min_{f\in F} \|x-f\|^2$. So, it is sufficient to find a k-center-set that gives $(1+\varepsilon)$-approximation for a coreset S (instead of the dataset C). There exists one-pass streaming coreset construction [77, 44] that uses $poly\left(k, \frac{1}{\varepsilon}, \log n\right)$ space and outputs a coreset of size $poly\left(k, \frac{1}{\varepsilon}, \log n\right)$. Using this, one can design a single-pass streaming algorithm for the k-means problem by first running the streaming algorithm to output a coreset and then finding a $(1+\varepsilon)$-approximate k center set for the small coreset. If the output is supposed to be a clustering, then we will need to make another pass over the data. Note that the same idea of working on coreset does not trivially carry over to the constrained k-service problem since there are additional constraints and general cost function $\|.\|^z$.

4.3 Notations and Identities

We give some basic definitions and results that is useful in the context of the Euclidean k-service problem. The unconstrained Euclidean k-service cost of a client set C with respect to a

center set F is given by the following cost function:

$$\text{euclid-cost}(F, C) \equiv \sum_{x \in C} \min_{f \in F} ||x - f||^z.$$

We will use the above cost function repeatedly in our discussion. Hence for simplicity, when $F = \{f\}$ is a singleton set, then we use euclid-cost(f, C) instead of euclid-cost$(\{f\}, C)$. We denote the optimal cost of the unconstrained Euclidean k-service instance by OPT(C, k). That is, OPT$(C, k) \equiv \min_{k\text{-center-set } F \subseteq R^p} \Big\{\text{euclid-cost}(F, C)\Big\}$

Our algorithm is based on simple sampling idea. The following sampling result of Cohen-Addad *et al.* [57] will be used in our analysis. The lemma says that an approximate solution to the 1-service problem can be computed by uniformly sampling a small set of points from the dataset C. We denote the optimal 1-service cost of any dataset $C \subset \mathbb{R}^p$ by $\Delta(C) = \min_{f \in \mathbb{R}^p} \big\{\text{euclid-cost}(f, C)\big\}$.

Lemma 32 (Theorem 1 [57])**.** *Let C be any arbitrary instance of the Euclidean 1-service problem. Then there exists an algorithm that uniformly samples $O(\varepsilon^{-z-3} \cdot polylog(\varepsilon^{-1})) \log^2(1/\delta)$ points from C and computes $(1+\varepsilon)$-approximate solution of C with probability at least $1 - \delta$.*

In particular, we use the following corollary of the above lemma in our algorithms and proofs.

Corollary 17 (Corollary of Lemma 32)**.** *Let C be any arbitrary instance of the Euclidean 1-service problem. Then there exists an algorithm that uniformly samples $\tau = O((\varepsilon/16)^{-z-3} \cdot polylog(\varepsilon^{-1}))$ points from C and computes $(1+\varepsilon/16)$-approximate solution of C with probability at least $3/4$.*

We denote a $(1+\varepsilon)$-approximate solution obtained by the above lemma, by $\mu(C)$, i.e., euclid-cost$(\mu(C), C) \leq (1+\varepsilon) \cdot \Delta(C)$.

We use the following facts frequently in our proofs. We also used these facts earlier in Chapter 3. For proofs of these facts, please see Section 3.3.

Fact 1 (Binomial Approximation)**.** *For $\varepsilon \cdot n \leq 1/2$, we have $(1+\varepsilon)^n \leq (1+2\varepsilon n)$*

Fact 2. *For any $\delta, z, a, b > 0$, we have $(a+b)^z \leq (1+\delta)^z \cdot b^z + \left(1+\frac{1}{\delta}\right)^z \cdot a^z$.*

4.4 Algorithm for List Outlier k-Service Problem

The algorithm `List-Outlier-Euclidean-k-Service` is given in Figure 4.1. The algorithm takes as input client set C, number of clusters k, dimension of the Euclidean space p, number of outliers m, and error parameter ε. The algorithm outputs a list $\mathcal{L}$ of k-center sets. Let $Z \subseteq C$ be the unknown set of m outliers, $\mathbb{O} = \{O_1, ..., O_k\}$ be the unknown clustering of the client set $C \setminus Z$. Our goal is to find (approximately) good centers for these clusters. Since $\mathbb{O}$ is not given we cannot hope to output a single such k-center-set. We are allowed to output a list of such center sets.

`List-Outlier-Euclidean-k-Service`$(C, k, p, \varepsilon, m)$

Inputs: Outlier Euclidean k-Service Instance (C, k, p, m) and accuracy ε
Output: A list $\mathcal{L}$, each element in $\mathcal{L}$ being a k-center set
Constants: $\eta = 2\tau/\gamma; \tau = O((\varepsilon/16)^{-z-3} \cdot \mathit{polylog}(\varepsilon^{-1}))$; $\zeta = (1+32z/\varepsilon)$;
$\beta = (1+16z/\varepsilon)^z \cdot (2+\varepsilon/2)$; $\gamma = \frac{\varepsilon}{2\,\alpha\,\beta\,k} \cdot \left(\frac{\varepsilon}{50\,\zeta}\right)^z$

(1) Run any α-approximation algorithm with $\alpha = \text{poly}(k+m)$ for the unconstrained Euclidean $(k+m)$-service instance $(C, k+m, p)$ and let F be the obtained center-set. (*k-means++ [16] is one such algorithm.*)
(2) $\mathcal{L} \leftarrow \emptyset$
(3) Repeat $2^k \cdot (\log n)$ times:
(4) Sample a multi-set M of ηk points from C using D^z-sampling w.r.t. center set F
(5) $M \leftarrow M \cup \{\tau\, k$ copies of each element in $F\}$
(6) For all disjoint subsets $S_1, ..., S_t$ of M such that $\forall i, |S_i| = \tau$:
(7) $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mu(S_1), ..., \mu(S_k))\}$
($\mu(S_j)$ *is $(1+\varepsilon)$-approximate solution to 1-service cost of S_j. It is obtained using Lemma 32*)
(8) return($\mathcal{L}$)

Algorithm 4.1: List Outlier Euclidean k-Service Algorithm

We will show the list $\mathcal{L}$ produced by the `List-Outlier-Euclidean-k-Service` algorithm, with high probability, will contain a k-center set S such that $\Phi(S, \mathbb{O}) \leq (1+\varepsilon) \cdot \Phi^*(\mathbb{O})$. We formally state our result as the next theorem.

Theorem 48. *Let (C, k, p, m) be any instance of the outlier Euclidean k-service problem and $0 < \varepsilon \leq 1/2$. Let $Z \setminus C$ be any arbitrary set of m outliers and $\mathbb{O} = \{O_1, ..., O_k\}$ denote an arbitrary clustering of $C \setminus Z$. Let $\mathcal{L}$ denote the list returned by the algorithm* `List-Outlier-Euclidean-k-Service`*$(C, k, p, \varepsilon, m)$. Then with probability at least $1 - 1/n$, $\mathcal{L}$ contains a center set S such that*

$$\Phi(S, \mathbb{O}) \leq (1+\varepsilon) \cdot \Phi^*(\mathbb{O})$$

where $\Phi^*(\mathbb{O}) = \sum_{i=1}^{k} \Delta(O_i)$. Moreover, $|\mathcal{L}| = (\log n) \cdot \left(((k+m)z/\varepsilon)^{O(kz)/\varepsilon^{O(z)}}\right)$ and the running time of the algorithm is $O(np|\mathcal{L}|)$.

First we analyse the running time of the algorithm. In line(1) of the algorithm, we run k-means++ algorithm [16] that gives an $O(4^z \cdot \log(k+m))$-approximation guarantee and a running time $O(np(k+m))$, where $k+m$ are the number of centers. Line (4) of the algorithm D^z-samples $(zk/\varepsilon)^{O(z)}$ points from C with respect to F. It takes $O(np(k+m)(zk/\varepsilon)^{O(z)})$ time. After line(5) of the algorithm, M contains $O(k+m)(zk/\varepsilon)^{O(z)}$ points. Then, in lines (6) and (7), the algorithm takes all possible τ subsets of M, computes 1-service cost of each subset, and take all k-combinations of them. It takes $O\left(((k+m)z/\varepsilon)^{O(kz)/\varepsilon^{O(z)}}\right)$ time. Since the algorithm repeats lines (4)-(7) $2^k \log n$ times, size of list $\mathcal{L}$ is $(\log n) \cdot \left(((k+m)z/\varepsilon)^{O(kz)/\varepsilon^{O(z)}}\right)$ and running time of the algorithm is $O\left(np \cdot ((k+m)z/\varepsilon)^{O(kz)/\varepsilon^{O(z)}})\right)$.

Now, we show that the list $\mathcal{L}$ contains a good k-center set for $\mathbb{O}$ with probability at least $1-1/n$. Note that the outer iteration (repeat $2^k \cdot (\log n)$ times in line (3)) is to amplify the probability that the list $\mathcal{L}$ containing a good k-center set. We will show that the probability of finding a good k-center set in one iteration is at least $(1/2)^k$ and the theorem follows from simple probability

calculation. So in the remaining discussion we will only discuss one iteration of the algorithm. Consider the multi-set M after line (5) of the algorithm. We will show that with probability at least $(1/2)^k$, there are disjoint (multi) subsets $T_1, ..., T_k$ each of size τ such that for every $j = 1, ..., k$,

$$\text{euclid-cost}(\mu(T_j), O_j) \leq \left(1 + \frac{\varepsilon}{2}\right) \cdot \Delta(O_j) + \frac{\varepsilon}{2k} \cdot \Phi^*(\mathbb{O}). \tag{4.1}$$

Since we try out all possible subsets in step (7), we will get the desired result. More precisely, we will argue in the following manner: consider the multi-set $F' = \{\tau\, k \text{ copies of each element in } F\}$. We can interpret F' as a union of multi-sets $F'_1, F'_2, ..., F'_t$, where $F'_j = \{\tau \text{ copies of each element in } F\}$. Also, since M consists of ηk independently sampled points, we can interpret M as a union of multi-sets $M'_1, M'_2, ..., M'_k$ where M'_1 is the first η points sampled, M'_2 is the second η points and so on. For all $j = 1, ..., k$, let $M_j = F'_j \cup (M'_j \cap O_j)$.[1] We will show that for every $j \in \{1, ..., k\}$, with probability at least $(1/2)$, M_j contains a subset T_j of size τ that satisfies eqn. (4.1). Note that T_j's being disjoint follows from the definition of M_j. It will be sufficient to prove the following lemma.

Lemma 33. *Consider the sets $M_1, ..., M_k$ as defined above. For any $j \in \{1, ..., k\}$,*

$$\mathbf{Pr}\left[\exists T_j \subseteq M_j \text{ s.t. } |T_j| = \tau \text{ and } \left(\text{euclid-cost}(\mu(T_j), O_j) \leq \left(1 + \frac{\varepsilon}{2}\right) \cdot \Delta(O_j) + \frac{\varepsilon}{2k}\Phi^*(\mathbb{O})\right)\right] \geq \frac{1}{2},$$

where $\mu(T_j)$ is the $(1 + \varepsilon)$-approximate solution to the optimal 1-service cost of T_j.

It is easy to see that the above lemma gives the desired result:

$$\Phi(\{\mu(T_1), \ldots, \mu(T_k)\}, \mathbb{O}) = \sum_{j=1}^{k} \text{euclid-cost}(\mu(T_j), O_j) \leq (1 + \varepsilon) \cdot \Phi^*(\mathbb{O})$$

We prove the above lemma in the remaining discussion. We do a case analysis that is based on whether $\frac{\text{euclid-cost}(F, O_j)}{\text{euclid-cost}(F, C)}$ is large or small for a particular $j \in \{1, ..., k\}$.

[1] $M'_j \cap O_j$ in this case, denotes those points in the multi-set M'_j that belongs to O_j.

- *Case-I* $\Big($euclid-cost$(F, O_j) \leq \frac{\varepsilon}{2\,\alpha\,\beta\,k} \cdot$ euclid-cost$(F, C)\Big)$: Here we will show that there is a subset $T_j \subseteq F'_j \subseteq M_j$ that satisfies eqn. (4.1).

- *Case-II* $\Big($euclid-cost$(F, O_j) > \frac{\varepsilon}{2\,\alpha\,\beta\,k} \cdot$ euclid-cost$(F, C)\Big)$: Here we will show that M_j contains a subset T_j such that euclid-cost$(\mu(T_j), O_j) \leq \Big(1 + \frac{\varepsilon}{2}\Big) \cdot \Delta(O_j)$ and hence T_j also satisfies eqn. (4.1).

We discuss these two cases next. The analysis is similar to the analysis of the D^2-sampling based algorithm for k-means by Bhattacharya *et al.* [28]. Since there are a few crucial differences, and for the sake of clarity we continue with the detailed proof.

4.4.1 Proof of Case-I: *Low-cost clusters*

Case-I: $\Big($**euclid-cost**$(F, O_j) \leq \frac{\varepsilon}{2\,\alpha\,\beta\,k} \cdot$ **euclid-cost**$(F, C)\Big)$

For any point $x \in C$, let $c(x)$ denote the center in the set F that is closest to x. That is, $c(x) = \arg\min_{c \in F} ||c - x||$. Given this definition, note that:

$$\sum_{x \in O_j} ||x - c(x)||^z = \text{euclid-cost}(F, O_j) \tag{4.2}$$

We define the multi-set $O'_j = \{c(x) : x \in O_j\}$. Let μ and μ' denote the optimal centers to the 1-service cost of the point sets O_j and O'_j, respectively. So, we have $\Delta(O_j) =$ euclid-cost(μ, O_j) and $\Delta(O'_j) =$ euclid-cost(μ', O'_j). We will show that $\Delta(O_j) \approx \Delta(O'_j)$. First, we note the following lemma:

Lemma 34. *For any constant $\delta > 0$,* $\Delta(O'_j) \leq (1+1/\delta)^z \cdot$euclid-cost$(F, O_j) + (1+\delta)^z \cdot \Delta(O_j)$.

Proof. The proof follows from the following sequence of inequalities.

$$\begin{aligned}
\Delta(O'_j) &= \sum_{x\in O'_j} \|x-\mu'\|^z \\
&\le \sum_{x\in O'_j} \|x-\mu\|^z, && (\because \mu' \text{ is optimal for } O'_j) \\
&= \sum_{x\in O_j} \|c(x)-\mu\|^z \\
&\le \sum_{x\in O_j} \left(\|x-c(x)\| + \|x-\mu\|\right)^z, && \text{(using triangle-inequality)} \\
&\le \sum_{x\in O_j} \left((1+1/\delta)^z\|x-c(x)\|^z + (1+\delta)^z\|x-\mu\|^z\right), && \text{(for any } \delta>0\text{, using Fact 2)} \\
&= (1+1/\delta)^z \cdot \text{euclid-cost}(F, O_j) + (1+\delta)^z \cdot \Delta(O_j),
\end{aligned}$$

□

Next, we show that a good center for the 1-service cost of O'_j is also a good center for O_j.

Lemma 35. *Let μ'' be a point such that* euclid-cost$(\mu'', O'_j) \le \left(1+\frac{\varepsilon}{8}\right)\cdot\Delta(O'_j)$. *Then,* euclid-cost$(\mu'', O_j) \le \left(1+\frac{\varepsilon}{2}\right)\cdot\Delta(O_j) + \frac{\varepsilon}{2k}\cdot\Phi^*(\mathbb{O})$.

Proof. The proof follows from the following sequence of inequalities.

euclid-cost(μ'', O_j)

$$\begin{aligned}
&= \sum_{x\in O_j} \|x-\mu''\|^z \\
&\le \sum_{x\in O_j} \left(\|x-c(x)\| + \|c(x)-\mu''\|\right)^z, \\
& && \text{(using triangle inequality)}
\end{aligned}$$

$$\leq \sum_{x \in O_j} \left((1+1/\delta)^z \cdot \|x - c(x)\|^z + (1+\delta)^z \cdot \|c(x) - \mu''\|^z \right),$$

(for any $\delta > 0$, using Fact 2)

(and we will later decide the value of δ)

$$= (1+1/\delta)^z \cdot \text{euclid-cost}(F, O_j) + (1+\delta)^z \cdot \text{euclid-cost}(\mu'', O'_j)$$

$$\leq (1+1/\delta)^z \cdot \text{euclid-cost}(F, O_j) + (1+\delta)^z \cdot (1+\varepsilon/8) \cdot \Delta(O'_j)$$

$$\leq (1+1/\delta)^z \cdot (1 + (1+\delta)^z \cdot (1+\varepsilon/8)) \cdot \text{euclid-cost}(F, O_j)$$
$$+ (1+\delta)^{2z} \cdot (1+\varepsilon/8) \cdot \Delta(O_j), \quad \text{(using Lemma 34)}$$

$$= (1+16z/\varepsilon)^z \cdot (1 + (1+\varepsilon/16z)^z \cdot (1+\varepsilon/8)) \cdot \text{euclid-cost}(F, O_j)$$
$$+ (1+\varepsilon/16z)^{2z} \cdot (1+\varepsilon/8) \cdot \Delta(O_j), \quad \text{(substituting } \delta = \varepsilon/16z)$$

$$\leq (1+16z/\varepsilon)^z \cdot (1 + (1+\varepsilon/16) \cdot (1+\varepsilon/8)) \cdot \text{euclid-cost}(F, O_j)$$
$$+ (1+\varepsilon/8) \cdot (1+\varepsilon/8) \cdot \Delta(O_j), \quad \text{(using Fact 1)}$$

$$\leq (1+16z/\varepsilon)^z \cdot (2+\varepsilon/2) \cdot \text{euclid-cost}(F, O_j) + (1+\varepsilon/2) \cdot \Delta(O_j)$$

(for $\varepsilon \leq 1$)

$$\leq (1+16z/\varepsilon)^z \cdot (2+\varepsilon/2) \cdot \frac{\varepsilon}{2\,\alpha\,\beta\,k} \cdot \text{euclid-cost}(F, C) + (1+\varepsilon/2) \cdot \Delta(O_j)$$

(by the defn. of Case-I)

$$\leq (1+16z/\varepsilon)^z \cdot (2+\varepsilon/2) \cdot \frac{\varepsilon}{2\,\beta\,k} \cdot \text{OPT}(C, k+m) + (1+\varepsilon/2) \cdot \Delta(O_j),$$

($\because$ euclid-cost$(F, C) \leq \alpha \cdot$ OPT$(C, k+m)$)

$$\leq (1+16z/\varepsilon)^z \cdot (2+\varepsilon/2) \cdot \frac{\varepsilon}{2\,\beta\,k} \cdot \Phi^*(\mathbb{O}) + (1+\varepsilon/2) \cdot \Delta(O_j)$$

($\because$ unconstrained optimal cost $\leq$ constrained optimal cost)

$$= \frac{\varepsilon}{2k} \cdot \Phi^*(\mathbb{O}) + \left(1 + \frac{\varepsilon}{2}\right) \cdot \Delta(O_j), \qquad (\because \beta = (1+16z/\varepsilon)^z \cdot (2+\varepsilon/2))$$

This completes the proof of the lemma. □

We know from Corollary 17 that there exists a (multi) subset S of O'_j of size τ such that $\mu(S)$ of these points satisfies the condition of the lemma above with probability at least $3/4$. Since F'_j contains at least τ copies of every element of F, there is a subset $T_j \subseteq F'_j$ that satisfies eqn. (4.1) with probability at least $3/4$. So, for any index $j \in \{1, ..., k\}$ such that $\frac{\text{euclid-cost}(F, O_j)}{\text{euclid-cost}(F, C)} \leq \frac{\varepsilon}{2\,\alpha\,\beta\,k}$, M_j has a good subset T_j with probability $3/4$.

4.4.2 Proof of Case-II: *High-cost clusters*

Case-II: $\Big($**euclid-cost**$(F, O_j) > \frac{\varepsilon}{2\,\alpha\,\beta\,k} \cdot$ **euclid-cost**$(F, C)\Big)$

If we can show that a D^z-sampled set with respect to center set F has a subset S that may be considered uniform sample from O_j, then we can use Lemma 32 to argue that M_j has a subset T_j such that $\mu(T_j)$ is a good center for O_j. Note that since $\frac{\text{euclid-cost}(F, O_j)}{\text{euclid-cost}(F, C)} > \frac{\varepsilon}{2\,\alpha\,\beta\,k}$, we can argue that if we D^z-sample $O\big((kz/\varepsilon)^{O(z/\varepsilon)}\big)$ elements, then we will get a good representation from O_j. However, some of the points from O_j may be very close to one of the centers in F and hence will have a very small chance of being D^z-sampled. In such a case, no subset S of a D^z-sampled set will behave like a uniform sample from O_j. So, we need to argue more carefully taking into consideration the fact that there may be points in O_j for which the chance of being D^z-sampled may be very small. Here is the high-level argument that we will build:

- Consider the set O'_j which is same as O_j except that points in O_j that are very close to F have been "collapsed" to their closest center in F.
- Argue that a good center for the set O'_j is a good center for O_j.
- Show that a convex combination of copies of centers in F (i.e., F'_j) and D^z-sampled points from O_j gives a good center for the set O'_j.

The closeness of point in O_j to points in F is quantified using radius R that is defined by the

equation:

$$R^z \stackrel{defn.}{=} \left(\frac{\varepsilon}{50\,\zeta}\right)^z \cdot \frac{\text{euclid-cost}(F, O_j)}{|O_j|}, \quad where \quad \zeta = (1 + 32z/\varepsilon). \tag{4.3}$$

Let O_j^{near} be the points in O_j that are within a distance of R from a point in set F and $O_j^{far} = O_j \setminus O_j^{near}$. That is, $O_j^{near} = \{x \in O_j : \min_{c \in F} ||x - c|| \leq R\}$ and $O_j^{far} = O_j \setminus O_j^{near}$. Using these we define the multi-set O'_j as:

$$O'_j = O_j^{far} \cup \{c(x) : x \in O_j^{near}\}$$

Note that $|O_j| = |O'_j|$. Let μ and μ' are the optimal centers for the 1-service cost of O_j and O'_j, respectively. Let $n = |O_j|$ and $\bar{n} = |O_j^{near}|$. We first show a lower bound on $\Delta(O_j)$ in terms of R.

Lemma 36. $\Delta(O_j) \geq \bar{n} \cdot \left(\frac{12\,\gamma\,R}{\varepsilon}\right)^z$, *where* $\zeta = (1 + 32z/\varepsilon)$.

Proof. Let $c = \arg\min_{c' \in F} ||\mu - c'||$. We do a case analysis:

1. Case 1: $||\mu - c|| \geq \frac{13\,\zeta\,R}{\varepsilon}$

   Consider any point $p \in O_j^{near}$. From triangle inequality, we have:

   $$||p - \mu|| \geq ||c(p) - \mu|| - ||c(p) - p|| \geq \frac{13\,\zeta\,R}{\varepsilon} - R \geq \frac{12\,\zeta\,R}{\varepsilon}.$$

   This gives: $\Delta(O_j) \geq \sum_{p \in O_j^{near}} ||p - \mu||^z \geq \bar{n} \cdot \left(\frac{12\,\zeta\,R}{\varepsilon}\right)^z$.

2. Case 2: $||\mu - c|| < \frac{13\,\zeta\,R}{\varepsilon}$

   In this case, we have:

   $$\text{euclid-cost}(F, O_j) \leq \text{euclid-cost}(c, O_j)$$

$$
\begin{aligned}
&= \sum_{x \in O_j} \|x - c\|^z \\
&\leq \sum_{x \in O_j} \left(\|x - \mu\| + \|\mu - c\|\right)^z, && \text{(using triangle inequality)} \\
&\leq 2^z \cdot \sum_{x \in O_j} \left(\|x - \mu\|^z + \|\mu - c\|^z\right), \\
& && \text{(using Fact 2 and } \delta = 1) \\
&\leq 2^z \cdot \Delta(O_j) + n \cdot 2^z \cdot \|\mu - c\|^z
\end{aligned}
$$

On rearranging the terms, we get:

$$
\begin{aligned}
\Delta(O_j) &\geq \frac{\text{euclid-cost}(F, O_j)}{2^z} - n \cdot ||\mu - c||^z \\
&\geq \frac{\text{euclid-cost}(F, O_j)}{2^z} - n \cdot \left(\frac{13\,\zeta\,R}{\varepsilon}\right)^z \\
&= n \cdot \left(\frac{50\,\zeta\,R}{2\,\varepsilon}\right)^z - n \cdot \left(\frac{13\,\zeta\,R}{\varepsilon}\right)^z \\
&\geq n \cdot \left(\frac{12\,\zeta\,R}{\varepsilon}\right)^z \\
&\geq \bar{n} \cdot \left(\frac{12\,\zeta\,R}{\varepsilon}\right)^z
\end{aligned}
$$

This completes the proof of the lemma. □

Next, we note the following lemma:

Lemma 37. *For any $\delta > 0$, $\Delta(O'_j) \leq (1 + 1/\delta)^z \cdot \text{euclid-cost}(F, O_j^{near}) + (1 + \delta)^z \cdot \Delta(O_j)$.*

Proof. The proof follows from the following sequence of inequalities.

$$\begin{aligned}
\Delta(O'_j) &= \sum_{x\in O'_j} \|x-\mu'\|^z \\
&\le \sum_{x\in O'_j} \|x-\mu\|^z, && (\because \mu' \text{ is optimal for } O'_j) \\
&= \sum_{x\in O_j^{near}} \|c(x)-\mu\|^z + \sum_{x\in O_j^{far}} \|x-\mu\|^z \\
&\le \sum_{x\in O_j^{near}} (\|x-c(x)\| + \|x-\mu\|)^z + \sum_{x\in O_j^{far}} \|x-\mu\|^z, && \text{(using triangle-inequality)} \\
&\le \sum_{x\in O_j^{near}} \left((1+1/\delta)^z \|x-c(x)\|^z + (1+\delta)^z \|x-\mu\|^z\right) \\
&\qquad + \sum_{x\in O_j^{far}} \|x-\mu\|^z, && \text{(for any } \delta>0 \text{ using Fact 2)} \\
&\le (1+1/\delta)^z \cdot \text{euclid-cost}(F, O_j^{near}) + (1+\delta)^z \cdot \Delta(O_j),
\end{aligned}$$

□

In the next lemma, we show that a good center for the 1-service cost of O'_j is a good center for O_j as well.

Lemma 38. *Let μ'' be a point such that* euclid-cost$(\mu'', O'_j) \le \left(1+\frac{\varepsilon}{16}\right)\cdot\Delta(O'_j)$. *Then,* euclid-cost$(\mu'', O_j) \le \left(1+\frac{\varepsilon}{2}\right)\cdot\Delta(O_j)$.

Proof. The proof follows from the following sequence of inequalities.

euclid-cost(μ'', O_j)

$$= \sum_{x\in O_j} \|x-\mu''\|^z$$

$$= \sum_{x \in O_j^{near}} \|x - \mu''\|^z + \sum_{x \in O_j^{far}} \|x - \mu''\|^z$$

$$\leq \sum_{x \in O_j^{near}} (\|x - c(x)\| + \|c(x) - \mu''\|)^z + \sum_{x \in O_j^{far}} \|x - \mu''\|^z,$$

(using triangle inequality)

$$\leq \sum_{x \in O_j^{near}} ((1 + 1/\delta)^z \cdot \|x - c(x)\|^z + (1 + \delta)^z \cdot \|c(x) - \mu''\|^z) + \sum_{x \in O_j^{far}} \|x - \mu''\|^z,$$

(for any $\delta > 0$, using Fact 2)

(we will later decide the value of δ)

$$= (1 + 1/\delta)^z \cdot \text{euclid-cost}(F, O_j^{near}) + (1 + \delta)^z \cdot \sum_{x \in O_j^{near}} \|c(x) - \mu''\|^z + \sum_{x \in O_j^{far}} \|x - \mu''\|^z$$

$$\leq (1 + 1/\delta)^z \cdot \text{euclid-cost}(F, O_j^{near}) + (1 + \delta)^z \cdot \sum_{x \in O_j'} \|x - \mu''\|^z$$

$$= (1 + 1/\delta)^z \cdot \text{euclid-cost}(F, O_j^{near}) + (1 + \delta)^z \cdot \text{euclid-cost}(\mu'', O_j')$$

$$\leq (1 + 1/\delta)^z \cdot \text{euclid-cost}(F, O_j^{near}) + (1 + \delta)^z \cdot (1 + \varepsilon/16) \cdot \Delta(O_j')$$

$$\leq (1 + 1/\delta)^z \cdot (1 + (1 + \delta)^z \cdot (1 + \varepsilon/16)) \cdot \text{euclid-cost}(F, O_j^{near}) + (1 + \delta)^{2z} \cdot (1 + \varepsilon/16) \cdot \Delta(O_j), \quad \text{(using Lemma 37)}$$

$$= (1 + 32z/\varepsilon)^z \cdot (1 + (1 + \varepsilon/32z)^z \cdot (1 + \varepsilon/16)) \cdot \text{euclid-cost}(F, O_j^{near}) + (1 + \varepsilon/32z)^{2z} \cdot (1 + \varepsilon/16) \cdot \Delta(O_j), \quad \text{(on substituting } \delta = \varepsilon/32z)$$

$$\leq (1 + 32z/\varepsilon)^z \cdot (1 + (1 + \varepsilon/32) \cdot (1 + \varepsilon/16)) \cdot \text{euclid-cost}(F, O_j^{near}) + (1 + \varepsilon/16) \cdot (1 + \varepsilon/16) \cdot \Delta(O_j), \quad \text{(using Fact 1)}$$

$$\leq (1 + 32z/\varepsilon)^z \cdot (1 + 1 + \varepsilon/4) \cdot \text{euclid-cost}(F, O_j^{near}) + (1 + \varepsilon/4) \cdot \Delta(O_j),$$

(for $\varepsilon \leq 1$)

$$\leq (1 + 32z/\varepsilon)^z \cdot (2 + \varepsilon/4) \cdot \bar{n}R^z + (1 + \varepsilon/4) \cdot \Delta(O_j)$$

$$\leq (1+32z/\varepsilon)^z \cdot (2+\varepsilon/4) \cdot \left(\frac{\varepsilon}{12\zeta}\right)^z \cdot \Delta(O_j) + (1+\varepsilon/4)\cdot\Delta(O_j),$$

(using Lemma 36)

$$\leq \frac{\varepsilon}{4}\cdot\Delta(O_j) + (1+\varepsilon/4)\cdot\Delta(O_j) \quad (\because \zeta = (1+32z/\varepsilon) \text{ and } \varepsilon \leq 1)$$

$$= (1+\varepsilon/2)\cdot\Delta(O_j)$$

This completes the proof of the lemma. □

Given the above lemma, all we need to argue is that our algorithm indeed considers a center μ'' such that euclid-cost$(\mu'', O'_j) \leq (1+\varepsilon/16)\cdot\Delta(O'_j)$. For this we would need about $\Omega(\tau)$ uniform samples from O'_j. However, our algorithm can only sample using D^z-sampling w.r.t. F. For ease of notation, let $c(O_j^{near})$ denote the multi-set $\{c(p) : p \in O_j^{near}\}$. Recall that O'_j consists of O_j^{far} and $c(O_j^{near})$. The first observation we make is that the probability of sampling an element from O_j^{far} is reasonably large (proportional to $\frac{\varepsilon}{k}$). Using this fact, we show how to sample from O'_j (almost uniformly). Finally, we show how to convert this almost uniform sampling to uniform sampling (at the cost of increasing the size of sample).

Lemma 39. *Let x be a sample from D^z-sampling w.r.t. F. Then, for any point $p \in O_j^{far}$, $\mathbf{Pr}[x=p] \geq \frac{\gamma}{|O_j|}$, where γ denotes $\frac{\varepsilon}{2\,\alpha\,\beta\,k}\cdot\left(\frac{\varepsilon}{50\,\zeta}\right)^z$.*

Proof. If $x \in O_j^{far}$, then euclid-cost$(F,\{x\}) \geq R^z = \left(\frac{\varepsilon}{50\,\zeta}\right)^z \cdot \frac{\text{euclid-cost}(F,O_j)}{|O_j|}$. Therefore,

$$\frac{\text{euclid-cost}(F,\{x\})}{\text{euclid-cost}(F,C)} \geq \frac{R^z}{\text{euclid-cost}(F,C)} \geq \frac{\text{euclid-cost}(F,O_j)}{\text{euclid-cost}(F,C)}\cdot\left(\frac{\varepsilon}{50\,\zeta}\right)^z \cdot \frac{1}{|O_j|}$$

$$\geq \frac{\varepsilon}{2\,\alpha\,\beta\,k}\cdot\left(\frac{\varepsilon}{50\,\zeta}\right)^z \cdot \frac{1}{|O_j|}.$$

This completes the proof of the lemma. □

Note that the algorithm adds the entire set F in M. It means the algorithm samples a point from $c(O_j^{near})$ with probability exactly 1. Therefore, based on this fact and the above lemma, it can be said that the algorithm is sampling each point in O'_j with probability at least $\gamma/|O_j|$.

Now, consider a new experiment in which E_c denote the event of choosing O_j and E_n denote the event of not choosing O_j. The experiment is such that Pr[E_c] $= \gamma$ and Pr[E_n] $= 1 - \gamma$. Let E_x denote the event of choosing a point $x \in O_j$. The experiment is such that given E_c, an element $x \in O_j$ is chosen with uniform probability $1/|O_j|$. In other words, Pr[$E_x \mid E_c$] $= 1/|O_j|$. Therefore, the probability of choosing a point $x \in O_j$ is exactly $\gamma/|O_j|$. Suppose we run this new experiment $\eta = 2\tau/\gamma$ times and S be the sampled set. Then, expected number of points in S is 2τ. Then, with probability at least $3/4$, S contains at least τ points (using Chernoff-Hoeffding bound). Since all points in S are uniformly sampled, using Corollary 17, $\mu(S)$ gives $(1 + \varepsilon/16)$ approximation for the 1-service cost of O'_j with probability at least $(3/4)^2 \geq 1/2$. Note that the set of points added by this new experiment is subset of the points added by our algorithm since our algorithm adds a point x to M with probability at $\geq \gamma/|O_j| =$ Pr[E_x]. Also, note that the algorithm adds τ copies of F to M to ensure that sufficient number of points are sampled from $c(O_j^{near})$. This implies that in steps 6-7 of the algorithm, the algorithm adds a point μ'' that is $(1 + \varepsilon/16)$-approximation for the 1-service cost of O'_j. Using Lemma 38, this means that M_j contains a subset T_j such that euclid-cost$(\mu(T_j), O_j) \leq \left(1 + \frac{\varepsilon}{2}\right) \cdot \Delta(O_j)$ with probability at least $1/2$. This concludes the proof of Lemma 33, and thus Theorem 48.

4.5 Streaming Algorithms

In this section, we convert the algorithm of the constrained outlier Euclidean k-service problem to a constant-pass log-space streaming algorithm. We use the same techniques that we used in Section 3.8. The offline algorithm for the constrained outlier Euclidean k-service problem has two main components: the list outlier Euclidean k-service algorithm and the outlier partition algorithm. The list outlier Euclidean k-service procedure is common to all constrained versions

of the problem. However, the outlier partition algorithm differs for different constrained versions. We convert the algorithm `List-Outlier-Euclidean-k-Service` to a streaming algorithm in the following manner:

Conversion of `List-Outlier-Euclidean-k-Service` to Streaming algorithm:

1. In the first pass, we run a streaming α-approximation algorithm for the Euclidean k-service instance $(C, k+m, p, z)$. For this, we use the streaming algorithm of Braverman *et al.* [32]. The algorithm runs in polynomial time and gives a constant-approximation with the space complexity of $O((k+m)\cdot \log n)$.
2. In the second pass, we perform the D^z-sampling step using the *reservoir sampling* technique [135].

This gives us the following result:

Lemma 40. *There is a 2-pass streaming algorithm for the list outlier Euclidean k-service problem. The algorithm outputs a list of size $f(k,m,\varepsilon,z)\cdot \log n$. Moreover, the running time of the algorithm is $O(np\cdot f(k,m,\varepsilon))$ and space complexity is $f(k,m,\varepsilon,z)\cdot \log n$, where $f(k,m,\varepsilon,z) = ((k+m)z/\varepsilon)^{O(kz/\varepsilon^{O(z)})}$.*

Now, we need the partition algorithms for the constrained Euclidean k-service problems in the streaming setting. For the chromatic k-service problems, there does not exist any deterministic log-space streaming algorithm [87]. In Section 3.9.4, we showed that this impossibility result also holds for the strongly-private k-service problem. For the ℓ-diversity and fair k-service problems, we neither know any log-space streaming algorithm nor any impossibility result. For the remaining constrained clustering problems mentioned in Table 3.1, we designed the streaming partition algorithms for their outlier and non-outlier versions in Section 3.9. Since

those algorithm are for the general metric spaces, they also holds for the Euclidean space. Together with Lemma 40, we obtain the following results for the constrained k-service problems in continuous Euclidean space:

Theorem 49. *There is a 5-pass streaming algorithm for the outlier version of the r-gather, r-capacity, and balanced Euclidean k-service problems that gives a $(1+\varepsilon)$-approximation guarantee. The algorithm has the space complexity of $O(f(k,m,\varepsilon,z)\cdot\log n)$ and the running time of $O(f(k,m,\varepsilon,z)\cdot n^{O(1)})$, where $f(k,m,\varepsilon,z)=((k+m)z/\varepsilon)^{O(kz/\varepsilon^{O(z)})}$.*

Corollary 18. *There is a 5-pass streaming algorithm for the r-gather, r-capacity, and balanced Euclidean k-service problems that gives a $(1+\varepsilon)$-approximation guarantee. The algorithm has the space complexity of $O(f(k,\varepsilon,z)\cdot\log n)$ and the running time of $O(f(k,\varepsilon,z)\cdot n^{O(1)})$, where $f(k,\varepsilon,z)=(kz/\varepsilon)^{O(kz/\varepsilon^{O(z)})}$.*

Theorem 50. *There is a 3-pass streaming algorithm for the fault-tolerant, ordered-weighted-average, and uncertain Euclidean k-service problems that gives a $(1+\varepsilon)$-approximation guarantee. The algorithm has the space complexity of $O(f(k,\varepsilon,z)\cdot\log n)$ and the running time of $O(f(k,\varepsilon,z)\cdot n)$, where $f(k,\varepsilon,z)=(kz/\varepsilon)^{O(kz/\varepsilon^{O(z)})}$.*

Theorem 51. *There is a 4-pass streaming algorithm for the outlier version of the fault-tolerant, ordered-weighted-average, and uncertain Euclidean k-service problems that gives a $(1+\varepsilon)$-approximation guarantee. The algorithm has the space complexity of $O(f(k,m,\varepsilon,z)\cdot\log n)$ and the running time of $O(f(k,m,\varepsilon,z)\cdot n)$, where $f(k,m,\varepsilon,z)=((k+m)z/\varepsilon)^{O(kz/\varepsilon^{O(z)})}$.*

4.6 Conclusion

Our results demonstrate the versatility of the sampling based approach for k-means. This has also been demonstrated in some of the past works. The effectiveness of k-means++ (which is basically D^2-sampling in k rounds) is well known [16]. The D^2-sampling technique has been used to give simple PTAS for versions of the k-means/median problems with various metric-like distance measures [98] and also various constrained variations of k-means [28]. It has also

been used to give efficient algorithms for coreset construction [115]. In this work, we see its use in the streaming and outlier settings. The nice property of the sampling based approach is that we have a uniform template of the algorithm that is simple and that works in various different settings. This essentially means that the algorithm remains the same while the analysis changes.

This work raises many interesting questions. Our main result on list-k-service is a sampling algorithm that helps us find good centers for *any* subset of k clusters. In the streaming setting for the constrained k-means, we give a generic algorithm within the unified framework of Ding and Xu [72]. The advantage of working in this unified framework is that we get streaming algorithms for various constrained versions of the k-means problem. However, it may be possible to obtain better streaming algorithms (in terms of space, time, and number of passes) for the constrained problems when considered separately as is the case for the classical k-means problem [32]. It may be worthwhile exploring these problems.

Chapter 5

Tight FPT Approximation for Socially Fair k-Median/Means

In this chapter, we study the *socially fair k-median/k-means problem*. We are given a set of points C in a metric space $\mathcal{X}$ with a distance function $d(.,.)$. There are ℓ groups: $C_1, \ldots, C_\ell \subseteq C$. We are also given a set L of feasible centers in $\mathcal{X}$. The goal in the socially fair k-median problem is to find a set $F \subseteq L$ of k centers that minimizes the maximum average cost over all the groups. That is, find F that minimizes the objective function fair-cost$(F, C) \equiv \max_j \left\{ \sum_{x \in C_j} d(F,x)/|C_j| \right\}$, where $d(F,x)$ is the distance of x to the closest center in F. The socially fair k-means problem is defined similarly by using squared distances, i.e., $d^2(.,.)$ instead of $d(.,.)$. The current best approximation guarantee for both the problems is $O\left(\frac{\log \ell}{\log \log \ell}\right)$ due to Makarychev and Vakilian [120]. In this work, we study the fixed parameter tractability of the problems with respect to parameter k. We design $(3+\varepsilon)$ and $(9+\varepsilon)$ approximation algorithms for the socially fair k-median and k-means problems, respectively, in FPT (fixed parameter tractable) time $f(k,\varepsilon) \cdot n^{O(1)}$, where $f(k,\varepsilon) = (k/\varepsilon)^{O(k)}$ and $n = |C \cup L|$. The algorithms are randomized and succeed with probability at least $1 - 1/n$. Furthermore, we show that if W[2] $\neq$ FPT, then better approximation guarantees are not possible in FPT time.

Note that FPT algorithms have polynomial running time if the parameter under consideration is a constant. This may be relevant even to a practitioner since the parameter k is indeed a small number in many real clustering scenarios.

5.1 Overview

In recent years, the topic: *fairness* in *machine learning*, has gained considerable attention, for example, see [25] and [48] for the recent developments in this area. The main motivation is that in many human centric applications, the input data is biased towards a particular demographic group that may be based on age, gender, ethnicity, occupation, nationality, etc. We do not want algorithms to discriminate among different groups due to biases in the dataset. In other words, we aim to design *fair* algorithms for problems.

In the context of clustering, in particular the k-median/k-means/k-center clustering, various notions of fair clustering have recently been proposed (see [46, 26, 11, 27, 104, 45, 116]). Most of these notions are based on *balanced* or *proportionality* clustering. In other words, a clustering is said to be fair if in every cluster, a protected group (e.g. demographic group) occurs in an almost the same proportion as it does in the overall population. By the virtue of this, no group is over-represented or under-represented in any cluster. However, recently, Abbasi *et al.* [1] demonstrated that "balance" based clustering is not desirable in applications where a cluster center represents an entire cluster. One such application is the placement of polling location for voting (see [1] for details). In such applications, the quality of representation of a group is determined by the proximity of the group members to their cluster centers. Such cost representation is not captured by "balance" based clustering. Therefore, they introduced a new notion of the fair clustering where each group has an equitable cost representation in the clustering. Informally, given a point set C and ℓ groups: $C_1, \ldots, C_\ell \subseteq C$, the task is to cluster C into k clusters such that the maximum of the average costs of the groups is minimized. In an independent work, Ghadiri *et al.* [83] used a similar notion that they called the "socially

fair" clustering problem. Recently, Makarychev and Vakilian [120] generalized the definition of the socially fair clustering problem using the weighted point set. The following is a formal definition of the problem as stated in [120].

Definition 44 (Socially Fair Clustering)**.** *We are given a set C of points and set L of feasible centers in a metric space $(\mathcal{X}, d)$. There are ℓ groups (possibly non-disjoint) of points $C_1, ..., C_\ell \subseteq C$ with weight function $w_j \colon C_j \to \mathbb{R}^+$ for each $j \in \{1, \ldots, \ell\}$. Let z be any real number ≥ 1. The unconstrained cost of a group C_j with respect to a center set $F \subseteq L$ is defined as:*

$$\text{service-cost}(F, C_j) \equiv \sum_{x \in C_j} d(F, x)^z \cdot w_j(x), \quad \textit{where } d(F, x) \coloneqq \min_{f \in F} \big\{ d(f, x) \big\}.$$

In socially fair clustering, the goal is to pick a center set $F \subseteq L$ of size k so as to minimize the objective function: $\max_{j \in [\ell]}$ service-cost(F, C_j)*, which we call the fair cost:*

$$\text{fair-cost}(F, C) \equiv \max_{j \in [\ell]} \Big\{ \text{service-cost}(F, C_j) \Big\}.$$

The case of averaging the cost of each group, i.e., $w_j : C_j \to 1/|C_j|$, was initially studied by Ghadiri *et al.* [83] and Abbasi *et al.* [1]. Both these works gave a polynomial time $O(\ell)$-approximation algorithm for the socially fair clustering problem for $z = 1$ and $z = 2$. Furthermore, Abbasi *et al.* [1] showed that the natural LP relaxation of the problem has an integrality gap of $\Omega(\ell)$. To overcome this barrier, Makarychev and Vakilian [120] designed a strengthened LP and improved the approximation guarantee to $O\left(e^{O(z)} \cdot \frac{\log \ell}{\log \log \ell}\right)$ in polynomial time.

Note that in the definition of the socially fair clustering, it is given that groups might not be disjoint. However, we can make the groups disjoint. If a point p appears in multiple groups say $C_{j_1}, \ldots, C_{j_t}$, then we create t copies of point x such that its i^{th} copy only belongs to j_i^{th} group. Moreover, the weight of the i^{th} copy is $w_{j_i}(x)$. The objective function does not change due to

this modification. Therefore, from now on, we will assume all the groups to be disjoint.

Now let us discuss some special cases of the problem. For $z = 1$ and $z = 2$, the problem is known as "socially fair k-median" and "socially fair k-means" problem, respectively. On the other hand, if z is arbitrary and $\ell = 1$, the problem is known as the "k-service" problem (or unconstrained clustering problem). Furthermore, in addition to $\ell = 1$, if $z = 1$ or $z = 2$, the problem becomes the classical (unconstrained) k-median/k-means problem, respectively.

5.2 Our Results

In this work, we study the fixed parameter tractability of the problem parameterized by k. It is known that the classical k-median and k-means problems when parameterized by k, are $\mathsf{W}[2]$-hard [53]. Hence, it immediately implies $\mathsf{W}[2]$-hardness of the socially fair clustering problem. Therefore, the problem does not admit an exact FPT algorithm unless $\mathsf{W}[2] = \mathsf{FPT}$. In this work, we design a $(3^z + \varepsilon)$-approximation algorithm for the problem, with FPT running time, parameterized by k. Furthermore, we show that this approximation guarantee is tight up to an ε additive factor. Also, note that in the running time analysis of the algorithms, we ignore the dependence on z since it is typically considered as constant. Formally, we state the main result as follows:

Theorem 52 (Main Theorem)**.** *Let $z \geq 1$ and $0 \leq \varepsilon \leq 1$ be arbitrary constants. Let $\mathcal{I} = (\mathcal{X}, P, P_1, \ldots, P_\ell, w_1, \ldots, w_\ell, F, d, k, z)$ be arbitrary instance of the socially fair clustering problem and $n = |P \cup F|$. Then, there is a randomized algorithm that outputs a $(3^z + \varepsilon)$-approximate solution to $\mathcal{I}$ with probability at least $1 - 1/n$. The running time of the algorithm is $(kz/\varepsilon)^{O(k)} \cdot n^{O(1)}$, which is FPT in k.*

The following are two immediate corollaries of the above theorem.

Corollary 19 (k-median)**.** *For the socially fair k-median problem, there is a randomized $(3 + \varepsilon)$-approximation algorithm with FPT running time of $(k/\varepsilon)^{O(k)} \cdot n^{O(1)}$ that succeeds with*

probability at least $1 - 1/n$.

Corollary 20 (k-means)**.** *For the socially fair k-means problem, there is a randomized $(9 + \varepsilon)$-approximation algorithm with* FPT *running time of $(k/\varepsilon)^{O(k)} \cdot n^{O(1)}$ that succeeds with probability at least $1 - 1/n$.*

In Section 5.6, we establish FPT hardness of approximation results for the problem that follow from the known hardness results of the unconstrained clustering problems. The following are two main results:

Theorem 53 (FPT Hardness for Parameters: ℓ and k)**.** *For any constant $z \geq 0$, $\varepsilon > 0$, and functions: $g \colon \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$ and $f \colon \mathbb{R}^+ \to \mathbb{R}^+$, the socially fair clustering problem can not be approximated to factor $\left(1 + (3^z - 1)/e - \varepsilon\right)$, in time $g(k, \ell) \cdot n^{f(\ell) \cdot O(1)}$ assuming* FPT $\neq$ W[2] *and in time $g(k, \ell) \cdot n^{f(\ell) \cdot o(k)}$ assuming Gap-ETH.*

Theorem 54 (FPT Hardness for Parameter k)**.** *For any constant $z \geq 0$, $\varepsilon > 0$, and function $g \colon \mathbb{R}^+ \to \mathbb{R}^+$, the socially fair clustering problem can not be approximated to factor $(3^z - \varepsilon)$ in time $g(k) \cdot n^{O(1)}$ assuming* FPT $\neq$ W[2] *and in time $g(k) \cdot n^{o(k)}$ assuming ETH.*

This completes the summary of our results. Note that Theorems 52 and 54 give tight approximation bounds (up to an additive ε additive factor) for the socially fair clustering problem when parameterized by k. Thus, it settles the complexity of the problem when parameterized by k. Next, we compare our work with the previous related works.

5.3 Related Work

The previous works of Abbasi *et al.* [1], and Makarychev and Vakilian [120] were based on the LP relaxation and rounding techniques. Abbasi *et al.* [1] gave $O(\ell)$ approximation guarantee, and Makarychev and Vakilian [120, 47] gave $O\left(\frac{\log \ell}{\log \log \ell}\right)$ approximation guarantees for the

socially fair k-median/means problem. On the other hand, Ghadiri *et al.* [83] designed a socially fair k-means algorithm with performance guarantees similar to the Lloyd's heuristics [114] (popularly known as the *k-means algorithm*).

Recently, Bandyapadhyay *et al.* [21] gave an FPT time constant factor approximation algorithm for a variant of "balance" based fair clustering problem. This variant was first studied by Bera *et al.* [26]; according to this variant, a clustering is said to be fair if within each cluster, the fraction of points that belongs to the j^{th} group is at least β_j and at most α_j, for some constants $0 \leq \alpha_j, \beta_j \leq 1$. This results in fair representation of every group within each cluster. It turns out that this variant falls under a broad class of the *constrained* k-median/k-means problem [21, 72]. Informally, the constrained k-median/k-means problem is a class of clustering problems where a set of constraints can be imposed on the clusters in addition to optimising the k-means/k-median cost. Various other problems like: *uniform capacitated* k-median/k-means problem [112], *outlier* k-median/k-means problem [106], *fault-tolerant* k-median/k-means problem [92], etc., fall in this category.

In Chapter 3, we study that if any constrained k-median/k-means problem admits an FPT time *partition algorithm*, then it also admits an FPT time constant factor approximation algorithm, in general metric spaces. The fair clustering problem studied by Bera *et al.* [26] fits the constrained clustering framework. We studied this clustering variant in Chapters 2 and 3 (see Definition 8 in Table 2.1 and Definition 9 in Table 3.1). Therefore, it is tempting to check if the socially fair clustering problem fits the constrained clustering framework. Unfortunately, the objective function of socially fair clustering differs from the classical k-service and k-supplier objectives. Therefore, the problem can not be treated as a constrained clustering problem. However, we note that the cost function for each group C_j is exactly the same as the k-median/k-means objective. We use this fact to design a polynomial time bi-criteria approximation algorithm for the problem. Then, we convert the bi-criteria approximation algorithm to a constant factor approximation algorithm in FPT time. We will formally define the bi-criteria approximation

algorithm in Section 5.5.1.

Another way of approaching this problem is to obtain a *strong coreset* for the socially fair clustering instance. The coreset can be easily obtained by computing the coresets for each group C_j individually. For the coreset definition and its construction, see the work of Ke Chen [44], or Cohen-Addad *et al.* [56]. After obtaining a coreset of the point set C, one can employ the techniques of Cohen-Addad *et al.* [53], and Cohen-Addad and Li [52] to obtain a constant factor approximation for the problem in FPT time. The main idea is to try all possible k combination of points in the coreset and choose the centers in L that are closest to those points. This gives $\binom{|S|}{k}$ distinct center sets, where $|S|$ is the number of points in the coreset. It can then be shown that the center set that gives the least clustering cost is a $(3^z + \varepsilon)$ approximation for the problem. For details, see Section 2.2 of [52] in the context of k-median and k-means objectives. However, there are an issue with this technique when we deal with the socially fair clustering objective. The issue is that the coreset size would have an ℓ term, where ℓ is the number of groups. Therefore, the running time would have a multiplicative factor of ℓ^k which makes the algorithm not be FPT in k. Also note that ℓ can be as large as $\Omega(n)$. In this work, without using coreset techniques, we design a $(3^z + \varepsilon)$ approximation algorithm for the problem. Moreover, the running time of the algorithm is FPT in k. In the following section, we mention some notations and facts that we use frequently in this chapter.

5.4 Notations and Identities

Let $\mathcal{I} = (\mathcal{X}, C, C_1, \ldots, C_\ell, w_1, \ldots, w_\ell, L, d, k, z)$ be an instance of the socially fair clustering problem. For a weighted set $S \subseteq C$ with weight function $w\colon S \to \mathbb{R}^+$ and a center set $F \subseteq L$, we denote the *unconstrained* clustering cost of S with respect to F by service-cost(F, S), i.e,

$$\text{service-cost}(F, S) \equiv \sum_{x \in S} w(x) \cdot d(F, x)^z, \text{ where } d(F, x) = \min_{f \in F}\{d(x, f)\}.$$

For simplicity, when $S = \{x\}$, we use the notation service-cost(F, x) instead of service-cost$(F, \{x\})$. Similarly, when $F = \{f\}$, we use the notation service-cost(f, S) instead of service-cost$(\{f\}, S)$. In the remaining discussion, we will refer to the *unconstrained clustering cost* simply as *clustering cost*.

We denote the *fair clustering* cost of C with respect to a center set F by

$$\text{fair-cost}(F, C) \equiv \max_{j \in [\ell]} \Big\{ \text{service-cost}(F, C_j) \Big\}.$$

Moreover, we denote the optimal fair clustering cost of C by OPT and optimal fair center set by $F^\star = \{f_1^\star, \ldots, f_k^\star\}$, i.e., fair-cost$(F^\star, C) = \text{OPT}$. We also use the notation $[t]$ to denote a set $\{1, \ldots, t\}$ for any integer $t \geq 1$. We also use the following inequality in our proofs. The inequality is a generalization of the triangle inequality and easily follows from the *power-mean* inequality.

Fact 4 (Approximate Triangle Inequality)**.** *For any $z \geq 1$, and any four points $q, r, s, t \in \mathcal{X}$:* $d(q,t)^z \leq (d(q,r) + d(r,s) + d(s,t))^z \leq 3^{z-1} \cdot (d(q,r)^z + d(r,s)^z + d(s,t)^z)$.

5.5 FPT Approximation

In this section, we design a $(3^z + \varepsilon)$-approximation algorithm for the socially fair clustering problem with FPT time of $(zk/\varepsilon)^{O(k)} \cdot n^{O(1)}$. The algorithm turns out to be surprisingly simple. Our algorithm consists of the following two parts:

1. A polynomial time $\left(1 + \varepsilon, O\left((z/\varepsilon^2) \cdot \ln^2 n\right)\right)$ bi-criteria approximation algorithm for the socially fair clustering problem. The algorithm outputs a center set $F \subseteq L$ of size $O((kz/\varepsilon^2) \cdot \ln^2 n)$ and gives $(1+\varepsilon)$-approximation with respect to the optimal solution with k centers.

2. We then show that there exists a k-sized subset $S \subset F$ that gives $(3^z + \varepsilon)$ approximation.

Note that since one needs to try all possible k-sized subsets of F, the overall running time of the algorithm has a multiplicative factor of $O\binom{|F|}{k}$. This results in an FPT algorithm.

We discuss the above two parts in Sections 5.5.1 and 5.5.2.

5.5.1 Bi-criteria approximation

We start with the definition of (α, β) bi-criteria approximation algorithm:

Definition 45 (Bi-criteria Approximation)**.** *An algorithm is said to be (α, β) bi-criteria approximation for the problem if it outputs a set F of βk centers with fair clustering cost at most α times the optimal fair clustering cost with k centers, i.e.,*

$$\text{fair-cost}(F, C) \leq \alpha \cdot \min_{|F'|=k \text{ and } F' \subseteq L} \Big\{ \text{fair-cost}(F', C) \Big\} = \alpha \cdot \text{OPT}$$

Note that for the *unconstrained* k-median problem, there exists a randomized $\big(1+\varepsilon, O(\ln(1/\varepsilon))\big)$ bi-criteria approximation algorithm due to Neal Young [140]. We extend that algorithm to the socially fair clustering problem and obtain a randomized $\big(1+\varepsilon, O\left((z/\varepsilon^2) \cdot \ln^2 n\right)\big)$ bi-criteria approximation algorithm. Formally, we state the result as follows:

Theorem 55 (Fair Bi-Criteria Approximation)**.** *Let $0 \leq \varepsilon \leq 1$ be any arbitrary constant. Let $\mathcal{I} = (\mathcal{X}, C, C_1, \ldots, C_\ell, w_1, \ldots, w_\ell, L, d, k, z)$ be any instance of the socially fair clustering problem. Then, there exists a polynomial time algorithm that with probability at least $(1-1/n)$ outputs a center set $F \subseteq L$ of size $O\left((kz/\varepsilon^2) \cdot \ln^2 n\right)$ that is a $(1+\varepsilon)$-approximation to the optimal fair clustering cost of $\mathcal{I}$ with k centers. That is,* fair-cost$(F, C) \leq (1+\varepsilon) \cdot$ OPT.

The theorem follows from the next two lemmas: Lemmas 41 and 42.

Lemma 41. *There is a polynomial time randomized algorithm* `Randomized-Fair-Subroutine` *that outputs a center set F' such that for every group $C_j \in \{C_1, \ldots, C_\ell\}$, the*

expected clustering cost of C_j with respect to F' is at most $(1+\varepsilon/2)$ times the optimal fair clustering cost of instance $\mathcal{I}$. That is, for all j,

$$\mathbb{E}\left[\,\text{service-cost}(F', C_j)\,\right] \leq \left(1+\frac{\varepsilon}{2}\right)\cdot \text{OPT}$$

Moreover, the center set F' contains at most $O(kz\ln(n/\varepsilon))$ centers.

The above lemma follows from a modification of the known bi-criteria approximation algorithm for the unconstrained clustering problem [140] which in turn follows from an LP-rounding technique with respect to the most natural linear programming formulation of the problem. We give the outline of the Linear Programming (LP) relaxation and the rounding procedure while deferring the analysis to the Appendix. We start with the natural LP-relaxation for the problem:

$$\begin{aligned}
&\text{minimize} \quad \gamma \\
&\text{subject to} \quad \sum_{f\in L} y_f = k && && (1)\\
&\qquad \sum_{f\in L} x_{f,x} = 1 && \text{for every point } x\in C && (2)\\
&\qquad \sum_{x\in C_j}\sum_{f\in L} x_{f,x}\cdot d(f,x)^z \cdot w_j(x) \leq \gamma && \text{for every } C_j\in\{C_1,\ldots,C_\ell\} && (3)\\
&\qquad x_{f,x} \leq y_f && \text{for every } f\in L \text{ and } x\in C && (4)\\
&\qquad y_f, x_{f,x} \geq 0 && \text{for every } f\in L \text{ and } x\in C && (5)
\end{aligned}$$

Here, y_f is a variable that denote the fraction of a center f picked in the solution. The variable $x_{f,x}$ denote the fraction of point x assigned to center f. The variable γ denote the fair clustering cost of a feasible fractional solution. We solve the above linear program to obtain the fractional

optimal solution: y_f^*, $x_{f,x}^*$, and γ^*. Since it is a relaxation to the original problem, $\gamma^* \leq \text{OPT}$. For simplicity, we use the notations: y_f, $x_{f,x}$, γ for y_f^*, $x_{f,x}^*$, and γ^*, respectively.

`Randomized-Fair-Subroutine` ($\mathcal{I}$, y_f's, $x_{f,x}$'s, ε)

Inputs: Socially fair clustering instance $\mathcal{I}$, fractional optimal solution: y_f's and $x_{f,x}$'s of the relaxed LP, and accuracy $\varepsilon \leq 1$.

Output: A center set $F' \subseteq L$ of size $O(k\, z \ln(n/\varepsilon))$ such that $\mathbb{E}\left[\text{service-cost}(F', C_j)\right] \leq (1+\varepsilon/2) \cdot \text{OPT}$ for every group $C_j \in \{C_1, \ldots, C_\ell\}$

(1) $F' \leftarrow \emptyset$ *(center set)*
(2) $C_u \leftarrow C$ *(set of unassigned points)*
(3) Repeat $k\,(z\,c + \ln(2n/\varepsilon))$ times for some constant c:
(4) Sample a center $f \in L$ with probability y_f/k. Let f^* be the sampled center.
(5) $F' \leftarrow F' \cup \{f^*\}$
(6) For each point $x \in C_u$:
(7) Assign x to f^* with probability $x_{f^*,x}/y_{f^*}$
(8) If x assigned to f^*, then $C_u \leftarrow C_u \setminus \{x\}$
(9) Run $O\left(e^{O(z)} \cdot \frac{\log \ell}{\log\log \ell}\right)$-approximation algorithm for socially fair clustering problem on C. Let $F_u \subseteq L$ be the obtained center set. Assign the points in C_u to F_u.
(10) $F' \leftarrow F' \cup F_u$
(11) return(F')

Algorithm 5.1: A rounding procedure used as a subroutine for bi-criteria approximation.

The randomized subroutine is described in Algorithm 5.1. The algorithm takes input the fractional optimal solution to the relaxed LP of the socially fair clustering problem. In Line (1), the algorithm initializes a center set F' as empty. In Line (2), the algorithm initializes a set C_u that denotes the set of unassigned points. Initially, no point is assigned to any center; therefore C_u is initialized to C. Then, the algorithm proceeds in two phases:

- Phase 1 constitutes Lines $(3) - (8)$ of the algorithm. In this phase, the algorithm samples a center from L with probability distribution defined by y_f/k. Note that the sum of probabilities over all $f \in L$, is 1 due to constraint (1) of the relaxed LP. Therefore, a center is always selected, and it added to F'. Suppose the selected center is f^*. Then, for each point $x \in C_u$, the algorithm independently assigns x to f^* with probability $x_{f^*,x}/y_{f^*}$.

The algorithm removes the points from C_u that are assigned to f^*. The algorithm repeats this procedure $k\,(z\,c + \ln(2n/\varepsilon))$ times. Here, c is a constant whose value will be defined later during the analysis of the algorithm.

- Phase 2 constitutes line (9) of the algorithm. In this phase, the algorithm runs the $O\left(e^{O(z)} \cdot \frac{\log \ell}{\log\log \ell}\right)$-approximation algorithm $\mathcal{A}$ of Makarychev and Vakilian [120] for the socially fair clustering problem on the entire point set C. Let F_u be the center set output by algorithm $\mathcal{A}$. The remaining points in C_u are assigned to their closest centers in F_u. After this phase, all the points are assigned. Lastly, the algorithm returns all the centers selected in Phase 1 and Phase 2.

Next, we give proof of Lemma 41. It mainly involves analysis of `Randomized-Fair-Subroutine`.

Proof. We bound the expected assignment cost of each group C_j with respect to F', the center set returned by `Randomized-Fair-Subroutine`. Let the centers selected in Phase 1 of the algorithm be: $F_f \coloneqq \{f_1^*, \ldots, f_t^*\}$, for $t = k\,(z\,c + \ln(2n/\varepsilon))$. In other words, f_i^* is the center sampled in the i^{th} iteration of Line (3) of the subroutine. For each point $x \in C_j$ and iteration $i \in \{1, \ldots, t\}$, we define a random variable A_x^i. It takes value 1 if x is unassigned after i iterations; otherwise it is 0. In other words, $A_x^i = 1$ if $x \in C_u$ after i iterations. Given $A_x^i = 1$, the probability that x is assigned to some center in $(i+1)^{th}$ iteration is:

$$\mathbf{Pr}[A_x^{i+1} = 0 \mid A_x^i = 1] = \sum_{f \in L} \mathbf{Pr}[x \text{ is assigned to } f \mid f_{i+1}^* = f] \cdot \mathbf{Pr}[f_{i+1}^* = f]$$

$$= \sum_{f \in L} \frac{x_{f,x}}{y_f} \cdot \frac{y_f}{k} = \frac{\sum_{f \in L} x_{f,x}}{k} = \frac{1}{k} \tag{5.1}$$

The last equality follows from the second constraint of relaxed LP. Also, note that $\mathbf{Pr}[A_x^1 = 0] = \frac{1}{k}$ using the above same analysis since the point was unassigned before the first iteration.

Now, we show that the probability that x is unassigned after i iterations is $\left(1 - \frac{1}{k}\right)^i$, i.e., $\mathbf{Pr}[A_x^i = 1] = \left(1 - \frac{1}{k}\right)^i$, for every $i \in \{1, \ldots, t\}$. We prove this statement using induction on i:

Base Case: For $i = 1$, the probability that x is unassigned after the first iteration is:

$$\mathbf{Pr}[A_x^1 = 1] = 1 - \mathbf{Pr}[A_x^1 = 0] = 1 - \frac{1}{k}$$

Induction Step: For $i > 1$, the probability that x is unassigned after i iterations is:

$$\mathbf{Pr}[A_x^i = 1] = \mathbf{Pr}[A_x^i = 1 \mid A_x^{i-1} = 1] \cdot \mathbf{Pr}[A_x^{i-1} = 1] + \mathbf{Pr}[A_x^i = 1 \mid A_x^{i-1} = 0] \cdot \mathbf{Pr}[A_x^{i-1} = 0],$$

(using the law of total probability)

Note that $\mathbf{Pr}[A_x^i = 1 \mid A_x^{i-1} = 0] = 0$ since $x \notin C_u$ after $(i-1)^{th}$ iteration; therefore it does not participate in i^{th} iteration. Hence we get,

$$\begin{aligned}
\mathbf{Pr}[A_x^i = 1] &= \mathbf{Pr}[A_x^i = 1 \mid A_x^{i-1} = 1] \cdot \mathbf{Pr}[A_x^{i-1} = 1] \\
&= \left(1 - \mathbf{Pr}[A_x^i = 0 \mid A_x^{i-1} = 1]\right) \cdot \mathbf{Pr}[A_x^{i-1} = 1] \\
&= \left(1 - \frac{1}{k}\right) \cdot \mathbf{Pr}[A_x^{i-1} = 1], && \text{(using Equation (5.1))} \\
&= \left(1 - \frac{1}{k}\right) \cdot \left(1 - \frac{1}{k}\right)^{i-1}, && \text{(using Induction Hypothesis)} \\
&= \left(1 - \frac{1}{k}\right)^i
\end{aligned}$$

This proves that $\mathbf{Pr}[A_x^i = 1] = \left(1 - \frac{1}{k}\right)^i$, for every $i \in \{1, \ldots, t\}$.

Now, we evaluate the expected assignment cost for each group $C_j \in \{C_1, \ldots, C_\ell\}$. Let α_x denote the assignment cost of each point $x \in C_j$ in the fractional optimal solution. That is, $\alpha_x = \sum_{f \in L} x_{f,x} \cdot d(f,x)^z \cdot w_j(x)$. For every point $x \in C_j$ and center $f \in L$, let $E_{f,x}$ denote the event that x is assigned to f during Phase 1. Let $E^{\,i}_{f,x}$ denote the event that x is assigned to f in the i^{th} iteration, during Phase 1. If x remains unassigned after Phase 1, then let β_x denote the cost of x during Phase 2. Then, the expected cost of group C_j with respect to the center set F' is:

$\mathbb{E}[\text{service-cost}(F', C_j)]$

$$= \sum_{x \in C_j} \mathbb{E}[\text{service-cost}(F', x)] \quad (\text{using linearity of expectation})$$

$$= \sum_{x \in C_j} \left(\sum_{f \in L} \mathbf{Pr}[E_{f,x}] \cdot d(f,x)^z \cdot w_j(x) + \mathbf{Pr}[A_x^t = 1] \cdot \beta_x \right)$$

$$= \sum_{x \in C_j} \left(\sum_{f \in L} \sum_{i=1}^{t} \mathbf{Pr}[E^{\,i}_{f,x}] \cdot d(f,x)^z \cdot w_j(x) + \mathbf{Pr}[A_x^t = 1] \cdot \beta_x \right) \tag{5.2}$$

Next, we show that $\sum_{f \in L} \sum_{i=1}^{t} \mathbf{Pr}[E^{\,i}_{f,x}] \cdot d(f,x)^z \cdot w_j(x) \leq \alpha_x$ for every point $x \in C_j$.

$\sum_{f \in L} \sum_{i=1}^{t} \mathbf{Pr}[E^{\,i}_{f,x}] \cdot d(f,x)^z \cdot w_j(x)$

$$\leq \sum_{f \in L} \sum_{i=1}^{t} \mathbf{Pr}[\, E^{\,i}_{f,x} \mid A_x^{i-1} = 1] \cdot \mathbf{Pr}[A_x^{i-1} = 1] \cdot d(f,x)^z \cdot w_j(x)$$

$$
\begin{aligned}
&= \sum_{f\in L}\sum_{i=1}^{t} \mathbf{Pr}[\, E^{i}_{f,x} \mid A_x^{i-1}=1] \cdot \left(1-\frac{1}{k}\right)^{i-1} \cdot d(f,x)^z \cdot w_j(x) \\
&= \sum_{f\in L}\sum_{i=1}^{t} \frac{x_{f,x}}{y_f} \cdot \frac{y_f}{k} \cdot \left(1-\frac{1}{k}\right)^{i-1} \cdot d(f,x)^z \cdot w_j(x) \\
&= \sum_{i=1}^{t} \frac{\sum_{f\in L} x_{f,x} \cdot d(f,x)^z \cdot w_j(x)}{k} \cdot \left(1-\frac{1}{k}\right)^{i-1} \\
&= \sum_{i=1}^{t} \frac{\alpha_x}{k} \cdot \left(1-\frac{1}{k}\right)^{i-1} \\
&= \frac{\alpha_x}{k} \cdot \frac{1-\left(1-\frac{1}{k}\right)^t}{1/k} \le \alpha_x
\end{aligned}
$$

This proves that $\sum_{f\in L}\sum_{i=1}^{t} \mathbf{Pr}[E^{i}_{f,x}] \cdot d(f,x)^z \cdot w_j(x) \le \alpha_x$. Now, substituting this inequality in Equation (5.2), we get

$\mathbb{E}[\text{service-cost}(F', C_j)]$

$$
\begin{aligned}
&\le \sum_{x\in C_j} \alpha_x + \sum_{x\in C_j} \mathbf{Pr}[A_x^t = 1] \cdot \beta_x \\
&= \sum_{x\in C_j} \alpha_x + \sum_{x\in C_j} \left(1-\frac{1}{k}\right)^t \cdot \beta_x \\
&= \sum_{x\in C_j} \alpha_x + \left(1-\frac{1}{k}\right)^t \cdot O\left(e^{O(z)} \cdot \frac{\log \ell}{\log\log \ell}\right) \cdot \mathbf{OPT}, \\
&\qquad\qquad (\because O\left(e^{O(z)} \cdot \frac{\log \ell}{\log\log \ell}\right)\text{-approximation algorithm used in Phase 2}) \\
&= \sum_{x\in C_j} \alpha_x + \left(1-\frac{1}{k}\right)^{k\,(z\,c+\ln(2n/\varepsilon))} \cdot O\left(e^{O(z)} \cdot \frac{\log \ell}{\log\log \ell}\right) \cdot \mathbf{OPT}, \\
&\qquad\qquad (\because t = k\,(z\,c+\ln(2n/\varepsilon))) \\
&\le \sum_{x\in C_j} \alpha_x + \frac{\varepsilon}{e^{cz}\cdot 2\cdot n} \cdot O\left(e^{O(z)} \cdot \frac{\log \ell}{\log\log \ell}\right) \cdot \mathbf{OPT}, \qquad \left(\because \left(1-\frac{1}{k}\right)^k \le 1/e\right)
\end{aligned}
$$

$$\leq \sum_{x \in C_j} \alpha_x + \frac{\varepsilon}{e^{cz} \cdot 2 \cdot n} \cdot e^{c'z} \cdot \frac{\log \ell}{\log \log \ell} \cdot \text{OPT}, \qquad (\text{for some constant } c' > 0)$$

$$\leq \sum_{x \in C_j} \alpha_x + \frac{\varepsilon}{e^{cz} \cdot 2} \cdot e^{c'z} \cdot \text{OPT}, \qquad (\because \ell \leq n)$$

$$= \sum_{x \in C_j} \alpha_x + \frac{\varepsilon}{2} \cdot \text{OPT}, \qquad (\text{we choose } c \text{ such that } c = c')$$

$$\leq \gamma + \frac{\varepsilon}{2} \cdot \text{OPT}, \qquad (\text{using constraint (3) of the relaxed LP})$$

$$\leq \left(1 + \frac{\varepsilon}{2}\right) \cdot \text{OPT}$$

This completes the proof of Lemma 41.

□

We now apply standard probability amplification method to bound the fair-cost.

Lemma 42. *Suppose* `Randomized-Fair-Subroutine` *is repeated* $r = \frac{8 \ln n}{\varepsilon}$ *times, independently. Let* $F'_1, \ldots, F'_r$ *be the obtained center sets for each call to the algorithm. Then, the center set* $F \coloneqq F'_1 \cup \ldots \cup F'_r$ *is a* $(1+\varepsilon)$ *approximation to the optimal fair clustering cost of* C*, i.e.,* fair-cost$(F, C) \leq (1+\varepsilon) \cdot$ OPT*, with probability at least* $1 - 1/n$*.*

Proof. We say that a group C_j *violates* the fairness bound with respect to some center set F'_i, if service-cost$(F'_i, C_j) > (1+\varepsilon) \cdot$ OPT. Now, note that for every center set $F'_i \in \{F'_1, \ldots, F'_r\}$ and group $C_j \in \{C_1 \ldots, C_\ell\}$, we have that $\mathbb{E}[\text{service-cost}(F'_i, C_j)] \leq (1 + \varepsilon/2) \cdot \text{OPT}$, using Lemma 41. Furthermore, using the Markov's inequality, we get the following probability bound:

$$\textbf{Pr}\left[\,\text{service-cost}(F'_i, C_j) > (1+\varepsilon) \cdot \text{OPT}\,\right] < \frac{1 + \varepsilon/2}{1+\varepsilon} = 1 - \frac{\varepsilon/2}{1+\varepsilon} \leq 1 - \frac{\varepsilon}{4}, \quad \text{for } \varepsilon \leq 1$$

In other words, C_j violates the fairness bound with respect to $F_i^{'}$ with probability at most $1 - \varepsilon/4$. Then, the probability that C_j violates the fairness bound with respect to every center set $F_i' \in \{F_1', \ldots, F_r'\}$ is:

$$\begin{aligned}\textbf{Pr}\left[\,\forall i \in \{1,\ldots,r\},\ \text{service-cost}(F_i', C_j) > (1+\varepsilon)\cdot \text{OPT}\,\right] &< \left(1 - \frac{\varepsilon}{4}\right)^r, \\ &\qquad (\because \text{Independent events}) \\ &\leq \frac{1}{n^2}, \quad \left(\because r = \frac{8\ln n}{\varepsilon}\right)\end{aligned}$$

Since $F \coloneqq F_1' \cup \ldots \cup F_r'$, we get

$$\begin{aligned}&\textbf{Pr}\left[\,\text{service-cost}(F, C_j) > (1+\varepsilon)\cdot \text{OPT}\,\right] \\ &\qquad \leq \textbf{Pr}\left[\,\forall i \in \{1,\ldots,r\},\ \text{service-cost}(F_i', C_j) > (1+\varepsilon)\cdot \text{OPT}\,\right] \\ &\qquad < \frac{1}{n^2}\end{aligned}$$

In other words, C_j violates the fairness bound with respect to F with probability at most $1/n^2$. Then, the probability that at least one of the groups in $\{C_1, \ldots, C_\ell\}$ violates the fairness bound with respect to F is:

$$\begin{aligned}\textbf{Pr}\left[\,\exists C_j \in \{C_1,\ldots,C_\ell\},\ \text{service-cost}(F, C_j) > (1+\varepsilon)\cdot \text{OPT}\,\right] &< \frac{\ell}{n^2}, \quad \text{(using union bound)} \\ &\leq 1/n \quad (\because \ell \leq n)\end{aligned}$$

Therefore, the probability that none of the groups in $\{C_1, \ldots, C_\ell\}$ violate the fairness bound with respect to F is:

$$\mathbf{Pr}\left[\,\forall C_j \in \{C_1,\ldots,C_\ell\},\ \text{service-cost}(F,C_j) \le (1+\varepsilon)\cdot \text{OPT}\,\right] \ge 1-\frac{1}{n}$$

This gives the following probability bound on the fair clustering cost of C:

$$\begin{aligned}
&\mathbf{Pr}\left[\text{fair-cost}(F,C) \le (1+\varepsilon)\cdot \text{OPT}\right] \\
&\quad= \mathbf{Pr}\left[\max_{j\in[\ell]}\Big\{\text{service-cost}(F,C_j)\Big\} \le (1+\varepsilon)\cdot \text{OPT}\right] \\
&\quad= \mathbf{Pr}\Big[\,\forall C_j \in \{C_1,\ldots,C_\ell\},\ \text{service-cost}(F,C_j) \le (1+\varepsilon)\cdot \text{OPT}\,\Big] \\
&\quad\ge 1-\frac{1}{n}
\end{aligned}$$

This proves that F is a $(1+\varepsilon)$ approximation to the optimal fair clustering cost of C with probability at least $1-1/n$. This completes the proof of the lemma.

□

Note that $r = \frac{8\ln n}{\varepsilon}$ and $|F_i'| = O(kz\ln(n/\varepsilon))$ for every $i \in \{1,\ldots,r\}$. This gives $|F| = O\left((kz/\varepsilon^2)\cdot \ln^2 n\right)$, for $\varepsilon \le 1$. This proves Theorem 55.

5.5.2 Conversion: *Bi-criteria to FPT approximation*

In this subsection, we convert the $\left(1+\varepsilon, O\left((z/\varepsilon^2)\cdot\ln^2 n\right)\right)$ bi-criteria approximation algorithm to $(3^z+\varepsilon)$-approximation algorithm in FPT time.

Lemma 43. *Let $F = \{f_1,\ldots,f_{\beta k}\} \subseteq L$ be any (α,β)-approximate solution to the socially fair clustering instance $\mathcal{I} = (\mathcal{X}, C, C_1,\ldots,C_\ell, w_1,\ldots,w_\ell, L, d, k, z)$. Then, there exists a k sized subset F' of F that is a $(3^{z-1}\cdot(\alpha+2))$-approximate solution to $\mathcal{I}$. Moreover, given F, the center set F' can be obtained in time $O((e\beta)^k\cdot nk)$.*

Proof. Let $F^* = \{f_1^*, \ldots, f_k^*\} \subseteq L$ be an optimal center set of $\mathcal{I}$. This set induces a Voronoi partitioning in each of the groups. We denote this partitioning using the notation $\mathbb{C}_j = \{C_j^1, C_j^2, \ldots, C_j^k\}$ for the j^{th} group. That is, C_j^i are the set of those points in group C_j for which the center $f_i^\star$ is the closest. For any point $x \in C \cup L$, let $g(x)$ denote the point in F that is closest to x. That is, $g(x) \coloneqq \arg\min_{f \in F} \big\{d(f,x)\big\}$. We define a new center set $F' \coloneqq \{g(f_1^*), \ldots, g(f_k^*)\} \subseteq F$. We show that F' is a $(3^{z-1} \cdot (\alpha+2))$-approximate solution to $\mathcal{I}$. The proof follows from the following sequence of inequalities:

fair-cost(F', C)

$$
\begin{aligned}
&= \max_{j \in [\ell]} \left[\text{service-cost}(F', C_j)\right] \\
&\leq \max_{j \in [\ell]} \left[\sum_{i=1}^{k} \text{service-cost}(g(f_i^*), C_j^i)\right] \\
&= \max_{j \in [\ell]} \left[\sum_{i=1}^{k} \sum_{x \in C_j^i} w_j(x) \cdot d(g(f_i^*), x)^z\right] \\
&\leq \max_{j \in [\ell]} \left[\sum_{i=1}^{k} \sum_{x \in C_j^i} w_j(x) \cdot (d(x, f_i^\star) + d(f_i^\star, g(f_i^*)))^z\right], \quad \text{(using triangle inequality)} \\
&\leq \max_{j \in [\ell]} \left[\sum_{i=1}^{k} \sum_{x \in C_j^i} w_j(x) \cdot (d(x, f_i^\star) + d(f_i^\star, g(x)))^z\right], \quad \text{(from definition of } g(f_i^*)) \\
&\leq \max_{j \in [\ell]} \left[\sum_{i=1}^{k} \sum_{x \in C_j^i} w_j(x) \cdot (d(x, f_i^\star) + d(x, f_i^\star) + d(x, g(x)))^z\right], \quad \text{(using triangle inequality)} \\
&\leq \max_{j \in [\ell]} \left[3^{z-1} \cdot \sum_{i=1}^{k} \sum_{x \in C_j^i} w_j(x) \cdot (d(x, f_i^\star)^z + d(x, f_i^\star)^z + d(x, g(x))^z)\right], \quad \text{(using Fact 4)}
\end{aligned}
$$

$$
\leq \max_{j \in [\ell]} \left[3^{z-1} \cdot 2 \cdot \sum_{i=1}^{k} \sum_{x \in C_j^i} w_j(x) \cdot d(x, f_i^\star)^z\right] + \max_{j \in [\ell]} \left[3^{z-1} \cdot \sum_{i=1}^{k} \sum_{x \in C_j^i} w_j(x) \cdot d(x, g(x))^z\right]
$$

$$= \max_{j\in[\ell]} \left[2\cdot 3^{z-1} \cdot \sum_{i=1}^{k} \text{service-cost}(f_i^\star, C_j^i) \right] + \max_{j\in[\ell]} \left[3^{z-1} \cdot \text{service-cost}(F, C_j) \right]$$

$$\leq 2\cdot 3^{z-1} \cdot \text{OPT} + \max_{j\in[\ell]} \left[3^{z-1} \cdot \text{service-cost}(F, C_j) \right]$$

$$\leq 2\cdot 3^{z-1} \cdot \text{OPT} + 3^{z-1} \cdot \alpha \cdot \text{OPT}, \qquad (\because F \text{ is an } \alpha\text{-approximate solution})$$

$$= 3^{z-1} \cdot (\alpha+2) \cdot \text{OPT}.$$

This proves that F' is a $(3^{z-1} \cdot (\alpha+2))$-approximate solution to $\mathcal{I}$.

Now, we find the center set F' using F. Since, we do not know F^*, we can not directly find F'. Therefore, we take all possible k sized subsets of F and compute the fair clustering cost for each of them. We output that center set that gives the least fair clustering cost. There are at most $\binom{\beta k}{k} \leq (e\beta)^k$ possibilities [1] for F'. For each such center set, the fair clustering cost can be computed in $O(nk)$ time using the Voronoi partitioning algorithm. Therefore, the overall running time is $O((e\beta)^k \cdot nk)$. This completes the proof of the lemma. □ □

In the above lemma, we substitute the bi-criteria approximation algorithm that we designed in the previous subsection. It had $\alpha = 1+\varepsilon$ and $\beta = O\left((z/\varepsilon^2)\cdot \ln^2 n\right)$. Moreover, we set $\varepsilon = \varepsilon'/3^{z-1}$ for some constant $\varepsilon' \leq 1$. Then, the above lemma gives the following main result for the fair clustering problem:

Corollary 21 (Main Result (Restatement of Theorem 52))**.** *Let $0 \leq \varepsilon \leq 1$ be any arbitrary constant. Let $\mathcal{I} = (\mathcal{X}, C, C_1, \ldots, C_\ell, w_1, \ldots, w_\ell, L, d, k, z)$ be any arbitrary instance of the socially fair clustering problem and $n = |C \cup L|$. Then, there is an algorithm that outputs a $(3^z + \varepsilon')$-approximate solution for $\mathcal{I}$ with probability at least $1 - 1/n$. The running time of the algorithm is $(kz/\varepsilon')^{O(k)} \cdot n^{O(1)}$, which is* FPT *in k.*

[1] Here, we use a well known inequality that $\binom{n}{k} \leq (\frac{e\cdot n}{k})^k$.

Proof. It is easy to see that the approximation guarantee of the algorithm is $(3^z + \varepsilon')$ since $\alpha = 1+\varepsilon$ and $\varepsilon = \varepsilon'/3^{z-1}$. Note that the bi-criteria approximation algorithm has polynomial running time. Furthermore, converting the bi-criteria approximation algorithm to FPT approximation algorithm requires $O((e(z/\varepsilon^2)\cdot\ln^2 n)^k \cdot nk)$ time since $\beta = O\left((z/\varepsilon^2)\cdot\ln^2 n\right)$. Using a standard inequality that $(\ln n)^k = k^{O(k)} \cdot n$ (for proof, see Hint 3.18 from the book: Parameterized Algorithms [63]), we get total running time of $(kz/\varepsilon')^{O(k)} \cdot n^{O(1)}$, which is FPT in k. Hence proved. □ □

5.6 FPT Lower Bounds

In this section, we establish the FPT hardness of approximation results for the socially fair clustering problem. For this, we use the hardness results of the unconstrained k-median and k-means problems. Firstly, note the following result from [121].

Theorem 56 (Corollary 3 of [121])**.** *For any constant $\varepsilon > 0$ and any function $g\colon \mathbb{R}^+ \to \mathbb{R}^+$, the k-median and k-means problems can not be approximated to factors $(1 + 2/e - \varepsilon)$ and $(1 + 8/e - \varepsilon)$, respectively, in time $g(k) \cdot n^{o(k)}$, assuming Gap-ETH.*

The above hardness result also holds under the assumption that W[2] $\neq$ FPT; however, with weaker running time lower bound of $g(k) \cdot n^{O(1)}$. Furthermore, the above result can be easily extended to general value of z (see Section 9.3 of [121]). It gives the following theorem:

Theorem 57 ([121])**.** *For any constant $z \geq 0$, $\varepsilon > 0$, and any function $g\colon \mathbb{R}^+ \to \mathbb{R}^+$, the unconstrained k-service problem can not be approximated to factor $(1 + (3^z - 1)/e - \varepsilon)$, in time $g(k) \cdot n^{o(k)}$, assuming Gap-ETH and in time $g(k) \cdot n^{O(1)}$, assuming* W[2] $\neq$ FPT.

It is easy to see that for $\ell = 1$, the socially fair clustering problem is equivalent to the unconstrained k-service problem. This immediately gives the following hardness result for the socially fair clustering problem.

Theorem 53 (FPT Hardness for Parameters: ℓ and k)**.** *For any constant $z \geq 0$, $\varepsilon > 0$, and functions: $g\colon \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$ and $f\colon \mathbb{R}^+ \to \mathbb{R}^+$, the socially fair clustering problem can not be approximated to factor $\left(1 + (3^z - 1)/e - \varepsilon\right)$, in time $g(k,\ell)\cdot n^{f(\ell)\cdot O(1)}$ assuming $\mathsf{FPT} \neq \mathsf{W}[2]$ and in time $g(k,\ell) \cdot n^{f(\ell)\cdot o(k)}$ assuming Gap-ETH.*

Proof. For the sake of contradiction, assume that there exists a constant $\varepsilon > 0$, and functions: $g\colon \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$ and $f\colon \mathbb{R}^+ \to \mathbb{R}^+$ such that the socially fair clustering problem can be approximated to factor $(1+(3^z-1)/e-\varepsilon)$ in time $g(k,\ell)\cdot n^{f(\ell)\cdot o(k)}$ or $g(k,\ell)\cdot n^{f(\ell)\cdot O(1)}$. Then, for $\ell = 1$, it implies that the unconstrained k-service problem can be approximated to factor $(1+(3^z-1)/e-\varepsilon)$ in time $g(k,1)\cdot n^{o(k)}$ or $g(k,1)\cdot n^{O(1)}$. This contradicts Theorem 57. Hence proved. □

The above hardness result assumed the parametrization by k and ℓ. Now we show stronger hardness result for the problem when it is parameterized by k alone. We show this using a reduction from the k-*supplier* problem. The k-supplier problem is defined as follows:

Definition 46 (k-Supplier Problem)**.** *Let z be any positive real number and k be any positive integer. Given a set C of points and set L of feasible centers in a metric space $(\mathcal{X}, d)$, find a set $F \subseteq L$ of k centers that minimizes the objective function* supplier-cost(F, C) *defined as follows:*

$$\text{supplier-cost}(F,C) \equiv \max_{x\in C}\Big\{d(F,x)^z\Big\}, \quad \textit{where} \quad d(F,x) = \min_{f\in F}\{d(f,x)\}$$

Hochbaum and Shmoys [94] showed that for $z = 1$, the k-supplier problem is NP-hard to approximate to any factor smaller than 3. The proof follows from the reduction from the *hitting set problem* (see Theorem 6 of [94]). A similar reduction is possible from the *set coverage problem* [2]. For the sake of completeness, we describe the reduction here:

[2] In the literature, a variant of this problem is called *max-k-coverage problem*.

The set coverage problem is defined as follows.

Definition 47 (Set Coverage)**.** *Given an integer $k > 0$, a set U, and a collection $\mathscr{C} = \{S_1, \ldots, S_m\}$ of subsets of U, i.e., $S_j \subseteq U$ for every $j \in [m]$, determine if there exist k sets in $\mathscr{C}$ that cover all elements in U.*

Now we describe the reduction. Given a set coverage instance $(U, \mathscr{C}, k)$, we construct a k-supplier instance (C, L, d, k) as follows. For every set $S_i \in \mathscr{C}$, we define a center $f_i \in L$. For every element $e \in U$, we define a point $x_e \in C$. Let us define the distance function $d(.,.)$ as follows. For any two points $x_e, x_{e'} \in C$, or $f_i, f_j \in L$, the distance $d(x_e, x_{e'}) = d(f_i, f_j) = 2$. For any point $x_e \in C$ and $f_i \in L$, if $e \notin S_i$, the distance $d(x_e, f_i) = 3$; otherwise $d(x_e, f_i) = 1$. Furthermore, assume that $d(.,.)$ is a symmetric function, i.e., $d(x, y) = d(y, x)$ for every $x, y \in C \cup L$. Also assume that $d(x, x) \geq 0$ for every $x \in C \cup L$. It is easy to see that $d(.,.)$ satisfies all the properties of a metric space.

Now, suppose that there exist k sets: $S_{i_1}, \ldots, S_{i_k}$ in $\mathscr{C}$ that cover all elements of U, i.e., $S_{i_1} \cup \ldots \cup S_{i_k} = U$, then the center set $F = \{f_{i_1}, \ldots, f_{i_k}\}$ gives the k-supplier cost 1. On the other hand, if there does not exist any k sets in $\mathscr{C}$ that could cover all elements of U, then for any center set $F \subseteq L$ of size k there would exist a point $x \in C$ at a distance of 3 from F, i.e., $d(F, x) = 3$. Therefore, the k-supplier cost would be 3^z. Since the set coverage problem is $\mathsf{W}[2]$-hard, it implies that the k-supplier problem can not be approximated to any factor better than 3^z, in polynomial time, assuming $\mathsf{W}[2] \neq \mathsf{FPT}$. Moreover, the following FPT hardness result for the set coverage problem follows from [128]: [3]

Theorem 58 ([128])**.** *For any function $g\colon \mathbb{R}^+ \to \mathbb{R}^+$, there is no $g(k) \cdot n^{o(k)}$ time algorithm for the set coverage problem assuming ETH, and no $g(k) \cdot n^{O(1)}$ time algorithm assuming $\mathsf{W}[2] \neq \mathsf{FPT}$.*

This implies the following FPT hardness of approximation for the k-supplier problem.

[3]The result follows from a trivial reduction from k-dominating set problem to the set coverage problem [101]

Theorem 59. *For any constant $z \geq 0$, $\varepsilon > 0$, and any function $g\colon \mathbb{R}^+ \to \mathbb{R}^+$, the k-supplier problem can not be approximated to factor $(3^z - \varepsilon)$ in time $g(k) \cdot n^{o(k)}$ assuming ETH, and in time $g(k) \cdot n^{O(1)}$ assuming $\mathsf{W}[2] \neq \mathsf{FPT}$.*

Using this, we easily get the following hardness result for the socially fair clustering problem.

Theorem 54 (FPT Hardness for Parameter k)**.** *For any constant $z \geq 0$, $\varepsilon > 0$, and function $g\colon \mathbb{R}^+ \to \mathbb{R}^+$, the socially fair clustering problem can not be approximated to factor $(3^z - \varepsilon)$ in time $g(k) \cdot n^{O(1)}$ assuming $\mathsf{FPT} \neq \mathsf{W}[2]$ and in time $g(k) \cdot n^{o(k)}$ assuming ETH.*

Proof. We prove this result by showing that the k-supplier problem is a special case of the socially fair clustering problem. Let $\mathcal{I} = (C, C_1, \ldots, C_\ell, w_1, \ldots, w_\ell, L, d, k, z)$ be an instance of the socially fair clustering problem defined in the following manner. The number of groups is the same as the number of points, i.e., $\ell = |C|$. For every point $x \in C$, we define a singleton group C_x as $C_x \coloneqq \{x\}$. Let the weight function be defined as $w_x\colon C_x \to 1$ for every $x \in C$, i.e., each point carries a unit weight. Then, for any center set $F \subseteq L$, the fair clustering cost of C is:

$$\text{fair-cost}(F, C) = \max_{x \in C} \Big\{ \text{service-cost}(F, C_x) \Big\} = \max_{x \in C} \Big\{ d(F, x)^z \Big\}$$

Recall that $\max_{x \in C} \Big\{ d(F, x)^z \Big\} \equiv$ supplier-cost(F, C) is the k-supplier cost of the instance (C, L, d, k). It means that the fair cost of instance $\mathcal{I}$ is the same as the k-supplier cost of the instance (C, L, d, k). Therefore, the k-supplier problem is a special case of the socially fair clustering problem. Therefore, the hardness result stated in Theorem 59 also holds for the socially fair clustering problem. Hence proved. □

The following are the two corollaries that immediately follow from the above theorem.

Corollary 22 (k-median)**.** *For any $\varepsilon > 0$ and any computable function $g(k)$, the socially fair k-median problem can not be approximated to factor $(3 - \varepsilon)$ in time $g(k) \cdot n^{o(k)}$ assuming ETH, and in time $g(k) \cdot n^{O(1)}$ assuming $\mathsf{W}[2] \neq \mathsf{FPT}$.*

Corollary 23 (k-means)**.** *For any $\varepsilon > 0$ and any computable function $g(k)$, the socially fair k-means problem can not be approximated to factor $(9 - \varepsilon)$ in time $g(k) \cdot n^{o(k)}$ assuming ETH, and in time $g(k) \cdot n^{O(1)}$ assuming $\mathsf{W}[2] \neq \mathsf{FPT}$.*

5.7 Conclusion

We designed a constant factor approximation algorithm for the socially fair k-median/k-means problem in FPT time. In addition, we gave tight FPT hardness of approximation bound using the observation that the k-supplier problem is a special case of the socially fair clustering problem. This settles the complexity of the problem when parameterized by k. The natural open problem is to obtain better approximation guarantees parameterized by both k and ℓ, or ℓ alone.

Chapter 6

Hardness of Approximation: k-Median

In this chapter, we show the hardness of approximation of the k-median problem in the continuous Euclidean space. The k-median problem in the continuous Euclidean space is defined in the following manner: given a set C of n points in d-dimensional Euclidean space $\mathbb{R}^p$, and an integer k, find a set $F \subset \mathbb{R}^p$ of k points (called centers) such that the cost function $\text{cost}(F, C) \equiv \sum_{x \in C} \min_{f \in F} \|x - f\|_2$ is minimized. The Euclidean k-means problem is defined similarly by replacing the distance with squared Euclidean distance in the cost function. Various hardness of approximation results are known for the Euclidean k-means problem [19, 109, 51]. However, no hardness of approximation result was known for the Euclidean k-median problem. In this work, assuming the *unique games conjecture* (UGC), we provide the hardness of approximation result for the Euclidean k-median problem in $O(\log k)$ dimensional space. This solves an open question posed explicitly in the work of Awasthi *et al.* [19].

Furthermore, we study the hardness of approximation for the Euclidean k-means/k-median problems in the bi-criteria setting where an algorithm is allowed to choose more than k centers. That is, bi-criteria approximation algorithms are allowed to output βk centers (for constant $\beta > 1$) and the approximation ratio is computed with respect to the optimal k-means/k-median

cost. We show the hardness of bi-criteria approximation result for the Euclidean k-median problem for any $\beta < 1.015$, assuming UGC. We also show a similar hardness of bi-criteria approximation result for the Euclidean k-means problem with a stronger bound of $\beta < 1.28$, again assuming UGC.

6.1 Overview

We start by formally defining the Euclidean k-median problem.

Definition 48 (Euclidean k-Median Problem)**.** *Given a set C of n points in $\mathbb{R}^p$, and a positive integer k, find a set of centers $F \subset \mathbb{R}^p$ of size k such that the cost function* $\text{cost}(F, C) \equiv \sum_{x \in C} \min_{f \in F} \|x - f\|$ *is minimized.*

The Euclidean k-means problem is defined similarly by replacing the distance with squared Euclidean distance in the cost function (i.e., replacing $\|x - c\|$ with $\|x - c\|^2$). This version of the problem is known as the *continuous* version. In the *discrete* version, the centers are restricted to be chosen from a specific set $L \subset \mathbb{R}^p$. In the approximation setting, the continuous version is not harder than its discrete counterpart since it is known(e.g., [77, 122]) that an α-approximation for the discrete problem gives an $\alpha+\varepsilon$ approximation for the continuous version, for arbitrary small constant $\varepsilon > 0$, in time poly$(n,d)\cdot(k/\varepsilon)^{O(1/\varepsilon)}$. In this work, we study only the continuous version of the problem. In this chapter, we use k-means/median to implicitly mean *continuous Euclidean* k-means/median unless specified otherwise [1].

For the k-means problem, there exists a constant $\varepsilon > 0$ such that there does not exist an efficient $(1 + \varepsilon)$-approximation algorithm, assuming $\mathsf{P} \neq \mathsf{NP}$ [19, 109, 51]. The best-known hardness of approximation result for the k-means problem is 1.07 due to Cohen-Addad and Karthik [51]. However, unlike the Euclidean k-means problem, no hardness of approximation result was known for the Euclidean k-median problem. Resolving the hardness of approximation for the

[1] in some literature, the Euclidean space implicitly means the dimension is bounded, but in our case the dimension d can be arbitrarily large

Euclidean k-median problem was left as an open problem in the work of Awasthi *et al.* [19]. They asked whether their techniques for proving the inpproximability results for Euclidean k-means can be used to prove the hardness of approximation result for the Euclidean k-median problem. From their paper,

> "*It would also be interesting to study whether our techniques give hardness of approximation results for the Euclidean k-median problem.*"

In this work, assuming UGC, we solve this open problem by obtaining the hardness of approximation result for the Euclidean k-median problem. Following is one of the main results of this work.

Theorem 60 (Main Theorem)**.** *There exists a constant $\varepsilon > 0$ such that the Euclidean k-median problem in $O(\log k)$ dimensional space cannot be approximated to a factor better than $(1+\varepsilon)$, assuming the Unique Games Conjecture.*

Having established the hardness of approximation results for k-means and k-median, the next natural step in the discussion is to allow more flexibility to the algorithm. One possible relaxation is to allow an approximation algorithm to choose more than k centers, say, βk centers (for some constant $\beta > 1$) and produce a solution that is close to the optimal solution with respect to k centers. This is known as bi-criteria approximation and the following definition formalizes this notion.

Definition 49 ((α, β)-approximation algorithm)**.** *An algorithm $\mathcal{A}$ is called an (α, β) approximation algorithm for the Euclidean k-means/k-median problem if given any instance $\mathcal{I} = (C, k)$ with $C \subset \mathbb{R}^p$, $\mathcal{A}$ outputs a center set $F \subset \mathbb{R}^p$ of size βk that has the cost at most α times the optimal cost with k centers. That is,*

$$\sum_{x \in C} \min_{f \in F} \{D(x, f)\} \leq \alpha \cdot \min_{\substack{F' \subset \mathbb{R}^p \\ |F'| = k}} \left\{ \sum_{x \in C} \min_{f \in F'} \{D(x, f)\} \right\}$$

For the Euclidean k-means problem, $D(p,q) \equiv \|p-q\|^2$ and for the k-median problem $D(p,q) \equiv \|p-q\|$.

One expects that as β grows, there would exist efficient (α, β)-approximation algorithms with smaller values of α. This is indeed observed in the work of Makarychev *et al.* [118]. For example, their algorithm gives a $(9+\varepsilon)$ approximation for $\beta = 1$; 2.59 approximation for $\beta = 2$; 1.4 approximation for $\beta = 3$. In other words, the approximation factor of their algorithm decreases as the value of β increases. Furthermore, their algorithm gives a $(1+\varepsilon)$-approximation guarantee with $O(k \log(1/\varepsilon))$ centers. Bandyapadhyay and Varadarajan [20] gave a $(1+\varepsilon)$ approximation algorithm that outputs $(1+\varepsilon)k$ centers in constant dimension. There are various other bi-criteria approximation algorithms that use distance-based sampling techniques and achieve better approximation guarantees than their non bi-criteria counterparts [12, 3, 136]. Unfortunately in these bi-criteria algorithms, at least one of α, β is large. Ideally, we would like to obtain a PTAS with a small violation of the number of output centers. More specifically, we would like to address the following question:

> *Does the Euclidean k-means or Euclidean k-median problem admit an efficient $(1+\varepsilon, 1+\varepsilon)$-approximation algorithm?*

Note that such type of bi-criteria approximation algorithms that outputs $(1+\varepsilon)k$ centers have been extremely useful in obtaining a constant approximation for the *capacitated* k-median problem [111, 112] for which no true constant approximation is known yet [2]. Therefore, the above question is worth exploring. Note that here we are specifically aiming for a PTAS since the k-means and k-median problems already admit a constant factor approximation algorithm. In this work, we give a negative answer to the above question by showing that there exists a constant $\varepsilon > 0$ such that an efficient $(1+\varepsilon, 1+\varepsilon)$-approximation algorithm for the k-means and

[2] In the capacitated k-median/k-means problem there is an additional constraint on each center that it cannot serve more than a specified number of clients (or points).

k-median problems does not exist assuming the Unique Games Conjecture. The following two theorems state this result more formally.

Theorem 61 (k-median)**.** *For any constant $1 < \beta < 1.015$, there exists a constant $\varepsilon > 0$ such that there is no $(1+\varepsilon, \beta)$-approximation algorithm for the Euclidean k-median problem in $O(\log k)$ dimensional space assuming the Unique Games Conjecture.*

Theorem 62 (k-means)**.** *For any constant $1 < \beta < 1.28$, there exists a constant $\varepsilon > 0$ such that there is no $(1+\varepsilon, \beta)$-approximation algorithm for the Euclidean k-means problem in $O(\log k)$ dimensional space assuming the Unique Games Conjecture. Moreover, the same result holds for any $1 < \beta < 1.1$ under the assumption that $\mathsf{P} \neq \mathsf{NP}$.*

For simplicity, we present the proof of our results in $O(n)$ dimensional space. However, the results easily extend to $O(\log k)$ dimensional space using dimensionality reduction techniques of Makarychev *et al.* [119].

Important note: We would like to note that assuming $\mathsf{P} \neq \mathsf{NP}$, a similar hardness of approximation result for the Euclidean k-median problem using different techniques has been obtained independently by Cohen-Addad *et al.* [58].

In the next subsection, we discuss the known results on hardness of approximation of the k-means and k-median problems in more detail.

6.2 Related Work

The k-means problem is known to be NP-hard even for fixed k or d [13, 64, 117, 133]. Similar NP hardness result is also known for the k-median problem [123]. Also, the 1-median problem, popularly known as the *Fermat-Weber* problem [75], is a difficult problem and designing efficient algorithms for this problem is a separate line of research in itself – see for e.g. [105, 137, 38, 33, 49]. These hardness barriers motivate approximation algorithms for

these problems and a lot of progress have been made in this area. For example, there are various polynomial time approximation schemes (PTASs) known for k-means and k-median when k is fixed (or constant) [122, 107, 77, 44, 98]. Similarly, various PTASs are known for fixed d [54, 82, 50]. A number of constant factor approximation algorithms are also known for k-means and k-median when k and d are considered as part of the input. For the k-means problem, constant approximation algorithms have been given [100, 8], the best being a 6.129 approximation algorithm by Grandoni *et al.* [89]. Also, for the k-median problem there are constant-approximation algorithms [39, 17, 113, 35, 8]. The best known approximation guarantee for k-median is 2.633 due to Ahmadian *et al.* [8].

The first hardness of approximation result for the Euclidean k-means problem was given by Awasthi *et al.* [19]. They obtained their result using a reduction from Vertex Cover on triangle-free graphs of bounded degree Δ to the Euclidean k-means instances. Their reduction yields a $(1 + \frac{\varepsilon}{\Delta})$ hardness factor for the k-means problem for a constant $\varepsilon > 0$. Lee *et al.* [109] showed the hardness of approximation of Vertex Cover on triangle-free graphs of bounded degree four. Using $\Delta = 4$, they obtained a 1.0013 hardness of approximation for the Euclidean k-means problem. Subsequently, Cohen-Addad and Karthik [51] improved the hardness of approximation to 1.07 using a modified reduction from the *vertex coverage problem* instead of a reduction from the vertex cover problem. Moreover, they also gave several improved hardness results for the discrete k-means/k-median problems in general and ℓ_p metric spaces. In their more recent work, they also improved the hardness of approximation results for the continuous k-means/k-median problem in general metric spaces [55]. Unlike the Euclidean k-means problem, no hardness of approximation result was known for the Euclidean k-median problem. In this work, we give hardness of approximation result for the Euclidean k-median problem assuming the Unique Game Conjecture. We summarize these results in Table 6.1.

As mentioned earlier, Cohen-Addad *et al.* [58] have independently obtained hardness of approximation result for the Euclidean k-median problem using different set of techniques and

	Fixed k		Fixed d		General	
	Lower Bound	Upper Bound	Lower Bound	Upper Bound	Lower Bound	Upper Bound
k-Median	OPEN	$(1+\varepsilon)$ [107]	NP-hard [123]	$(1+\varepsilon)$ [54]	$(1+\varepsilon)$ **(this work)**	**2.633** [8]
k-Means	NP-hard [13]	$(1+\varepsilon)$ [107]	NP-hard [117]	$(1+\varepsilon)$ [54]	**1.07** [51]	**6.129** [89]

Table 6.1: The known approximation guarantees for the Euclidean k-Median and k-Means problems. For fixed k or d, all the mentioned $(1+\varepsilon)$-approximation algorithms have FPT running time.

under the assumption that P $\neq$ NP. They also gave bi-criteria hardness of approximation results in ℓ_∞-metric for the k-means and k-median problems. We would like to point out that in the bi-criteria setting, our result is the first hardness of approximation result for the Euclidean k-means/k-median problem to the best of our knowledge.

6.3 Summary of Our Contributions

Awasthi *et al.* [19] proved the first hardness of approximation result for the Euclidean k-means problem. Given any instance $\mathcal{I} = (C, k)$ for the Euclidean k-means problem, they showed that there exists an $\epsilon > 0$ such that obtaining $(1+\epsilon)$-approximation for Euclidean k-means is NP-hard. In this work we build on their techniques to prove the inapproximability result for the Euclidean k-median problem. First, we describe the reduction employed by Awasthi *et al.* for the Euclidean k-means problem and some related results.

Construction of k-means instance: Let (G, k) be a hard Vertex Cover instance where the graph G has bounded degree Δ. Let n and m denote respectively the number of vertices and the number of edges in the graph. A k-means instance $\mathcal{I} \coloneqq (C, k)$ with $C \subset \mathbb{R}^n$ is constructed as follows. For every vertex $i \in V$, we have an n-dimensional vector $x_i \in \{0,1\}^n$, which has a 1 at i^{th} coordinate and 0

everywhere else. For each edge $e = (i, j) \in E$, a point $x_e := x_i + x_j$ is defined in $\{0, 1\}^n$. The set $C := \{x_e \mid e \in E\}$ with m points in $\mathbb{R}^n$ and the parameter k define the k-means instance.

Awasthi *et al.* [19] proved the following theorem based on the above construction.

Theorem 63 (Theorem 4.1 [19])**.** *There is an efficient reduction from vertex cover on bounded degree triangle-free graphs to the Euclidean k-means problem that satisfies the following properties:*

1. *If vertex cover of the instance is k, then there is a k-means clustering of cost at most $(m - k)$.*
2. *If vertex cover of the instance is at least $(1 + \varepsilon)k$, then the cost of optimal k-means clustering is at least $(m - k + \delta k)$.*

Here, ε is some fixed constant > 0 and $\delta = \Omega(\varepsilon)$.

Awasthi *et al.* [19] used the following hardness result for the vertex cover problem on bounded degree triangle-free graphs.

Theorem 64 (Corollary 5.3 [19])**.** *Given any unweighted bounded degree triangle-free graph G, it is* NP*-hard to approximate Vertex Cover within any factor smaller than* 1.36.

Theorem 63 and Theorem 64 together imply that the Euclidean k-means problem is APX-hard. A formal statement for the same is given as follows (see Section 4 of [19] for the proof of this result).

Corollary 24. *There exists a constant $\varepsilon' > 0$ such that it is* NP*-hard to approximate the Euclidean k-means problem to any factor better than $(1 + \varepsilon')$.*

We would like to obtain a similar gap-preserving reduction for the Euclidean k-median problem. The first obstacle one encounters in this direction is that unlike the 1-mean problem, there does not exist a closed form expression for the 1-median problem, and hence we don't have an exact expression for the optimal 1-median cost. We overcome this barrier by obtaining good upper and lower bounds on the optimal 1-median cost and showing that these bounds suffice for our purpose. More concretely, to upper bound the optimal 1-median cost, we use the centroid as the 1-median and compute the 1-median cost with respect to the centroid. To obtain a lower bound on the 1-median cost of a cluster, we use a decomposition technique to break a cluster into smaller sub-clusters[3] for which we can compute exact or good approximate lower bounds on the 1-median cost. Here we use a simple observation that the optimal 1-median cost of a cluster is at least the sum of the optimal 1-median costs of the sub-clusters. For any sub-cluster that corresponds to a star graph, one can compute the exact 1-median cost using our reduction. In order to bound the 1-median cost for sub-clusters that correspond to non-star graphs, we use the following observation crucially: the optimal 1-median cost is preserved under any transformation that preserves the pairwise distances. For non-star graphs, we first employ such a transformation that preserves the 1-median cost and then compute this cost exactly in the projected space. Note that this technique does not give exact 1-median cost for any arbitrary non-star graph, but works only for some *special families* of non-star graphs. The main idea of the decomposition technique is to ensure that only these kinds of non-star graphs are created in the decomposition process. The upper and lower bounds on the 1-median cost, as constructed in the above manner, are used in the completeness and soundness steps of the proof of the reduction, respectively.

The analysis for the completeness part of the reduction is relatively straightforward. If the vertex cover of a graph is k, then the edges of the graph can be divided into k star sub-graphs, each of which results in a star cluster in the k-median instance. The cost for this clustering with

[3]Since a set of edges in a graph form a cluster of points in the reduction, we use the terms sub-graphs and sub-clusters interchangeably.

k star clusters can be found using the reduction easily.

In the proof for the soundness part of our reduction, we prove the contrapositive statement that assumes the k-median clustering cost to be bounded and proves that the vertex cover of the graph is not too large. Our analysis crucially depends on the relation between the vertex cover of a subgraph and the 1-median cost for that subgraph. More specifically, we need to answer the following question. Given a graph with r edges having vertex cover z, how does the optimal 1-median cost for that graph behave with respect to z. For example, for star graphs, $z = 1$ and the optimal 1-median cost of a star graph on r edges is exactly $\sqrt{r(r-1)}$. For any non-star graph with r edges, we first show that the optimal 1-median cost of the non-star graph is at least the optimal 1-median cost of a star graph with r edges. For any non-star graph F with r edges, we denote by $\delta(F)$ the *extra cost* of F, defined as the difference of the optimal 1-median cost of F and the optimal 1-median cost of a star graph with r edges. If we can figure out non-trivial lower bounds for $\delta(F)$ for different non-star graphs F, then we would be done. But, figuring out these non-trivial lower bounds that work for any non-star graph is quite a daunting prospect. The way we overcome this in our work is as follows. We characterize the non-star graphs as having maximum matching of size two or more than two, and for each, we relate the *extra cost* of 1-median clustering of that graph with the vertex cover of that graph. We show that the extra cost of a non-star sub-graph is proportional to the number of vertex-disjoint edges in the sub-graph. And since we assume the k-median cost to be bounded, the number of vertex disjoint edges is also bounded, giving a small vertex cover.

We need one more idea to finish the proof for the soundness part of the reduction. We call a cluster 'singleton' if there is only one point in the cluster. Note that any such cluster would cost zero in a k-median clustering. If there are a large number of singleton clusters, say $t < k$, then they pay zero to the cost of the solution, even though those edges have vertex cover t. We prove a key lemma showing that for any *hard* instance of the vertex cover, the vertex cover of the sub-graph spanned by t singleton edges is at most $\frac{2t}{3}$. We combine these ideas to prove that

if k-median clustering cost is bounded, the vertex cover of the graph cannot be too large.

We also prove the hardness of bi-criteria approximation results for Euclidean k-means and k-median problems. The hardness of bi-criteria approximation for Euclidean k-median is obtained by extending the proof for the hardness of approximation for the Euclidean k-median problem. We use the same reduction from the vertex cover problem and show that the soundness guarantees hold even if one is allowed to use βk centers, for some $\beta > 1$. We also show that similar techniques give the hardness of bi-criteria approximation results for the Euclidean k-means problem.

6.4 Notations and Useful Inequalities

In this section, we discuss some basic notations and inequalities that we use frequently in our proofs. Recall that a point in C corresponds to an edge of the graph. Therefore, a sub-graph S of G corresponds to a subset of the point set C. We denote this subset as $C(S)$ such that $C(S) \coloneqq \{x_e \mid e \in E(S)\} \subseteq C$. We define the 1-median cost of $C(S)$ with respect to a center $f \in \mathbb{R}^n$ as cost$(f, S) \equiv \sum_{x \in C(S)} \|x - f\|$. Furthermore, we define the **optimal** 1-median cost of the point set $C(S)$ as cost$^*(S)$. That is, cost$^*(S) \equiv \min_{f \in \mathbb{R}^n}$ cost(f, S). We often use these statements interchangeably, "optimal 1-median cost of a graph S" to mean "optimal 1-median cost of the cluster $C(S)$".

First, we note that the Fermat-Weber problem is not difficult for all 1-median instances. We can efficiently obtain 1-median for some special instances. For example, for a set of equidistant points, the 1-median is simply the centroid of the point set. We give a proof of this statement in the next section. Most importantly, we use the following fact and lemma to compute the 1-median cost.

Fact 5 ([124])**.** *For a set of non-collinear points the optimal* 1*-median is unique.*

Next, we give a simple lemma, that is used to prove various bounds related to the quantity

$\sqrt{m(m-1)}$.

Lemma 44. *Let m and t be any positive real numbers greater than one. If $m \geq t$, the following bound holds:*

$$m - (t - \sqrt{t(t-1)}) \leq \sqrt{m(m-1)} \leq m - 1/2.$$

Proof. The upper bound follows from the sequence of inequalities: $\sqrt{m(m-1)} < \sqrt{m^2 - m + 1/4} = \sqrt{(m-1/2)^2} = m - 1/2$. The lower bound follows from the following sequence of inequalities:

$$\sqrt{m(m-1)} = m + m \cdot \left(\sqrt{\frac{m-1}{m}} - 1 \right) \geq m + t \cdot \left(\sqrt{\frac{t-1}{t}} - 1 \right) = m - (t - \sqrt{t(t-1)}).$$

The second inequality holds because $\frac{a+1}{b+1} \geq \frac{a}{b}$ for $b \geq a$. This completes the proof of the lemma. □

In some vector spaces, it is tricky to compute the optimal 1-median exactly. In such cases, we transform the space to a different vector space, where computing the 1-median is relatively simpler. More specifically, we employ a rigid transformation since it preserves pairwise distances. Moreover, a rigid transformation preserves the 1-median cost of the instance. Formally, we state the result as follows:

Lemma 45. *Let $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ be any two sets of n points in $\mathbb{R}^p$. If the pairwise distances between points within A is the same as pairwise distance between points within B. That is, for all $i, j \in \{1, \ldots, n\}$, $\|a_i - a_j\| = \|b_i - b_j\|$. Then the optimal 1-median cost of A is the same as the optimal 1-median cost of B.*

Let $co(A)$ and $co(B)$ denote the convex hulls of A and B, respectively. We split the proof of the Lemma 45 in two parts. In the first part (Lemma 46), we show that there exists a distance preserving transformation $\mathcal{R}$ from $co(A)$ to $co(B)$ such that $\mathcal{R}(a_i) = b_i$ for every $i \in \{1, \ldots, n\}$.

By distance preserving transformation, we mean that for any two points $x, y \in co(A)$, the distance $\|x-y\|$ is preserved after applying the transformation $\mathcal{R}$, i.e., $\|x-y\| = \|\mathcal{R}(x)-\mathcal{R}(y)\|$. In the second part (Lemma 47), we show that applying the transformation $\mathcal{R}$ preserves the optimal 1-median cost of A.

Lemma 46. *Given two sets of points $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$ in $\mathbb{R}^p$ such that $\|a_i - a_j\| = \|b_i - b_j\|$ for all $i, j \in \{1, \ldots, n\}$. Then there exists a distance preserving transformation $\mathcal{R}\colon co(A) \to co(B)$ such that $\mathcal{R}(a_i) = b_i$ for every $i \in \{1, \ldots, n\}$.*

Proof. Let $\boldsymbol{X_i}$ be a vector[4] defined as $\boldsymbol{a_i} - \boldsymbol{a_1}$ for every $\boldsymbol{a_i} \in A$. Similarly, we define a vector $\boldsymbol{Y_i} \coloneqq \boldsymbol{b_i} - \boldsymbol{b_1}$ for every $\boldsymbol{b_i} \in B$. We will use these vectors to define the transformation $\mathcal{R}$. For now, note the following property of inner product of $\boldsymbol{X_i}$ and $\boldsymbol{X_j}$.

$$\langle \boldsymbol{X_i}, \boldsymbol{X_j} \rangle = \langle \boldsymbol{Y_i}, \boldsymbol{Y_j} \rangle \quad \text{for every } i, j \in \{1, \ldots, n\} \tag{6.1}$$

The proof of the above property follows from the following sequence of inequalities:

$$\begin{aligned}
2 \cdot \langle \boldsymbol{X_i}, \boldsymbol{X_j} \rangle &= \|\boldsymbol{X_i}\|^2 + \|\boldsymbol{X_j}\|^2 - \|\boldsymbol{X_i} - \boldsymbol{X_j}\|^2 \\
&= \|\boldsymbol{Y_i}\|^2 + \|\boldsymbol{Y_j}\|^2 - \|\boldsymbol{X_i} - \boldsymbol{X_j}\|^2, \\
&\qquad\qquad (\because \|\boldsymbol{X_i}\| = \|\boldsymbol{a_i} - \boldsymbol{a_1}\| = \|\boldsymbol{b_i} - \boldsymbol{b_1}\| = \|\boldsymbol{Y_i}\|^2) \\
&\qquad\qquad \text{for every } 1 \leq i \leq n \\
&= \|\boldsymbol{Y_i}\|^2 + \|\boldsymbol{Y_j}\|^2 - \|\boldsymbol{Y_i} - \boldsymbol{Y_j}\|^2, \\
&\qquad\qquad (\because \|\boldsymbol{X_i} - \boldsymbol{X_j}\| = \|\boldsymbol{a_i} - \boldsymbol{a_j}\| = \|\boldsymbol{b_i} - \boldsymbol{b_j}\| = \|\boldsymbol{Y_i} - \boldsymbol{Y_j}\|^2) \\
&= 2 \cdot \langle \boldsymbol{Y_i}, \boldsymbol{Y_j} \rangle
\end{aligned}$$

In other words, the triangles (a_1, a_i, a_j) and (b_1, b_i, b_j) are congruent for all $i, j \in \{1, \ldots, n\}$. Therefore, the inner product $\langle \boldsymbol{X_i}, \boldsymbol{X_j} \rangle$ is the same as $\langle \boldsymbol{Y_i}, \boldsymbol{Y_j} \rangle$.

[4]For better readability, we boldfaced the vector symbols to distinguish them from any scalar quantity.

Now, we describe the transformation $\mathcal{R}$ from $co(A)$ to $co(B)$. By the definition of $co(A)$, any point $\boldsymbol{x} \in co(A)$ can be expressed in the form $\sum_{i=1}^{n}\lambda_i \cdot \boldsymbol{a_i}$ for some $0 \leq \lambda_i's \leq 1$ and $\sum_{i=1}^{n}\lambda_i = 1$. Equivalently, $\boldsymbol{x}$ can be expressed as $\boldsymbol{a_1} + \sum_{i=2}^{n}\lambda_i \cdot \boldsymbol{X_i}$. For $\boldsymbol{x} \in co(A)$, we define the transformation $\mathcal{R}$ as $\mathcal{R}(\boldsymbol{x}) \coloneqq \lambda_i \boldsymbol{b_i}$. Again, $\mathcal{R}(\boldsymbol{x})$ can be equivalently expressed as $\boldsymbol{b_1} + \sum_{i=2}^{n}\lambda_i \cdot \boldsymbol{Y_i}$. It is easy to see that $\lambda_i \cdot \boldsymbol{b_i}$ indeed belongs to $co(B)$ since $0 \leq \lambda_i \leq 1$ and $\sum_{i=1}^{n}\lambda_i = 1$. Now, we show that $\mathcal{R}$ is a distance preserving transformation. Let $\boldsymbol{x} \coloneqq \boldsymbol{a_1} + \sum_{i=2}^{n}\lambda_i \cdot \boldsymbol{X_i}$ and $\boldsymbol{y} \coloneqq \boldsymbol{a_1} + \sum_{i=2}^{n}\gamma_i \cdot \boldsymbol{X_i}$ be any two points in $co(A)$. The following sequence of inequalities prove that $\|\boldsymbol{x} - \boldsymbol{y}\| = \|\mathcal{R}(\boldsymbol{x}) - \mathcal{R}(\boldsymbol{y})\|$.

$$
\begin{aligned}
\|\boldsymbol{x} - \boldsymbol{y}\|^2 &= (\boldsymbol{x} - \boldsymbol{y})^T(\boldsymbol{x} - \boldsymbol{y}) \\
&= \left(\sum_{i=2}^{n}(\lambda_i - \gamma_i) \cdot \boldsymbol{X_i}\right)^T \cdot \left(\sum_{i=2}^{n}(\lambda_i - \gamma_i) \cdot \boldsymbol{X_i}\right) \\
&= \sum_{i=2}^{n}\sum_{j=2}^{n}(\lambda_i - \gamma_i) \cdot (\lambda_j - \gamma_j) \cdot \langle \boldsymbol{X_i}, \boldsymbol{X_j} \rangle \\
&= \sum_{i=2}^{n}\sum_{j=2}^{n}(\lambda_i - \gamma_i) \cdot (\lambda_j - \gamma_j) \cdot \langle \boldsymbol{Y_i}, \boldsymbol{Y_j} \rangle, \qquad \text{using Equation 6.1} \\
&= \left(\sum_{i=2}^{n}(\lambda_i - \gamma_i) \cdot \boldsymbol{Y_i}\right)^T \cdot \left(\sum_{i=2}^{n}(\lambda_i - \gamma_i) \cdot \boldsymbol{Y_i}\right) \\
&= (\mathcal{R}(\boldsymbol{x}) - \mathcal{R}(\boldsymbol{y}))^T(\mathcal{R}(\boldsymbol{x}) - \mathcal{R}(\boldsymbol{y})) \\
&= \|\mathcal{R}(\boldsymbol{x}) - \mathcal{R}(\boldsymbol{y})\|^2
\end{aligned}
$$

This proves that $\mathcal{R}$ is a distance preserving transformation from $co(A)$ to $co(B)$. Moreover, note that $\mathcal{R}$ is a bijective function. It is possible that a vector $\boldsymbol{x} \in co(A)$ has multiple forms, say $\sum_{i=1}^{n}\lambda_i \cdot \boldsymbol{a_i}$ and $\sum_{i=1}^{n}\Lambda_i \cdot \boldsymbol{a_i}$. Therefore, it appears that $\boldsymbol{x}$ maps to different vectors in $co(B)$. However, it always maps to the same vector. For the sake of contradiction, assume that $\boldsymbol{x}$ maps

to two different vectors $\boldsymbol{p} \coloneqq \sum_{i=1}^{n} \lambda_i \cdot \boldsymbol{b_i}$ and $\boldsymbol{q} \coloneqq \sum_{i=1}^{n} \Lambda_i \cdot \boldsymbol{b_i}$ in $co(B)$. Then $\|\boldsymbol{p} - \boldsymbol{q}\| \neq 0$. It contradicts the fact that $\mathcal{R}$ is a distance preserving transformation. Similarly, we can show that any two different vectors $\boldsymbol{x}, \boldsymbol{y} \in co(A)$ can not map to the same vector in $co(B)$. This proves that $\mathcal{R}$ is a bijective function.

Furthermore, note that $\mathcal{R}(\boldsymbol{a_i}) = \boldsymbol{b_i}$ for every $i \in \{1, \ldots, n\}$. To see this, consider $\lambda_i = 1$ and $\lambda_j = 0$ for all $j \in \{1, \ldots, n\} \backslash \{i\}$. Then $\boldsymbol{a_i} = \sum_{j=1}^{n} \lambda_j \cdot \boldsymbol{a}_j$ and therefore $\mathcal{R}(\boldsymbol{a_i}) = \sum_{j=1}^{n} \lambda_j \cdot \boldsymbol{b}_j = \boldsymbol{b_j}$. This completes the proof of the lemma. □

Similar to $\mathcal{R}$, we can also define a distance preserving transformation $\mathcal{R}^{-1}$ from $co(B)$ to $co(A)$. The transformation $\mathcal{R}^{-1}$ is defined such that for any $\boldsymbol{x} = \sum_{i=1}^{n} \lambda_i \cdot \boldsymbol{b_i} \in co(B)$, $\mathcal{R}^{-1}(\boldsymbol{x}) = \sum_{i=1}^{n} \lambda_i \cdot \boldsymbol{a_i} \in co(A)$. Furthermore, as per this definition of $\mathcal{R}^{-1}$, $\mathcal{R}^{-1}(\boldsymbol{b_i}) = \boldsymbol{a_i}$ for every $i \in \{1, \ldots, n\}$. Now, we show that applying the transformation $\mathcal{R}$ on A preserves the optimal 1-median cost of A.

Lemma 47. *If there exists distance preserving transformations $\mathcal{R}\colon co(A) \to co(B)$ and $\mathcal{R}^{-1}\colon co(B) \to co(A)$ such that $\mathcal{R}(\boldsymbol{a_i}) = \boldsymbol{b_i}$ and $\mathcal{R}^{-1}(\boldsymbol{b_i}) = \boldsymbol{a_i}$ for every $i \in \{1, \ldots, n\}$. Then the optimal 1-median cost of A is the same as the optimal 1-median cost of B.*

Proof. Recall that 1-median cost of an instance A with respect to a center $\boldsymbol{c} \in \mathbb{R}^p$ is denoted by $\text{cost}(\boldsymbol{c}, A) \equiv \sum_{\boldsymbol{a_i} \in A} \|\boldsymbol{a_i} - \boldsymbol{c}\|$. Let $\boldsymbol{c_1^*}$ be the optimal 1-median of A. Furthermore, we can assume that $\boldsymbol{c_1^*} \in co(A)$ since the optimal 1-median lies in the convex hull of A (see *e.g.* Remark 2.1 in [125]). Similarly, let $\boldsymbol{c_2^*} \in co(B)$ be the optimal 1-median of B. Now, we show that $\text{cost}(\boldsymbol{c_1^*}, A) \geq \text{cost}(\boldsymbol{c_2^*}, B)$ and $\text{cost}(\boldsymbol{c_2^*}, B) \geq \text{cost}(\boldsymbol{c_1^*}, A)$ using the following sequence of inequalities:

$$\text{cost}(\boldsymbol{c_1^*}, A) = \sum_{\boldsymbol{a_i} \in A} \|\boldsymbol{a_i} - \boldsymbol{c_1^*}\|$$

$$
\begin{aligned}
&= \sum_{\boldsymbol{a_i} \in A} \|\mathcal{R}(\boldsymbol{a_i}) - \mathcal{R}(\boldsymbol{c_1^*})\|, && \because \mathcal{R} \text{ preserves the pairwise distances} \\
&= \sum_{\boldsymbol{b_i} \in B} \|\boldsymbol{b_i} - \mathcal{R}(\boldsymbol{c_1^*})\|, && \because \mathcal{R}(a_i) = b_i \\
&\geq \sum_{\boldsymbol{b_i} \in B} \|\boldsymbol{b_i} - \boldsymbol{c_2^*}\|, && \because \boldsymbol{c_2^*} \text{ is the optimal 1-median of } B \\
&= \text{cost}(\boldsymbol{c_2^*}, B)
\end{aligned}
$$

Similarly, we show that $\text{cost}(\boldsymbol{c_2^*}, B) \geq \text{cost}(\boldsymbol{c_1^*}, A)$ as follows:

$$
\begin{aligned}
\text{cost}(\boldsymbol{c_2^*}, B) &= \sum_{\boldsymbol{b_i} \in B} \|\boldsymbol{b_i} - \boldsymbol{c_2^*}\| \\
&= \sum_{\boldsymbol{b_i} \in B} \|\mathcal{R}^{-1}(\boldsymbol{b_i}) - \mathcal{R}^{-1}(\boldsymbol{c_2^*})\|, && \because \mathcal{R}^{-1} \text{ preserves the pairwise distances} \\
&= \sum_{\boldsymbol{a_i} \in A} \|\boldsymbol{a_i} - \mathcal{R}^{-1}(\boldsymbol{c_2^*})\|, && \because \mathcal{R}^{-1}(b_i) = a_i \\
&\geq \sum_{\boldsymbol{a_i} \in A} \|\boldsymbol{a_i} - \boldsymbol{c_1^*}\|, && \because \boldsymbol{c_1^*} \text{ is the optimal 1-median of } A \\
&= \text{cost}(\boldsymbol{c_1^*}, A)
\end{aligned}
$$

This proves that $\text{cost}(\boldsymbol{c_1^*}, A) = \text{cost}(\boldsymbol{c_2^*}, B)$. Hence it proves the lemma. □

Therefore, Lemmas 46 and 47 together proves Lemma 45.

6.5 Inapproximability of Euclidean k-Median

In this section, we show the inapproximability result of the Euclidean k-median problem. We obtain this result by showing a gap preserving reduction from Vertex Cover on bounded degree triangle-free graphs to the Euclidean k-median. For Vertex Cover on bounded degree triangle-free graphs, the inapproximability result is stated in Corollary 25. The corollary simply follows from the following two results of Austrin *et al.* [18] and Awasthi *et al.* [19].

Theorem 65 (Austrin *et al.* [18])**.** *Given any unweighted bounded degree graph $G = (V, E)$ of maximum degree Δ, Vertex Cover can not be approximated within any factor smaller than $2 - \varepsilon$, for $\varepsilon = (2 + o_\Delta(1)) \cdot \frac{\log\log\Delta}{\log\Delta}$ assuming the Unique Games Conjecture.*

In the above theorem, ε can be set to arbitrarily small value by taking sufficiently large value of Δ.

Theorem 66 (Awasthi *et al.* [19])**.** *There is a $(1+\varepsilon)$-approximation-preserving reduction from Vertex Cover on bounded degree graphs to Vertex Cover on triangle-free graphs of bounded degree.*

Corollary 25. *Given any unweighted triangle-free graph G of bounded degree, Vertex Cover can not be approximated within a factor smaller than $2 - \varepsilon$, for any constant $\varepsilon > 0$, assuming the Unique Games Conjecture.*

In Section 6.3, we described the reduction used by Awasthi *et al.* [19] to construct instances for Euclidean k-means from a Vertex Cover instance. We use the same construction for the Euclidean k-median instances. Let $G = (V, E)$ denote a triangle-free graph of bounded degree Δ. Let $\mathcal{I} = (C, k)$ denote the Euclidean k-median instance constructed from G. We establish the following theorem based on this construction.

Theorem 67. *There is an efficient reduction from Vertex Cover on bounded degree triangle-free graphs with m edges to the Euclidean k-median problem that satisfies the following properties:*

1. *If the graph has a vertex cover of size k, then the k-median instance has a solution of cost at most $m - k/2$.*
2. *If the graph has no vertex cover of size at most $(2 - \varepsilon) \cdot k$, then the cost of any k-median solution on the instance is at least $m - k/2 + \delta k$.*

Here, ε is some fixed constant, $\delta = \Omega(\varepsilon)$, and $k \geq$ the size of maximum matching of the graph.

The graphs with a vertex cover of size at most k are said to be "Yes" instances and the graphs with no vertex cover of size at most $(2-\varepsilon)k$ are said to be "No" instances. Now, the above theorem gives the following inapproximability result for the Euclidean k-median problem.

Corollary 26. *There exists a constant $\varepsilon' > 0$ such that the Euclidean k-median problem can not be approximated to a factor better than $(1+\varepsilon')$, assuming the Unique Games Conjecture.*

Proof. Since the hard Vertex Cover instances have bounded degree Δ, the maximum matching of such graphs is at least $\lceil \frac{m}{2\Delta} \rceil$. First, let us prove this statement. Suppose M be a matching, that is initially empty, i.e., $M = \emptyset$. We construct M in an iterative manner. First, we pick an arbitrary edge from the graph and add it to M. Then, we remove this edge and all the edges incident on it. We repeat this process for the remaining graph until the graph becomes empty. In each iteration, we remove at most 2Δ edges. Therefore, the matching size of the graph is at least $\lceil \frac{m}{2\Delta} \rceil$.

Now, suppose $k < \frac{m}{2\Delta}$. Then, the graph does not have a vertex cover of size k since matching size is at least $\lceil \frac{m}{2\Delta} \rceil$. Therefore, such graph instances can be classified as "No" instances in polynomial time. So, they are not the hard Vertex Cover instances. Therefore, we can assume $k \geq \frac{m}{2\Delta}$ for all the hard Vertex Cover instances. In that case, the second property of Theorem 67, implies that the cost of k-median instance is $(m - \frac{k}{2}) + \delta k \geq (1 + \frac{\delta}{2\Delta}) \cdot (m - \frac{k}{2})$. Thus, the k-median problem can not be approximated within any factor smaller than $1 + \frac{\delta}{2\Delta} = 1 + \Omega(\varepsilon)$. □

6.5.1 Completeness

Let $W = \{v_1, \ldots, v_k\}$ be a vertex cover of G. Let S_i denote the set of edges covered by v_i. If an edge is covered by two vertices v_i and v_j, then we arbitrarily keep the edge either in S_i or S_j. Let m_i denote the number of edges in S_i. We define $\{C(S_1), \ldots, C(S_k)\}$ as a clustering of the point set C. Now, we show that the cost of this clustering is at most $m - k/2$. Note that each S_i forms a star graph centered at v_i. Moreover, the point set $C(S_i)$ forms a regular simplex of

side length $\sqrt{2}$. We compute the optimal cost of $C(S_i)$ using the following lemma.

Lemma 48. *For a regular simplex on r vertices and side length s, the optimal 1-median is the centroid of the simplex. Moreover, the optimal 1-median cost is $s \cdot \sqrt{\frac{r(r-1)}{2}}$.*

Proof. The statement is easy to see for $r = 1$. For $r = 2$, there are two points s distance apart. The optimal 1-median cost is s. So, for the rest of the proof, we assume that $r > 2$. Suppose $A = \{a_1, a_2, \ldots, a_r\}$ denote the vertex set of a regular simplex. Let s be the side length of the simplex. Using Lemma 45, we can represent each point a_i in an r-dimensional space as follows; we use the same notation to denote the points after such transformations.

$$a_1 \coloneqq \left(\frac{s}{\sqrt{2}}, 0, ..., 0\right), \quad a_2 \coloneqq \left(0, \frac{s}{\sqrt{2}}, ..., 0\right), \quad \ldots, \quad a_r \coloneqq \left(0, 0, ..., \frac{s}{\sqrt{2}}\right)$$

Note that the distance between any a_i and a_j is s, which is the side length of the simplex. Let $c^* = (c_1, \ldots, c_r)$ be an optimal 1-median of point set A. Then, the 1-median cost is the following:

$$\text{cost}(c^*, A) = \sum_{i=1}^{r} \|a_i - c^*\| = \sum_{i=1}^{r} \left(\sum_{j=1}^{r} c_j^2 - c_i^2 + \left(\frac{s}{\sqrt{2}} - c_i\right)^2 \right)^{1/2}$$

Suppose $c_i \neq c_j$ for any $i \neq j$. Then, we can swap c_i and c_j to create a different median, while keeping the 1-median cost the same. It contradicts the fact that there is only one optimal 1-median, by Fact 5. Therefore, we can assume $c^* = (c, c, \ldots, c)$. Now, the optimal 1-median cost is:

$$\text{cost}^*(A) = \text{cost}(c^*, A) \coloneqq r \cdot \sqrt{\left(c - \frac{s}{\sqrt{2}}\right)^2 + (r-1) \cdot c^2}$$

The function $\text{cost}(c^*, A)$ is strictly convex and attains minimum at $c = \frac{s}{m \cdot \sqrt{2}}$, which is the centroid of A. The optimal 1-median clustering cost is $\text{cost}(c^*, A) = s \cdot \sqrt{\frac{r(r-1)}{2}}$. This completes the proof of the lemma. □

The following corollary establishes the cost of a star graph S_i.

Corollary 27. *Any star graph S_i with r edges has the optimal 1-median cost of $\sqrt{r(r-1)}$*

Furthermore, note that a set of r pairwise vertex-disjoint edges forms a regular simplex in C, of side length 2. The following corollary establishes the cost of such clusters.

Corollary 28. *Let F be any non-star graph with r pairwise vertex-disjoint edges, then the optimal 1-median cost of F is $\sqrt{2} \cdot \sqrt{r(r-1)}$*

We use Corollary 28 in Section 6.6. For now, we only use Corollary 27 to bound the optimal k-median cost of C. Let $\text{OPT}(C, k)$ denote the optimal k-median cost of C. The following sequence of inequalities proves the first property of Theorem 67.

$$\text{OPT}(C,k) \leq \sum_{i=1}^{k} \text{cost}^*(S_i) \overset{(Corollary\,27)}{=} \sum_{i=1}^{k} \sqrt{m_i(m_i-1)} \overset{(Lemma\,44)}{\leq} \sum_{i=1}^{k} \left(m_i - \frac{1}{2}\right) = m - \frac{k}{2}.$$

6.5.2 Soundness

Now, we prove the second property of Theorem 67. For this, we prove the equivalent contrapositive statement: If the optimal k-median clustering of C has cost at most $\left(m - \frac{k}{2} + \delta k\right)$, for some constant $\delta > 0$, then G has a vertex cover of size at most $(2-\varepsilon)k$, for some constant $\varepsilon > 0$. Let $\mathcal{C}$ denote an optimal k-median clustering of C. We classify its optimal clusters into two categories: (1) *star* and (2) *non-star*. Let $F_1, F_2, \ldots, F_t$ denote the non-star clusters, and $S_1, \ldots, S_{k-t}$ denote the star clusters. For any star cluster, the vertex cover size is exactly one. Moreover, using Corollary 27, the optimal 1-median cost of any star cluster with r edges is $\sqrt{r(r-1)}$. On the other hand, it may be tricky to exactly compute the vertex cover or the optimal cost of any non-star cluster. Suppose the optimal 1-median cost of a non-star cluster F on r edges is given as $\sqrt{r(r-1)} + \delta(F)$, where $\delta(F)$ denotes the *extra-cost* due to a non-star

cluster F. Using this, we define $\delta(F)$ as the following:

$$\delta(F) \equiv \text{cost}^*(F) - \sqrt{|F|(|F|-1)}$$

The following lemmas bound the vertex cover of F in terms of $\delta(F)$.

Lemma 49. *Any non-star cluster F with a maximum matching of size two has a vertex cover of size at most* $1.62 + \left(\sqrt{2}+1\right)\delta(F)$.

Lemma 50. *Any non-star cluster F with a maximum matching of size at least three has a vertex cover of size at most* $1.8 + \left(\sqrt{2}+1\right)\delta(F)$.

These lemmas are the key to proving the main result. We discussed the main proof ideas of these lemmas earlier in Section 6.3; however we give the complete proof in Section 6.6. Now, let us see how these lemmas give a vertex cover of size at most $(2-\varepsilon)k$. Let us classify the star clusters into the following two sub-categories:

(a) Clusters composed of exactly one edge. Let these clusters be: $P_1, P_2, \ldots, P_{t_1}$.

(b) Clusters composed of at least two edges. Let these clusters be: $S_1, S_2, \ldots, S_{t_2}$.

Similarly, we classify the non-star clusters into the following two sub-categories:

(i) Clusters with a maximum matching of size two. Let these clusters be: $W_1, W_2, \ldots, W_{t_3}$

(ii) Clusters with a maximum matching of size at least three. Let these clusters be: $Y_1, Y_2, \ldots, Y_{t_4}$

Note that $t_1 + t_2 + t_3 + t_4$ equals k. Now, consider the following strategy of computing the vertex cover of G. Suppose, we compute the vertex cover for every cluster separately. Let C_i be any cluster, and $|VC(C_i)|$ denote the vertex cover size of C_i. Then, the vertex cover of G

can be simply bounded in the following manner:

$$|VC(G)| \leq \sum_{i=1}^{t_1} |VC(P_i)| + \sum_{i=1}^{t_2} |VC(S_i)| + \sum_{i=1}^{t_3} |VC(W_i)| + \sum_{i=1}^{t_4} |VC(Y_i)|$$

However, we can obtain a vertex cover of smaller size using a slightly different strategy. In this strategy, we first compute a minimum vertex cover of all the clusters except single edge clusters $P_1, P_2, \ldots, P_{t_1}$. Suppose that vertex cover is VC'. Then we compute a vertex cover for $P_1, P_2, \ldots, P_{t_1}$. Now, let us see why this strategy gives a vertex cover of smaller size than before. Note that some vertices in VC' may also cover the edges in $P_1, \ldots, P_{t_1}$. Suppose there are t'_1 clusters in $P_1, \ldots, P_{t_1}$ that remain uncovered by VC'. Without loss of generality, assume these clusters to be $P_1, \ldots, P_{t'_1}$. Now, the vertex cover of G is bounded in the following manner:

$$\begin{aligned}
|VC(G)| &\leq |VC\left(\cup_{i=1}^{t'_1} P_i\right)| + |VC'| \\
&= |VC\left(\cup_{i=1}^{t'_1} P_i\right)| + |VC\left((\cup_{j=1}^{t_2} S_j) \cup (\cup_{k=1}^{t_3} W_k) \cup (\cup_{l=1}^{t_4} Y_l)\right)| \\
&\leq |VC\left(\cup_{i=1}^{t'_1} P_i\right)| + \sum_{i=1}^{t_2} |VC(S_i)| + \sum_{i=1}^{t_3} |VC(W_i)| + \sum_{i=1}^{t_4} |VC(Y_i)|
\end{aligned}$$

Now, we will try to bound the size of the vertex cover of $P_1 \cup ... \cup P_{t'_1}$. Note that we can cover all these single-edge clusters with t'_1 vertices by choosing one vertex per cluster. However, it may be possible to obtain a vertex cover of smaller size if we collectively consider all these clusters. Suppose E_P denote the set of all edges in $P_1, \ldots, P_{t'_1}$ and V_P denote the vertex set spanned by them. We define a graph $G_P = (V_P, E_P)$. Furthermore, suppose that M_P is a maximal matching of G_P. Then, it is easy to see that if $|M_P| \leq t'_1/3 + 4\delta k$ for some $\delta > 0$, we can simply pick both end-points of every edge in M_P, and it would give a vertex cover of G_P of size at most $2t'_1/3 + 8\delta k$. On the other hand, if $|M_P| > t'_1/3 + 4\delta k$, we show that the graph G admits a vertex cover of size at most $(2k - 2\delta k)$. We prove this statement in the following lemma:

Lemma 51. *Let $\delta > 0$ be any constant and $G_P = (V_P, E_P)$ be the graph spanned by single*

edge clusters $P_1, ..., P_{t'_1}$. If G_P does not have a vertex cover of size $\leq (\frac{2t'_1}{3} + 8\delta k)$, then G has a vertex cover of size at most $(2k - 2\delta k)$.

Proof. Let us define another subgraph $\overline{G}_P$ of G that is obtained after removing the edges of E_P from G. That is, $\overline{G}_P = (V, E \setminus E_P)$. In other words, $\overline{G}_P$ is the graph spanned by the remaining clusters: $S_1, \ldots, S_{t_2}$; $W_1, \ldots, W_{t_3}$; $Y_1, \ldots, Y_{t_4}$ and $P_{t'_1+1}, \ldots, P_{t_1}$. An important property of G_P is that any edge of $\overline{G}_P$ does not have its both endpoints in V_P. In other words, every edge of $\overline{G}_P$ is incident on at most one edge of G_P. This is because every edge of $\overline{G}_P$ has at least one endpoint in VC', and G_P is only defined on the edges that are not incident on VC'. This property will help us in obtaining a better vertex cover for G.

Let M_P be a maximal matching of G_P. It is easy to see that $|M_P| > (t'_1/3 + 4\delta k)$; otherwise we can cover all edges of G_P by picking both endpoints of every edge in M_P. It would contradict that G_P does not have a vertex cover of size less than $(2t'_1/3 + 8\delta k)$. Therefore, $|M_P| > (t'_1/3 + 4\delta k)$; using it we show that G has a vertex cover of size at most $(2k - 2\delta k)$.

We will incrementally construct a vertex cover VC_G of the graph G of size at most $(2k - 2\delta k)$. First, let us discuss the main idea of this incremental construction. During the construction, we maintain a maximal matching M_G of the graph G. Then, for every edge in M_G, we add both its endpoints to VC_G, except at least $2\delta k$ edges for which we choose only one endpoint in VC_G. Note that for the hard Vertex Cover instances, we can assume that maximum matching size is at most k. This is because the graphs with a matching of size $> k$ have a minimum vertex cover of size $> k$. Therefore, such instances can be simply classified as "No" instances in polynomial time. Since the size of a maximal matching is always less than the size of a maximum matching, we can further assume $|M_G| \leq k$ for the hard Vertex Cover instances. This implies that VC_G has size at most $2(k - 2\delta k) + 2\delta k = (2k - 2\delta k)$. Now, let us discuss the construction of such a matching M_G and correspondingly the vertex cover VC_G that covers all edges of G.

Initially, both M_G and VC_G are empty sets. That is, $M_G = \emptyset$ and $VC_G = \emptyset$. Recall that G_P is

the graph spanned by E_P and $\overline{G}_P$ is the graph spanned by $E \setminus E_P$. Let E_I denote the set of edges in $\overline{G}_P$ that are incident on M_P. Based on this, we define two new graphs: (1) $G_R \coloneqq (V, E_R)$ where $E_R = E_P \cup E_I$, and (2) $G' \coloneqq (V, E')$ where $E' = E \setminus (E_P \cup E_I)$. In other words, G_R is the graph spanned by the edges of G_P and the edges of $\overline{G}_P$ that are incident on M_P; and G' is the graph spanned by the edges of $\overline{G}_P$ that are not incident on M_P. Now, we compute a maximal matching M' of G', and then execute the following procedure:

(1) $M_G \leftarrow M'$
(2) For every edge $e \equiv (u, v) \in M_G$:
(3) $\quad VC_G \leftarrow VC_G \cup \{u, v\}$
(4) $\quad$ Update G_R by removing all the edges in them that are incident on u and v

Figure 6.1: Adding vertices to V_G by picking both end points of every edge in M'.

Note that the above procedure removes every edge of G' since M' is a maximal matching of G'. Now, we will find a vertex cover and maximal matching of updated G_R. Note that any maximal matching of G_R can be combined with that of M' to form a maximal matching of the original graph G since we already removed the edges which were incident on M'.

Note that G_R is composed of the edge sets E_P and a subset of edges from E_I. Therefore, G_P is also a subgraph of G_R. Recall that M_P is a maximal matching of G_P. We define a new edge set U_P that denote the set of unmatched edges in G_P, i.e., $U_P = E_P \setminus M_P$. Note that $|U_P| \leq 2t_1'/3 - 4\delta k$ since $|M_P| > t_1'/3 + 4\delta k$. Moreover, every edge in U_P is incident on M_P, since M_P is a maximal matching of G_P. In the next two procedures, we remove some edges from U_P and M_P such that the updated U_P only contains those edges that are incident on one edge of the updated M_P. Then, we will use this graph to obtain a vertex cover of G of size at most $2k - 2\delta k$. Following is the first procedure:

```
Procedure 1
(1)  while there is an edge e ≡ (u, v) ∈ M_P that is incident on at least two edges of U_P
(2)     M_G ← M_G ∪ {e}
(3)     VC_G ← VC_G ∪ {u, v}
(4)     Update G_R, M_P, and U_P by removing all the edges in them that are
           incident on u and v
```

Figure 6.2: Adding vertices to V_G by picking both end points of every edge in M_P that is incident on at least two edges of U_P.

Note that before the beginning of this procedure there were at most $\leq 2t'_1/3 - 4\delta k$ edges in U_P and at least $t'_1/3 + 4\delta k$ edges in M_P. Then, the above procedure removes at least two edges from U_P and one edge from M_P in each iteration of the while loop. Therefore, at the end of the procedure at least $6\delta k$ edges remain in M_P. Moreover, at the end of the procedure, M_P has the property that no two edges in U_P are incident on the same edge of M_P. Next, consider the following procedure:

```
Procedure 2
(1)  while there is an edge e ≡ (u, v) ∈ U_P that is incident on two edges e_1, e_2 ∈ M_P
(2)     Arbitrarily pick one edge from {e_1, e_2}. W.l.o.g., let e_1 ≡ (u, v) be that edge.
(3)     M_G ← M_G ∪ {e_1}
(4)     VC_G ← VC_G ∪ {u, v}
(5)     Update G_R, M_P, and U_P by remove all the edges in them that are
           incident on u and v
```

Figure 6.3: Adding vertices to V_G on the basis of the edges in U_P that are incident on two edges of M_P.

Let $p \geq 6\delta k$ denote the number of edges in M_P, before the beginning of the above procedure. Now, observe that the above procedure removes one edge from U_P and one edge from M_P in each iteration of the while loop. Furthermore, the while loop executes at most $p/2$ times since M_P had the property that two edges of U_P do not incident on the same edge of M_P. Therefore, at the end of the procedure, M_P contains at least $p/2 \geq 3\delta k$ edges. Moreover, U_P has obtained the property that all its edges are incident on exactly one edge of M_P. Now, recall that all edges

in E_I are also incident on exactly one edge of M_P. We proved this property earlier (just at the beginning of the proof) for every edge of $\overline{G}_p$ that was incident on G_p. Therefore, at this point, G_R consists of the edge set M_P of size at least $3\delta k$, and the edges that are incident on exactly one edge of M_P. Now, we will find a maximal matching and vertex cover of the remaining graph G_R.

We color the edges of M_P with red color and the remaining edges of G_R with blue color. Now, let us define the concept of "*plank edge*". A plank edge is red edge $e \equiv (u, v) \in M_P$ that satisfies the following two conditions. The first condition is that at least one blue edge in G_R is incident on u and at least one incident on v. Let e_u and e_v denote one of the blue edges incident on u and v, respectively. The second condition is that e_u and e_v should be vertex disjoint from every edge of M_G (with respect to the current set M_G which keeps getting updated). In other words, we should be able to add e_u and e_v to M_G. Note that e_u and e_v do not share any common vertex; otherwise it would form a triangle. Therefore, we can add both of them to M_G. Now, we complete the construction of the maximal matching M_G using the following procedure.

```
Procedure 3
(1)  T ← ∅
(2)  M_Y ← ∅    *(this variable accumulates plank edges below)*
(3)  M_N ← M_P    *(variable for non-plank edges)*
(4)  while there is a plank edge e ≡ (u, v) ∈ M_N
(5)       M_Y ← M_Y ∪ {e}
(6)       M_N ← M_N \ {e}
(7)       T ← T ∪ {e_u, e_v}
(8)       M_G ← M_G ∪ {e_u, e_v}
(9)  M_G ← M_G ∪ M_N
```

Figure 6.4: Adding edges to M_G by picking two blue edges each of which is incident on different vertices of a plank edge. Then, adding the remaining non-plank red edges to M_G.

Note that the above procedure adds one edge in M_Y and two edges in T in every iteration of the while loop. Therefore, $|T| = 2 \cdot |M_Y|$. Also, note that $T \subseteq M_G$. Now, we complete the construction of the vertex cover VC_G. We consider two sub-cases based on the size of M_Y.

And, for each of the sub-cases, we construct the vertex cover separately.

1. Sub-case: ($|M_Y| \geq \delta k$)

   For every edge in M_P, we simply add both its endpoints to VC_G. This completes the construction of VC_G. It covers all edges of G_R since all edges are incident on some edge of M_P. Let us compute the size of VC_G. Note that for every edge in $M_Y \subseteq M_P$, there are two edges in $T \subseteq M_G$. Therefore, the size of vertex cover is:

   $$|VC_G| = 2|M_G| - |T| = 2|M_G| - 2|M_Y| \leq 2|M_G| - 2\delta k \leq 2k - 2\delta k$$

2. Sub-case: ($|M_Y| < \delta k$)

   For this sub-case, we construct the vertex cover in the following manner. For every edge in T, we add its both endpoints to VC_G. And, we remove all the edges in G_R covered by them. The remaining graph contains the set M_N, and some blue edges incident on it. Since M_N is defined on non-plank edges, the remaining blue edges can not incident on both endpoints of any edge of M_N. Now, for every edge in M_N, we pick its that endpoint in the vertex cover that it shares with the blue edges incident on it. It completes the construction of VC_G, and it covers all edges of G_R. Note that for every edge in M_G, we added its both endpoints to VC_G except the edges that came from M_N for which, we just added one endpoint in VC_G. Also, note that $|M_N| = |M_P| - |M_Y| > 3\delta k - \delta k = 2\delta k$. Therefore, the size of the vertex cover is:

   $$|VC_G| = 2|M_G| - |M_N| < 2|M_G| - 2\delta k \leq 2k - 2\delta k$$

Hence, we have a vertex cover of size at most $2k - 2\delta k$. This completes the proof of the lemma.

□

Based on the above lemma, we assume that all single edge clusters can be covered with $(\frac{2t_1'}{3} + 8\delta k) \leq (\frac{2t_1}{3} + 8\delta k)$ vertices; otherwise the graph has a vertex cover of size at most $(2k - 2\delta k)$ and the soundness proof would be complete. Now, we bound the vertex cover of the entire graph in the following manner.

$$
\begin{aligned}
&|VC(G)| \\
&\leq |VC\left(\cup_{i=1}^{t_1'} P_i\right)| + |VC'| \\
&= |VC\left(\cup_{i=1}^{t_1'} P_i\right)| + |VC\left((\cup_{j=1}^{t_2} S_j) \cup (\cup_{k=1}^{t_3} W_k) \cup (\cup_{l=1}^{t_4} Y_l)\right)| \\
&\leq \sum_{i=1}^{t_1'} |VC(P_i)| + \sum_{i=1}^{t_2} |VC(S_i)| + \sum_{i=1}^{t_3} |VC(W_i)| + \sum_{i=1}^{t_4} |VC(Y_i)| \\
&\leq \left(\frac{2t_1}{3} + 8\delta k\right) + t_2 + \sum_{i=1}^{t_3} \left(\left(\sqrt{2}+1\right)\delta(W_i) + 1.62\right) + \sum_{i=1}^{t_4} \left(\left(\sqrt{2}+1\right)\delta(Y_i) + 1.8\right), \\
&\qquad\qquad \text{(using Lemmas 49, 50, and 51)} \\
&= (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + \left(\sqrt{2}+1\right)\left(\sum_{i=1}^{t_3}\delta(W_i) + \sum_{i=1}^{t_4}\delta(Y_i)\right)
\end{aligned}
$$

Since the optimal cost $\text{OPT}(C,k) = \sum_{j=1}^{k}\sqrt{m_j(m_j-1)} + \sum_{i=1}^{t_3}\delta(W_i) + \sum_{i=1}^{t_4}\delta(Y_i) \leq m - k/2 + \delta k$, we get $\sum_{i=1}^{t_3}\delta(W_i) + \sum_{i=1}^{t_4}\delta(Y_i) \leq m - k/2 + \delta k - \sum_{j=1}^{k}\sqrt{m_j(m_j-1)}$. We substitute this value in the previous equation, and get the following inequality:

$$
\begin{aligned}
|VC(G)| \leq (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 \\
+ \left(\sqrt{2}+1\right)\cdot\left(m - k/2 - \sum_{j=1}^{k}\sqrt{m_j(m_j-1)} + \delta k\right)
\end{aligned}
$$

Using Lemma 44, we obtain the following inequalities:

1. For any cluster P_j with $|P_j| = 1$, we have $\sqrt{|P_j|\,(|P_j|-1)} \geq |P_j| - 1$
2. For any cluster S_j with $|S_j| \geq 2$, we have $\sqrt{|S_j|\,(|S_j|-1)} \geq |S_j| - (2-\sqrt{2})$
3. For any cluster W_j with $|W_j| \geq 2$, we have $\sqrt{|W_j|\,(|W_j|-1)} \geq |W_j| - (2-\sqrt{2})$
4. For any cluster Y_j with $|Y_j| \geq 3$, we have $\sqrt{|Y_j|\,(|Y_j|-1)} \geq |Y_j| - (3-\sqrt{6})$

We substitute these values in the previous equation, and get the following inequality:

$$|VC(G)| \leq (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + \left(\sqrt{2}+1\right)\cdot\left(m - k/2 - \sum_{j=1}^{t_1}(|P_j|-1) - \sum_{j=1}^{t_2}\left(|S_j| - (2-\sqrt{2})\right) - \sum_{j=1}^{t_3}\left(|W_j| - (2-\sqrt{2})\right) - \sum_{j=1}^{t_4}\left(|Y_j| - (3-\sqrt{6})\right) + \delta k\right)$$

Since the number of edges $m = \sum_{j=1}^{t_1}|P_j| + \sum_{j=1}^{t_2}|S_j| + \sum_{j=1}^{t_3}|W_j| + \sum_{j=1}^{t_4}|Y_j|$, we get the following inequality:

$$|VC(G)| \leq (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + \left(\sqrt{2}+1\right)\cdot\left(-k/2 + t_1 + t_2\cdot\left(2-\sqrt{2}\right) + + t_3\cdot\left(2-\sqrt{2}\right) + t_4\cdot\left(3-\sqrt{6}\right) + \delta k\right)$$

We substitute $k = t_1 + t_2 + t_3 + t_4$, and obtain the following inequality:

$$\begin{aligned}
|VC(G)| &\leq (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + \left(\sqrt{2}+1\right)\cdot\left(\frac{t_1}{2} + \frac{t_2}{10} + \frac{t_3}{10} + \frac{3t_4}{50} + \delta k\right) \\
&= (1.88)t_1 + (1.25)t_2 + (1.87)t_3 + (1.95)t_4 + \left(\sqrt{2}+9\right)\delta k \\
&< (1.95)k + \left(\sqrt{2}+9\right)\delta k \qquad \text{(using } t_3 + t_4 + t_1 + t_2 = k\text{)} \\
&\leq (2-\varepsilon)k, \quad \text{for appropriately small constants } \varepsilon, \delta > 0
\end{aligned}$$

This proves the soundness condition and it completes the proof of Theorem 67. Note that the

result holds under the Unique Games Conjecture. To prove the result in a weaker assumtion of $\mathsf{P} \neq \mathsf{NP}$, it would require to show that $|VC(G)| < (1.36)k$ instead of $|VC(G)| < (1.95)k$. That would require tighter analysis of the cost of k-median instances than the one done in this work.

6.6 Vertex Cover of Non-Star Graphs

In this section, we bound the vertex cover size of any non-star graph F. We aim to obtain this bound in terms of $\delta(F)$, i.e., the extra cost of the graph F. To do so, we require a bound on the extra cost. The 1-median cost of an arbitrary non-star graph is tricky to compute. Fortunately, we do not require the exact optimal cost of a graph; a lower bound on the optimal cost suffices. Furthermore, for some graph instances we can compute the exact optimal cost. For example, a star graph with r edges corresponds to a regular simplex of size length s that has the optimal 1-median cost: $s \cdot \sqrt{\frac{r(r-1)}{2}}$ We proved this earlier in Lemma 48. For the more complex graph instances, we use the following decomposition lemma to bound their optimal cost.

Lemma 52 (Decomposition lemma)**.** *Let $G = (V, E)$ be any graph and let $E_1, ..., E_t$ be any partition of the edges and let $G_1, G_2, ..., G_t$ be the subgraphs induced by these edges respectively. The following inequality bounds the optimal 1-median cost of G in terms of the optimal 1-median costs of subgraphs $G_1, ..., G_t$.*

$$\text{cost}^*(G) \geq \text{cost}^*(G_1) + \ldots + \text{cost}^*(G_t)$$

Proof. Let f^* be an optimal median of $C(G)$. For any edge $e \in E$, let x_e denote the corresponding point in $C(G)$. The proof follows from the following sequence of inequalities: $\text{cost}^*(G) = \sum_{e \in E} d(x_e, f^*) = \sum_{i=1}^{t} \sum_{e \in E_i} d(x_e, f^*) \geq \sum_{i=1}^{t} \text{cost}^*(G_i)$. □

If we can compute the optimal 1-median cost for each subgraph, then we can lower bound the overall cost of the graph using the above decomposition lemma. In general, a graph may be

decomposed in various ways. However, we prefer that decomposition that gives better bound on the optimal cost. In the next subsection, we use this decomposition lemma to bound the extra cost of any non-star graph. We will use that bound in subsequent subsections to prove Lemmas 49 and 50.

6.6.1 1-median cost of non-star graphs

In this section, we show that any non-star graph on m edges has the optimal 1-median cost at least $m - 0.342$ which is at least $\sqrt{m(m-1)} + 0.158$. We assume all graphs to be triangle-free since hard Vertex Cover instances we start with are triangle-free. Therefore, we do not explicitly mention the "triangle-freeness" of graphs whenever we state a lemma related to graphs.

To obtain a lower bound on the optimal cost, we decompose a graph into so called "Fundamental non-star graphs". We will show this decomposition process later. For now, we first bound the 1-median cost of these fundamental graphs. We then bound the cost of any graph using the decomposition lemma. Here is the formal description of fundamental graphs.

Definition 50 (Fundamental Non-Star Graph)**.** *A fundamental non-star graph is a graph that becomes a star graph when **any** pair of vertex-disjoint edges are removed from it. The graph that has only two vertex-disjoint edges is also a fundamental non-star graph.*

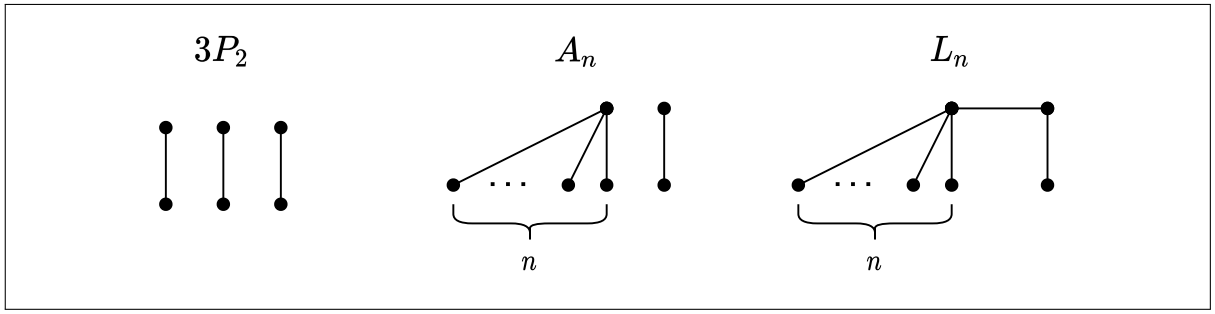

Figure 6.5: Fundamental non-star graphs: 3-P_2, A_n, and L_n.

The following lemma shows that there are exactly three types of fundamental non-star graphs. These are shown in Figure 6.5.

Lemma 53. *There are only three types of fundamental non-star graphs:* 3-P_2*,* A_n*, and* L_n*, for* $n \geq 1$*. These are shown in Figure 6.5.*

Proof. It is easy to see that the graphs 3-P_2, A_n, and L_n, shown in Figure 6.5, are fundamental non-star graphs. We need to argue that these are the *only* fundamental non-star graphs. For this we do a case analysis. Let M denote a maximum matching of any graph $G = (V, E)$. We divide the analysis into three cases based on the size of the maximum matching.

1. Case 1: $|M| \geq 4$.

   Suppose we remove any pair of vertex-disjoint edges from M. Then, the remaining graph is still a non-star graph due to a matching of size at least two. Therefore, any such graph can not be a fundamental non-star graph.

2. Case 2: $|M| = 3$.

   Let U denote the set of unmatched edges in the graph, i.e, $U = E \setminus M$. Suppose $U \neq \emptyset$, and l be any edge in U. Observe that l can be incident on at most two edges of M. Since the size of matching M is three, there is always an edge in M that is vertex-disjoint from l. Let this edge be $m_e \in M$. The set of edges $\{l, m_e\}$ forms a 2-P_2 (i.e., two vertex-disjoint edges), and we can remove it from G. The remaining graph still has a matching of size at least two. Therefore, such a graph can not be a fundamental non-star graph. Now, let us consider the case when $U = \emptyset$. In this case, the graph is equivalent to a 3-P_2.

3. Case 3: $|M| = 2$.

   Let $e_1 \coloneqq (u_1, v_1)$, and $e_2 \coloneqq (u_2, v_2)$ be the matched edges in M. Let $U = E \setminus M$ denote the set of unmatched edges. If U forms a non-star graph, we can remove a pair of vertex-disjoint edges from it. The remaining graph is a non-star due to $|M| = 2$. Therefore, any graph with $|M| = 2$ and U as a non-star graph, can not be a fundamental non-star graph. Therefore, let us consider the case when U forms a star graph. Suppose all edges

of U share a common vertex w. There are two possibilities: w is an endpoint of some matching edge or not. Let us consider these two possibilities one by one.

(a) Subcase: w is an endpoint of some matched edge

Without loss of generality, we can assume $w \equiv u_1$. Now, no two edges of U can be incident on u_2 and v_2; otherwise, it would form a triangle (w, u_2, v_2). Therefore, at most one edge of U can be incident on either u_2 or v_2. If such an edge exists, without loss of generality, we can assume that it is incident on u_2. This graph is of type L_n. On the other hand, if no edge of U is incident on either u_2 or v_2, the graph is of type A_n.

(b) Subcase: w is not an endpoint of any matched edge.

If $|U| = 1$, the graph is simply A_1. Now, let us assume that $|U| \geq 2$. Let l_1 and l_2 be any two edges in U. The edges l_1 and l_2, can not be incident on the same matched edge, say (u_1, v_1). This is because then it would form a triangle: (w, u_1, v_1). Also, note that every edge $e \in U$ must be incident on some matching edge; otherwise $M \cup \{e\}$ would form a matching of size $> |M|$. It would contradict that M is a maximum matching. Therefore, without loss of generality, we can assume that l_1 is incident on u_1, and l_2 is incident on u_2. Now, it forms a path of length four: (v_1, u_1, w, u_2, v_2). We can always remove an alternating pair of edges from the path, and the resulting graph still contains a pair of vertex-disjoint edges, i.e., a 2-P_2. Therefore, any such graph can not be a fundamental non-star graph.

From the above case analysis we conclude that all fundamental non-star graphs are of type 3-P_2, A_n, or L_n. This completes the proof of the lemma. □

Next, we bound the optimal 1-median cost of each fundamental non-star graph. In this discussion, we will use r to denote the number of edges in various cases.

Lemma 54. *Let $r \coloneqq 3$ denote the number of edges in 3-P_2. The optimal cost of 3-P_2 is at least $(r + 0.46)$.*

Proof. It is easy to see that $C(3\text{-}P_2)$ forms a simplex of side length 2. Therefore, we can use Corollary 28 to compute its optimal 1-median cost: $\text{cost}^*(3\text{-}P_2) = \sqrt{2}\sqrt{3(3-1)} = 2\sqrt{3} \geq r + 0.46$ (since $r = 3$). □

Lemma 55. *Let $r \coloneqq n + 1$ denote the number of edges in A_n. The optimal cost of A_n is at least:*

1. *r, for $n \geq 1$.*
2. *$r + 0.095$, for $n \geq 2$.*
3. *$r + 0.135$, for $n \geq 3$.*

Proof. Consider the point set $C(A_n)$. It forms a simplex with $(r-1)$ points at a distance of $\sqrt{2}$ from each other and the remaining point at a distance of 2 from the rest of the points. Based on Lemma 45, we represent the coordinates of every point in an $(r-1)$-dimensional space in the following way.

$$a_1 = (1, 0, ..., 0), \quad a_2 = (0, 1, ..., 0), \quad ..., \quad a_{r-1} = (0, ..., 0, 1), \quad a_r = (u, \ldots, u)$$

Here $u = \frac{1}{r-1} + \sqrt{\frac{3}{r-1} + \frac{1}{(r-1)^2}}$. Let $(c_1, \ldots, c_{r-1})$ be an optimal 1-median of $S = \{a_1, ..., a_r\}$. If $c_i \neq c_j$ for any $i \neq j$, we can swap c_i and c_j to create a different median with the same 1-median cost. Since the 1-median is always unique for a set of non-collinear points (by Fact 5), we assume $c^* = \{c, \ldots, c\}$ as the optimal median. Then, the optimal 1-median cost is:

$$\text{cost}(c^*, S) = \text{cost}(c^*, a_r) + \sum_{i=1}^{r-1} \text{cost}(c^*, a_i)$$

$$= (u-c)\cdot\sqrt{r-1} + (r-1)\cdot\sqrt{(1-c)^2 + (r-2)c^2}$$

The function is strictly convex and attains minimum at $c = \frac{\sqrt{r}+1}{\sqrt{r}\cdot(r-1)}$. We get the following optimal cost on substituting the values of t and c in the previous equation:

$$\text{cost}(c^*, S) = \sqrt{r(r-1)} + \sqrt{3 + \frac{1}{r-1}} - \sqrt{\frac{r}{r-1}} = \sqrt{r(r-1)} + \frac{2}{\sqrt{3+\frac{1}{r-1}} + \sqrt{\frac{r}{r-1}}}$$

For $r \geq t$, we get:

$$\begin{aligned}
\text{cost}(c^*, S) &\geq \sqrt{r(r-1)} + \frac{2}{\sqrt{3+\frac{1}{t-1}} + \sqrt{\frac{t}{t-1}}}, \qquad (\because r \geq t) \\
&= \sqrt{r(r-1)} + \sqrt{3 + \frac{1}{t-1}} - \sqrt{\frac{t}{t-1}} \\
&\geq r - (t - \sqrt{t(t-1)}) + \sqrt{3 + \frac{1}{t-1}} - \sqrt{\frac{t}{t-1}}, \qquad \text{(using Lemma 44)}
\end{aligned}$$

Substituting $t = 2$, we get: $\text{cost}(c^*, S) \geq r - (2 - \sqrt{2}) + \sqrt{4} - \sqrt{2} \geq r$.

Substituting $t = 3$, we get: $\text{cost}(c^*, S) \geq r - (3 - \sqrt{6}) + \sqrt{7/2} - \sqrt{3/2} > r + 0.095$.

Substituting $t = 4$, we get: $\text{cost}(c^*, S) \geq r - (4 - \sqrt{12}) + \sqrt{10/3} - \sqrt{4/3} > r + 0.135$.

This completes the proof of the lemma. □

Lemma 56. *Let $r \coloneqq n + 2$ denote the number of edges in L_n. Then the optimal cost of L_n is at least:*

1. $r - 0.268$, *for* $n = 1$.
2. $r - 0.334$, *for* $n = 2$.
3. $r - 0.342$, *for* $n \geq 3$.

Proof. Let us prove the first statement for $r = 3$ which corresponds to the graph being L_1 (i.e., $n = 1$). In $C(L_1)$, there are two points at distance of 2 from each other, and the third point at a distance of $\sqrt{2}$ from the other two points. It forms a simplex S of dimension two. Based on Lemma 45, we represent the coordinates of the simplex in the following way:

$$a_1 = (0, 0), \quad a_2 = (\sqrt{2}, 0), \quad a_3 = (0, \sqrt{2}).$$

Note that the pairwise distances are preserved in this representation. Let (c_1, c_2) be the optimal 1-median of S. If $c_i \neq c_j$ for any $i \neq j$, we can swap c_i and c_j to create a different median with the same 1-median cost. Therefore, we consider $c^* \coloneqq (c, c)$ as the optimal 1-median of S. Then, we get the following optimal 1-median cost of S.

$$\text{cost}(c^*, S) = \text{cost}(c^*, a_1) + \text{cost}(c^*, a_2) + \text{cost}(c^*, a_3) = \sqrt{2} \cdot c + 2 \cdot \sqrt{\left(c - \sqrt{2}\right)^2 + c^2}$$

The function is strictly convex and attains minimum at $c = \sqrt{\frac{1}{2}} - \sqrt{\frac{1}{6}}$. Substituting the value of c in $\text{cost}(c^*, S)$, we get $\text{cost}(c^*, S) = 1 + \sqrt{3} > r - 0.268$, for $r = 3$. This completes the proof of the first statement.

Let us prove the second statement. Here, we have $r = 4$ (or $n = 2$). We create three copies of L_2 (i.e., 3-L_2), and decompose them into three subgraphs: 2-P_4, A_2, and S_3. The decomposition is shown in Figure 6.6. Note that P_4 is the same as L_1. There are also other ways to decompose the graph. However, some of those decompositions give weak bound on the optimal cost. And, this decomposition gives sufficiently good bound on the optimal cost of L_2.

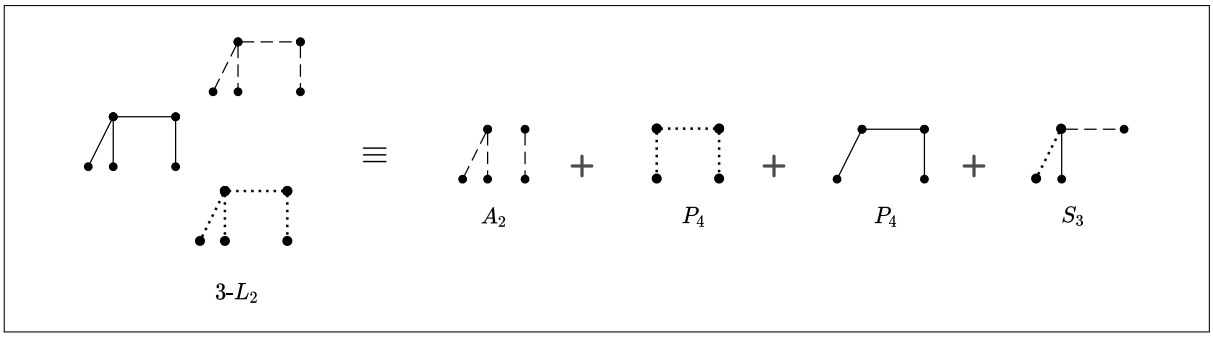

Figure 6.6: Decomposition of 3-L_2

Let c^* be the optimal 1-median for L_2. Based on the decomposition, we bound the optimal 1-median cost of L_2 as follows:

$$3 \cdot \text{cost}^*(L_2) \geq \text{cost}^*(A_2) + 2 \cdot \text{cost}^*(P_4) + \text{cost}^*(S_3) \tag{6.2}$$

We already have the bounds on the optimal costs of A_2, P_4, and S_3. That is,

- For A_2, we have $\text{cost}^*(A_2) \geq 3 + 0.095$. This follows from Statement 2 of Lemma 55.
- For P_4, we have $\text{cost}^*(P_4) \geq |P_4| - 0.268 = 2.732$. This follows from Statement 1 of Lemma 56 since P_4 is the same as L_1, and that the number of edges in P_4, denoted by $|P_4|$, equals 3.
- For S_3, we have $\text{cost}^*(S_3) = \sqrt{3(3-1)} = \sqrt{6}$. This follows from Corollary 27, for $r = 3$.

Substituting the above values in Equation (6.2), we get the following inequality:

$$3 \cdot \text{cost}^*(L_2) \geq 3.095 + 2 \cdot (2.732) + \sqrt{6} > 11$$

Thus, the optimal cost of L_2 is at least $11/3 \geq 3.666 = |L_2| - 0.334$. This completes the proof of Statement 2.

Now, let us prove the third statement. Here, we have $r \geq 5$ (or $n \geq 3$). We create two copies of L_n, and decompose it into three subgraphs: A_n, S_n, and P_4. The decomposition is shown in Figure 6.7. Again, note that there are many ways to decompose the graph. However, those decompositions may yield weak bound on the optimal cost. Whereas, this decomposition gives sufficiently good bound on the optimal 1-median cost of L_n.

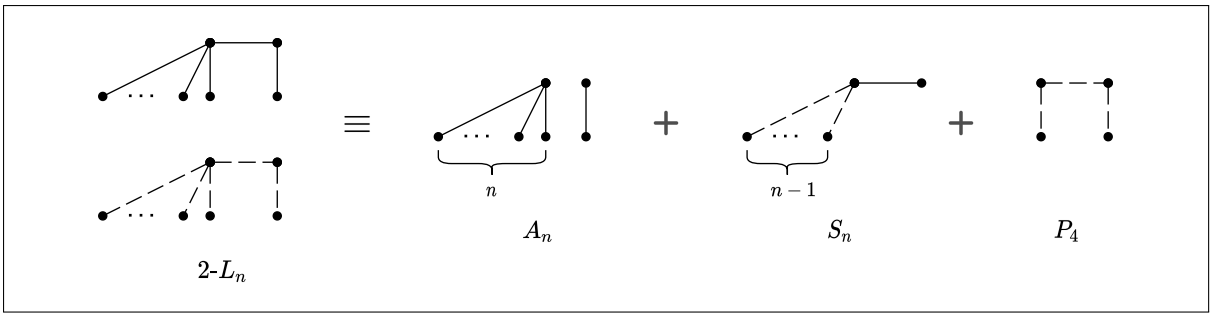

Figure 6.7: Decomposition of 2-L_n for $n \geq 3$

Let c^* be the optimal 1-median for L_n. Based on the decomposition, we bound its optimal 1-median cost in the following manner:

$$2 \cdot \text{cost}^*(L_n) \geq \text{cost}^*(A_n) + \text{cost}^*(S_n) + \text{cost}^*(P_4) \tag{6.3}$$

We already know the bounds on the optimal costs of A_n, S_n and P_4. That is,

- For A_n, we have $\text{cost}^*(A_n) \geq (n+1) + 0.135$. This follows from Statement 3 of Lemma 55 ($\because n \geq 3$).
- For S_n, we have $\text{cost}^*(S_n) = \sqrt{n(n-1)} \geq n - (3 - \sqrt{6})$. The first equality follows from Corollary 27) whereas the second inequality follows from Lemma 44, for $n \geq 3$.
- Since P_4 is is the same as L_1, we have $\text{cost}^*(P_4) \geq |P_4| - 0.268$. This follows from Statement 1 of Lemma 56.

Substituting the above values in Equation (6.3) and using the fact that $|P_4| = 3$, we obtain the following inequality:

$$2 \cdot \text{cost}^*(L_n) \geq (2n + 1 + |P_4|) + (0.135 + \sqrt{6} - 3 - 0.268) \geq 2 \cdot (n + 2) - 0.684$$

Thus, the optimal cost of L_n is at least $(n + 2) - 0.342 = |L_n| - 0.342$. This completes the proof of the lemma. □

Corollary 29. *The cost of any fundamental non-star graph with r edges is at least $r - 0.342$.*

Proof. The proof simply follows from Lemmas 54, 55, and 56. □

We will now bound the cost of *any* non-star graph by decomposing it into fundamental non-star graphs. For this, we define the concept of "*safe pair*". A safe pair is a pair of vertex-disjoint edges in the graph such that when we remove it from the graph, the remaining graph remains a non-star graph. Let us see why such a pair is important. First, note that the optimal cost of a 2-P_2 is exactly two, using Corollary 28. Suppose we remove a 2-P_2 from the graph F. Let the remaining graph be F'. Suppose $\text{cost}^*(F') = |F'| + \gamma$, where γ is some constant. Then it is easy to see that $\text{cost}^*(F) \geq \text{cost}^*(F') + \text{cost}^*(2\text{-}P_2) = |F| + \gamma$. Note that γ value is preserved in this decomposition. Suppose we keep removing a safe pair from F until we obtain a graph that does not contain any safe pair. Then the remaining graph is simply a fundamental non-star graph by the definition of fundamental non-star graph. Moreover, we showed earlier that the optimal cost of any fundamental non-star graph with r edges, is at least $r - 0.342$. Note that here $\gamma \geq -0.342$. Therefore, F has the optimal cost at least $|F| - 0.342$. This is the main reason for defining a safe pair and a fundamental non-star graphs. The decomposition procedure described above is given below.

```
Decompose(F)
        Input: A non-star graph F.
        Output: A fundamental non-star graph D ∈ {3-P_2, A_n, L_n}
(1)     D ← F
(2)     while D ∉ {3-P_2, A_n, L_n}
(3)         Let {e, e'} ∈ E(D) be a safe pair
(3)         E(D) ← E(D) \ {e, e'}
(4)     return D
```

Figure 6.8: Decomposition of any non-star graph F into the *fundamental* non-star graphs.

The above discussion is formalised as the next lemma.

Lemma 57. *The cost of any non-star graph F is at least* $(|F| - 0.342) \geq \sqrt{|F|(|F|-1)} + 0.158$.

Proof. Suppose the procedure `Decompose`(F) runs the while loop t times. This means that F is composed of t safe pairs and a fundamental non-star graph $D \in \{3\text{-}P_2, A_n, L_n\}$. We call D the *residual* graph of F. Note that D has exactly $|F| - 2t$ edges. Also, note that $2\text{-}P_2$ is a fundamental non-star graph since it the same as A_1. Based on the decomposition of F into $2\text{-}P_2$'s and D, we bound the optimal cost of F as follows:

$$\begin{aligned}
\text{cost}^*(F) &\geq t \cdot \text{cost}^*(2\text{-}P_2) + \text{cost}^*(D) \\
&= 2 \cdot t + \text{cost}^*(D) && \text{(using Corollary 28)} \\
&\geq 2 \cdot t + (|F| - 2t) - 0.342 && \text{(using Corollary 29)} \\
&= |F| - 0.342 \\
&> \sqrt{|F|(|F|-1)} + 1/2 - 0.342 && \text{(using Lemma 44)} \\
&= \sqrt{|F|(|F|-1)} + 0.158
\end{aligned}$$

This completes the proof of the lemma. □

Next, we show a stronger bound on the optimal cost than the one stated in the previous lemma. However, this bound applies for a particular set of graph instances. For positive integers p, q, let us define a new graph $L_{p,q}$. This graph is composed of two star graphs S_p and S_q, such that there is an edge between the center vertices of S_p and S_q. Here, the *center* vertex is the vertex that is the common endpoint of all edges in a star graph and the remaining vertices are called *pendent* vertices. Let s_1 and s_2 denote the center vertices of S_p and S_q respectively. We call the edge (s_1, s_2), the *bridge* edge, and the graph $L_{p,q}$, the *bridge* graph. Also, we call the pendent vertices of S_p and S_q as the *left* and *right* pendent vertices respectively. Please see Figure 6.9 for the pictorial depiction of $L_{p,q}$. Note that when $p = n$ and $q = 1$, the bridge graph is the same as L_n.

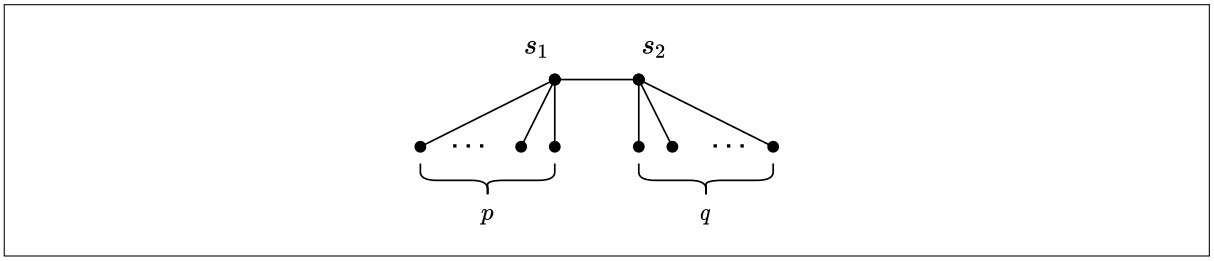

Figure 6.9: A Bridge Graph: $L_{p,q}$, for $p, q \geq 1$.

Now, we state the lemma that bounds the optimal 1-median cost of any non-star non-bridge graph.

Lemma 58. *Suppose F is a non-star non-bridge graph. Then F has the optimal 1-median cost at least $|F| \geq \sqrt{|F|(|F|-1)} + 0.5$.*

Proof. Here, we need to define the new concept of "*ultra-safe*" pair. An ultra-safe pair is a pair of vertex-disjoint edges such that removing it from the graph does not make the resulting graph a star or a bridge graph. We decompose F in a similar way as we did before. However, instead of removing a safe-pair from the graph, we remove an ultra-safe pair in every iteration of the while loop. We decompose F using the following procedure.

```
UltraDecompose(F)
        Input: A non-star non-bridge graph F.
        Output: A fundamental non-star graph D ∈ {3-P_2, A_n, L_n}
(1)     D ← F
(2)     while D ∉ {3-P_2, A_n, L_n}
(3)         Let {e, e'} ∈ E(D) be an ultra safe pair
(3)         E(D) ← E(D) \ {e, e'}
(4)     return D
```

Figure 6.10: Decomposition of a non-star non-bridge graph F into fundamental non-star graphs.

First, note that the procedure `UltraDecompose`(F) produces a residual graph of type 3-P_2 or A_n. It does not produce a residual graph of type L_n since we are always removing an ultra-safe pair from the graph, and L_n is equivalent to $L_{n,1}$. Next, we show that we can always remove an ultra-safe pair from G until we obtain a graph of type 3-P_2 or A_n. Consider the i^{th} iteration of the while loop given that it is executed. Let D be the graph at the start of this iteration. It is clear that D is neither a 3-P_2 nor A_n; otherwise, this while loop would not have been executed. Also, D can neither be a star nor a bridge graph since an ultra-safe pair was removed in the previous iteration. This fact also holds for the first iteration since the input graph is neither a star nor a bridge graph. It implies that D is a non-star graph but not a fundamental non-star graph. Since the graph is not a fundamental non-star graph, it must contain a safe pair. Let $e_1 \equiv (u_1, v_1)$ and $e_2 \equiv (u_2, v_2)$ form a safe pair in D. If $\{e_1, e_2\}$ is also an ultra-safe pair, we are done. On the other hand, if $\{e_1, e_2\}$ is not an ultra-safe pair, we show that there is another ultra-safe pair in D. Since $\{e_1, e_2\}$ is a safe pair but not an ultra safe-pair, removing it would make the resulting graph, a bridge graph $L_{p,q}$. Therefore, D is composed of a graph of type $L_{p,q}$, and the two additional edges: e_1 and e_2. Let $b \equiv (s_1, s_2)$ denote the bridge edge of $L_{p,q}$. We split the analysis into two cases, based on the orientation of e_1 and e_2 in the graph. For each case, we show that D contains an ultra-safe pair.

- Case-I: *At least one of the two edges, e_1, e_2 connects a left pendent vertex with a right*

one.

Without loss of generality, let $e_1 \equiv (u_1, v_1)$ be the edge that connects a left pendent vertex u_1 with a right pendent vertex v_1. We claim that e_1 and the bridge edge b, form an ultra-safe pair. Indeed, suppose we remove this pair from the graph. The resulting graph would be a non-star graph since (s_1, u_1) and (s_2, v_1) are still present in the graph, and they form a 2-P_2. Therefore, the pair $\{e_1, b\}$ satisfies the condition of being a safe pair. Furthermore, the resulting graph is a non-bridge graph since there is no common edge incident on (s_1, u_1) and (s_2, v_1). This proves that $\{e_1, b\}$ is an ultra-safe pair.

- Case-II: *Neither e_1 nor e_2 connects any left pendent vertex with a right one.*

  First, let us consider the possibility that both the edges e_1 and e_2 are incident on the bridge edge, i.e., e_1 is incident on s_1 and e_2 is incident on s_2. Then the graph D is a bridge graph of the form $L_{p+1,q+1}$ which is not possible as per earlier discussion. Hence, without loss of generality, we can assume that e_1 is not incident on b. Now, we claim that the pair $\{e_1, b\}$ forms an ultra-safe pair. Note that it forms a 2-P_2 since both edges are vertex-disjoint. Now, suppose we remove this pair from the graph. Then in the resulting graph S_p and S_q are not connected by any edge since e_2 does not connect left and right pendent vertices. Such a graph can neither be a bridge graph or a star graph, since both of these graphs are connected graphs.

The above discussion implies that there D always contains an ultra-safe pair unless D is of type 3-P_2 or A_n. This means that if the procedure `UltraDecompose`(F) runs the while loop 't' times, then F is composed of t ultra-safe pairs and a fundamental non-star graph $D \in \{3\text{-}P_2, A_n\}$. Based on this decomposition, we bound the optimal cost of F in the following manner:

$$\begin{aligned}\text{cost}^*(F) &\geq t \cdot \text{cost}^*(2\text{-}P_2) + \text{cost}^*(D) \\ &= 2 \cdot t + \text{cost}^*(D), \qquad \text{(using Corollary 28)}\end{aligned}$$

$$\geq 2 \cdot t + (|F| - 2t), \qquad \text{(using Lemma 54 and 55)}$$

$$= |F|$$

$$> \sqrt{|F|(|F|-1)} + 1/2, \qquad \text{(using Lemma 44)}$$

This completes the proof of the lemma. □

In the next two subsections, we bound the vertex cover size of any non-star graph F in terms of the extra cost $\delta(F)$.

6.6.2 Vertex cover for matching size two

In this section, we show that any graph with a maximum matching of size exactly two has a vertex cover of size at most $2\left(\sqrt{2}+1\right)\delta(F) + 1.62$. Let C_5 denote a cycle on five vertices. In the following lemma, we show that C_5 is the only graph with a maximum matching of size two and a vertex cover of size three. The rest of the graphs with a maximum matching of size two, have a vertex cover of size two.

Lemma 59. *Let F be any graph other than C_5. If F has a maximum matching of size two, it has a vertex cover of size two. Furthermore, C_5 has a vertex cover of size three.*

Proof. Let M be a maximum matching of F. Let $e_1 \equiv (u_1, v_1)$ and $e_2 \equiv (u_2, v_2)$ denote the edges in M. Let V_M denote the vertex set spanned by M, i.e., $V_M \coloneqq \{u_1, v_1, u_2, v_2\}$. Let U denote the unmatched edges of F. i.e., $U \coloneqq E(F) \setminus M$. Note that all edges in U are incident on at least one of the matching edges; otherwise it forms a matching of size three and this contradicts the fact that F has a maximum matching of size two. Let U_1 denote the edges in U that are incident on exactly one of the matching edges and $U_2 = U \setminus U_1$ be the remaining unmatched edges. In other words, U_2 contains the edges that have their both endpoints in V_M.

First, we claim that no two edges in U_1 can be incident on different endpoints of the same

matching edge. For the sake of contradiction, suppose (x, u_1) and (y, v_1) are the edges in U_1 such that $x, y \notin V_M$. If $x = y$, it forms a triangle (x, u_1, v_1), which is not allowed. If $x \neq y$, we have a matching of size three – $\{(x, u_1), (y, v_1), (u_2, v_2)\}$. This contradicts the fact that F has a maximum matching M of size two. Therefore, U_1 cannot contain both (x, u_1) and (y, v_1). Similarly, U_1 cannot contain both (x, u_2) and (y, v_2). Now, without loss of generality, we can assume that all the edges in U_1 have their one endpoint in the vertex-set: $\{u_1, u_2\}$. Let us divide the remaining analysis into following two cases based on the existence of edge (v_1, v_2) in the graph.

- Case 1: $(v_1, v_2) \notin E(F)$.

  U_2 can only contain the following edges: $(u_1, v_2), (u_2, v_1)$, and (u_1, u_2). Note that all edges in U_2 have at least one endpoint in $\{u_1, u_2\}$. Previously, we showed all edges of U_1 are incident on $\{u_1, u_2\}$. Based on both of these facts, we can cover all edges in U using only two vertices: $\{u_1, u_2\}$. Furthermore, these vertices also cover the matching edges in M. Therefore, all edges of the graph are covered, and we have a vertex cover of size two.

- Case 2: $(v_1, v_2) \in E(F)$.

  Now, note that U_2 can not contain the edges: (u_1, v_2) and (u_2, v_1), since they form the triangles: (v_1, v_2, u_1) and (v_1, v_2, u_2). However, U_2 can contain the edge: (u_1, u_2). Let us consider the following two sub-cases based on the existence of (u_1, u_2) in the graph.

  (a) Sub-case: $(u_1, u_2) \in E(F)$.

  We claim that either all the edges in U_1 are incident on u_1 or all of them are incident on u_2. For the sake of contradiction, let (x, u_1) and (y, u_2) be the edges in U_1 such that $x, y \notin V_M$. If $x = y$, it forms a triangle (x, u_1, u_2), which is not allowed. If $x \neq y$, we get a matching of size three – $\{(x, u_1), (y, u_2), (v_1, v_2)\}$ which contradicts the fact that F has a maximum matching M of size two. Without loss of generality, we can assume that all edges in U_1 are incident on u_1. Therefore, we can cover

all edges of U_1 using only u_1. Furthermore, u_1 covers one of the matching edge $(u_1, v_1) \in M$ and the edge $(u_1, u_2) \in U_2$. Only two edges remain uncovered in the graph, which are $(u_2, v_2) \in M$ and $(v_1, v_2) \in U_2$. We cover both these edges by picking the vertex v_2. Thus, we get a vertex cover of size two.

(b) Sub-case: $(u_1, u_2) \notin E(F)$.

Let us consider the case when all the edges in U_1 are incident on either u_1 or u_2. In this case, either $\{u_1, v_2\}$ or $\{u_2, v_1\}$ forms a vertex cover of size two. Hence, we are done. Let us consider the other case. Suppose, there are two edges (x, u_1) and (y, u_2) in U_1 such that $x, y \notin V_M$. If $x \neq y$, we get a matching of size three – $\{(x, u_1), (y, u_2), (v_1, v_2)\}$, which is not possible. On the other hand, if $x = y$, then the only possibility is that F is a cycle of length five – $C_5 : (x, u_1, v_1, v_2, u_2)$. In this case, the vertex cover of F is of size 3.

We showed that all graphs with maximum matching 2 has a vertex cover of size 2 except for C_5 that has a vertex cover of size 3. This completes the proof of the lemma. □

Corollary 30. *Let F be any graph with a maximum matching of size two. If the graph is not a C_5, it has a vertex cover of size at most $\left(\sqrt{2}+1\right)\delta(F) + 1.62$.*

Proof. In Lemma 57, we showed that any graph F has an extra cost at least 0.158. In other words, $\delta(F) \geq 0.158$. It is easy to see that $|VC(F)| = 2 \leq \left(\sqrt{2}+1\right)\delta(F) + 1.62$. Hence proved. □

Next, we consider the particular case of C_5. The following lemma bounds the optimal 1-median cost of C_5.

Lemma 60. *The optimal 1-median cost of C_5 is at least $\sqrt{|C_5|(|C_5|-1)} + 0.622$.*

Proof. Let C_5 be (u, v, w, x, y). We decompose the graph into two fundamental non-star graphs: $A_1 \colon \{(u, v), (w, x)\}$ and $A_2 \colon \{(v, w), (x, y), (y, u)\}$. The following sequence of inequalities bound the optimal cost of C_5.

$$\begin{aligned}
\text{cost}^*(C_5) &\geq \text{cost}^*(A_1) + \text{cost}^*(A_2) \\
&\geq 2 + (3 + 0.095) && \text{(using Statement 1 and 2, of Lemma 55)} \\
&= \sqrt{20} + \left(5 - \sqrt{20}\right) + 0.095 \\
&= \sqrt{|C_5|(|C_5| - 1)} + 0.622 && (\because |C_5| = 5)
\end{aligned}$$

This completes the proof of the lemma. □

Corollary 31. *The graph C_5 has a vertex cover of size at most $\left(\sqrt{2} + 1\right) \delta(C_5) + 1.62$.*

Proof. Since $\delta(C_5) \geq 0.622$, we get $|VC(C_5)| = 3 \leq \left(\sqrt{2} + 1\right) \delta(C_5) + 1.62$ which proves the corollary. □

6.6.3 Vertex cover for matching size at least three

In this section, we show that any non-star graph F, with a maximum matching of size at least three, has a vertex cover of size at most $1.8 + \left(\sqrt{2} + 1\right) \delta(F)$. First, let us define some notations. Let M denote a maximum matching of F, and G_M denote the subgraph spanned by M. Let F' be the graph obtained by removing M from F, i.e., $F' = (V, E(F) \setminus M)$. Let L denote a maximum matching of F', and G_L denote the subgraph spanned by L. We call L the *second maximum matching* of F after M. Now, we remove L from the graph. Let F'' be the graph obtained by removing L from F', i.e., $F'' = (V, E(F) \setminus (M \cup L))$. Recall that in this entire discussion, we are using $|.|$ to denote the number of edges in any given graph.

Now, we obtain a relation between the vertex cover size and extra cost of a graph. To establish this relation, we show that both of them are proportional to the number of vertex disjoint edges

in the graph. For example, a graph with a maximum matching M has a vertex cover of size at most $2|M|$. Similarly, a set of m vertex-disjoint edges has an extra cost of $\left(\sqrt{2}-1\right) \cdot \sqrt{m(m-1)}$ (using Corollary 28). Also, note that a star graph which has a maximum matching of size one, has an extra cost of only zero. In the next two lemmas, we formally establish these relations of the vertex cover size and extra cost in terms of number of vertex-disjoint edges in the graph. Then we will use the two lemmas to bound the vertex cover size in terms of the extra cost. First, let us bound the vertex cover size in terms of $|L|$ and $|M|$.

Lemma 61. *Any non-star graph F has a vertex cover of size at most* $(|M| + |L| - 1)$.

Proof. Let G_{ML} denote the graph spanned by the edge set $M \cup L$. Note that there are no odd cycles in G_{ML}; otherwise there would be two adjacent edges in the cycle that would belong to the same matching set L or M. Thus, G_{ML} is a bipartite graph. There is a well-known result that says that in a bipartite graph, the size of a maximum matching is equal to the size of a vertex cover [31]. Therefore, G_{ML} has a vertex cover of size exactly $|M|$. Let S' denote a vertex cover of G_{ML}.

Now, we give an incremental construction of a vertex cover of the entire graph G. Let this vertex cover be denoted by S. Initially, we add all vertices of S' to S. Therefore, at this stage, S covers all edges in L and M which means that for every edge in L, at least one of its endpoints must belong to S. Now, we include its other endpoint in S as well and we do this for all edges in L. We observe that S now covers all edges in F' since L is a maximum matching of F', and all edges in F' are incident on L. Therefore, S covers all edges in G, and has a size of $|M| + |L|$.

Our main goal is to obtain a vertex cover of size $|M| + |L| - 1$. We again give an incremental construction and let $\mathcal{S}$ denote this incrementally constructed vertex cover. Initially, $\mathcal{S}$ is empty. Let us color the edges of the graph. We color the edges in M with red color, L with green color, and $E(F'')$ with blue color. Note that non-red edges are the edges of the graph F'. Now,

for every edge in L except one, we add both its endpoints in $\mathcal{S}$. Let $e' \equiv (u', v') \in L$ be the remaining edge of L. Now, we remove all the edges of F covered by $\mathcal{S}$. Let the resulting graph be $G_{\mathcal{S}}$. $G_{\mathcal{S}}$ contains some red edges, some blue edges, and exactly one green edge e'. Also, note that all non-red edges in $G_{\mathcal{S}}$ form a star graph. This is because, if they form a non-star graph, it would have a matching of size at least two and this matching together with the removed green edges form a matching of F' of size $\geq |L| + 1$. This contradicts with the fact that that F' has a maximum matching of size $|L|$. Therefore, non-red edges of $G_{\mathcal{S}}$ form a star graph. Now, let us construct a vertex cover of $G_{\mathcal{S}}$. Let R be the set of red edges in $G_{\mathcal{S}}$. Let NR be the set of non-red edges in $G_{\mathcal{S}}$. Further, assume that $NR \coloneqq \{(u, v_1), (u, v_1), \ldots, (u, v_t)\}$, i.e., all non-red edges are incident on a common vertex u. We consider three different cases depending on the number of red edges in R. For each of these cases, we construct a vertex cover for $G_{\mathcal{S}}$.

1. Case 1: $|R| \leq |M| - |L|$.

   Since NR forms a star graph, we cover it using a single vertex u. For every edge in R, we pick one vertex per edge in the vertex cover. Thus, all edges of $G_{\mathcal{S}}$ are covered. So, the size of the entire vertex cover of F is $(|M| - |L|) + 1 + 2 \cdot (|L| - 1) = |M| + |L| - 1$.

2. Case 2: $|R| \geq |M| - |L| + 2$.

   In this case, R and the removed green edges form a matching of size $|M| + 1$ which contradicts with the fact that G has the maximum matching of size $|M|$. Therefore, we can rule out this case.

3. Case 3: $|R| = |M| - |L| + 1$

   Here, we claim that every non-red edge in NR must be incident on some red edge in R. For the sake of contradiction, suppose this is not true and there is a non-red $e_i \in NR$ that is not incident on any of the red edges in R. It is easy to see that $\{e_i\} \cup R \cup L \setminus \{e'\}$ forms a matching of size $|M| + 1$. It contradicts that F has a matching of size $|M|$. Therefore, each non-red edge in NR must be incident on some red edge in R. Moreover, note that

no two edges in NR can be incident on the same red edge $(r_1, r_2) \in R$. Otherwise, it would form a triangle – (u, r_1, r_2), which is not allowed. Now, for every edge in R, we pick exactly one of its endpoints. This is the endpoint that it shares with some non-red edge in NR if one exists; otherwise an arbitrary endpoint is picked. Thus, we cover the edges in G_S using only $|R|$ vertices. So, the size of the vertex cover of the entire graph F in this case is $|R| + 2|L| - 2 = |M| + |L| - 1$.

This completes the proof of the lemma. □

Now, we bound the extra cost of a graph in terms of $|M|$ and $|L|$. There are some special graph instances for which we do the analysis separately. For the following lemma, we assume that $|L| \geq 3$ and F'' is a non-star non-bridge graph.

Lemma 62. *Let $|L| \geq 3$, and F'' be a non-star non-bridge graph. Then the extra cost of F is at least $(\sqrt{2} - 1) \cdot (\,|M| + |L|\,) - 1.06$.*

Proof. We decompose F into three subgraphs: G_M, G_L, and F''. It gives the following bound on the optimal cost of F.

$$\text{cost}^*(F) \geq \text{cost}^*(G_M) + \text{cost}^*(G_L) + \text{cost}^*(F'') \tag{6.4}$$

We already know the bounds on the optimal costs of G_M, G_L and F''. That is,

- $\text{cost}^*(G_M) \overset{\text{Corollary 28}}{\geq} \sqrt{2} \cdot \sqrt{|M|\,(|M|-1)} \overset{\text{(Lemma 44, } |M| \geq 3)}{\geq} \sqrt{2} \cdot \big(|M| - (3 - \sqrt{6})\big) \geq \sqrt{2} \cdot |M| - 0.78$.
- $\text{cost}^*(G_L) \overset{\text{Corollary 28}}{=} \sqrt{2} \cdot \sqrt{|L|\,(|L|-1)} \overset{\text{(Lemma 44, } |L| \geq 3)}{\geq} \sqrt{2} \cdot \big(|L| - (3 - \sqrt{6})\big) \geq \sqrt{2} \cdot |L| - 0.78$.

- $\text{cost}^*(F'') \overset{\text{Lemma 58}}{\geq} |F''|$.

Substituting the above values in equation 6.4, we obtain the following inequality:

$$\begin{aligned}
\text{cost}^*(F) &\geq |M| + |L| + |F''| + (\sqrt{2}-1)\cdot(|M|+|L|) - 1.56 \\
&= |F| + (\sqrt{2}-1)\cdot(|M|+|L|) - 1.56 \\
&> \sqrt{|F|(|F|-1)} + 0.5 + (\sqrt{2}-1)\cdot(|M|+|L|) - 1.56 \quad \text{(using Lemma 44)} \\
&= \sqrt{|F|(|F|-1)} + (\sqrt{2}-1)\cdot(|M|+|L|) - 1.06
\end{aligned}$$

This completes the proof of the lemma. □

Note that in Lemma 61, we bound the vertex cover size in terms of $|M|$ and $|L|$. Then in Lemma 62, we bound the extra cost in terms of $|M| + |L|$. Now, we put these two results together and obtain a relation between the extra cost and vertex cover size.

Corollary 32. *Let $|L| \geq 3$ and F'' is a non-star non-bridge graph. Then F has a vertex cover of size at most $1.6 + \left(\sqrt{2}+1\right)\delta(F)$.*

Proof. The proof follows from the following sequence of inequalities:
$1.6+\left(\sqrt{2}+1\right)\delta(F) \overset{\text{Lemma 62}}{\geq} |M|+|L|+1.6-\left(\sqrt{2}+1\right)(1.06) > |M|+|L|-1 \overset{\text{Lemma 61}}{=} |VC(F)|$.
□

There are some special graph instances for which either Lemma 61 gives a weak bound on the vertex cover size or Lemma 62 gives a weak bound the extra cost of the graph. This would give an overall weak relation between the vertex cover size and extra cost of the instances. Therefore, we analyse such instances separately. We divide the remaining instances into the following five categories.

1. $|L| = 0$: In this case, we show $|VC(F)| \leq (\sqrt{2}+1)\delta(F) + 0.551$.
2. $|L| = 1$: In this case, we show $|VC(F)| \leq (\sqrt{2}+1)\delta(F) + \mathbf{1.8}$.
3. $|L| = 2$ and F' is a bridge graph: In this case, we show $|VC(F)| \leq (\sqrt{2}+1)\delta(F) + 1.53$.
4. $|L| = 2$ and F' is a non-bridge graph: In this case, we show $|VC(F)| \leq (\sqrt{2}+1)\delta(F) + 1.68$.
5. $|L| \geq 3$ and F'' is a bridge graph: In this case, we show $|VC(F)| \leq (\sqrt{2}+1)\delta(F) + 1.4$.

We analyse these instance one by one. Note that the overall technique remains the same. That is, we first bound the vertex cover size in terms of $|M|$ and $|L|$. Then, we obtain a lower bound on the extra cost in terms of $|M|$ and $|L|$. And, finally we state a corollary (similar to the corollary above) combining these two results. Also, note that for all the following cases we will consider $|M| \geq 3$ since we have already dealt with the case $|M| = 2$ in Section 6.6.2.

6.6.3.1 Case: $|\mathbf{L}| = \mathbf{0}$

The following lemma is trivial.

Lemma 63 (Vertex Cover)**.** *If $|L| = 0$, F has a vertex cover of size exactly $|M|$.*

Lemma 64 (Extra Cost)**.** *If $|L| = 0$, the extra cost of F is exactly $\left(\sqrt{2}-1\right) \cdot \sqrt{|M|\left(|M|-1\right)}$.*

Proof. The proof simply follows from Corollary 28. □

Corollary 33. *If $|L| = 0$, F has a vertex of size at most $0.551 + \left(\sqrt{2}+1\right)\delta(F)$.*

Proof. The proof follows from the following series of inequalities:
$0.551 + \left(\sqrt{2}+1\right)\delta(F) \overset{\text{Lemma 64}}{=} 0.551 + \sqrt{|M|\left(|M|-1\right)} \overset{\text{(Lemma 44, } |M| \geq 3\text{)}}{\geq} 0.551 + |M| - (3-\sqrt{6}) > |M| \overset{\text{Lemma 63}}{=} |VC(F)|$. □

6.6.3.2 Case: $|\mathbf{L}| = 1$

Note that the condition $|L| = 1$ is equivalent to F' being a star graph.

Lemma 65 (Vertex Cover)**.** *If $|L| = 1$, F has a vertex cover of size exactly $|M|$*

Proof. The proof follows from Lemma 61 and substituting $|L| = 1$. □

Lemma 66 (Extra Cost)**.** *If $|L| = 1$, the extra cost of F is at least $\left(\sqrt{2} - 1\right)(|M|) - 0.743$*

Proof. Let e be some edge in $E(F')$. The edge e must incident on some edge of M; otherwise M would not be a maximum matching. Furthermore, e can only be incident on at most two edges of M. Let us define the edges l_1 and l_2 in the graph depending on the orientation of e in the graph.

- If e is incident on two edges of M, then l_1 and l_2 are defined as the corresponding incident edges in M.
- If e is incident on only one edge of M, then $l_1 \in M$ is defined as the incident edge and l_2 is defined as any other edge in M.

Let $M' \coloneqq (M \setminus \{\ell_1, \ell_2\}) \cup \{e\}$ and $L' = E(F') \cup \{\ell_1, \ell_2\} \setminus \{e\}$. Given this, note that M' forms a matching of size $(|M| - 1)$ and L' spans a graph of type A_n for $n \geq 1$. Let $G_{M'}$ denote the graph spanned by M', and $G_{L'}$ denote the graph spanned by L'. We decompose F into these two subgraphs, i.e., $G_{M'}$ and $G_{L'}$. It gives the following bound on the optimal cost of F.

$$\text{cost}^*(F) \geq \text{cost}^*\left(G_{M'}\right) + \text{cost}^*\left(G_{L'}\right) \tag{6.5}$$

We already know the bounds on the optimal costs of $G_{M'}$ and $G_{L'}$. That is,

- $\text{cost}^*(G_{M'}) \overset{\text{Corollary 28}}{\geq} \sqrt{2} \cdot \sqrt{|M'|\left(|M'| - 1\right)} \overset{\text{(Lemma 44, } |M'| \geq 2\text{)}}{\geq} \sqrt{2} \cdot \left(|M'| - (2 - \sqrt{2})\right)$.

- $\text{cost}^*(G_{L'}) \overset{\text{(Lemma 55 statement 1)}}{\geq} |L'|$.

We substitute the above values in Equation (6.5). This gives the following inequality:

$$\begin{aligned}
\text{cost}^*(F) &\geq |M'| + |L'| + \left(\sqrt{2}-1\right)|M'| + 2 - 2\sqrt{2} \\
&= |M| + |F'| + \left(\sqrt{2}-1\right)|M| + 3 - 3\sqrt{2} \\
&\quad \text{(substituting } |M'| = |M| - 1 \text{ and } |L'| = |F'| + 1\text{)} \\
&= |F| + \left(\sqrt{2}-1\right)|M| + 3 - 3\sqrt{2} && (\because |F| = |M| + |F'|) \\
&> \sqrt{|F|(|F|-1)} + 0.5 + \left(\sqrt{2}-1\right)|M| + 3 - 3\sqrt{2} && \text{(using Lemma 44)} \\
&> \sqrt{|F|(|F|-1)} + \left(\sqrt{2}-1\right)|M| - 0.743
\end{aligned}$$

This completes the proof of the lemma. □

Corollary 34. *If $|L| = 1$, then F has a vertex cover of size at most $1.8 + \left(\sqrt{2}+1\right)\delta(F)$.*

Proof. The proof follows from the following sequence of inequalities:

$$1.8 + \left(\sqrt{2}+1\right)\delta(F) \overset{\text{Lemma 66}}{\geq} |M| + 1.8 - \left(\sqrt{2}+1\right)(0.743) > |M| \overset{\text{Lemma 65}}{=} |VC(F)|.$$

□

6.6.3.3 Case: $|\mathbf{L}| = 2$ and $\mathbf{F'}$ is Bridge Graph

Since F' is a bridge graph, $|L| = 2$. For this case, Lemma 61 gives a vertex cover of size at most $|M| + 1$. However, we show a stronger bound than this in the following lemma.

Lemma 67 (Vetex Cover)**.** *If F' is a bridge graph $L_{p,q}$ for some $p, q \geq 1$, then F has a vertex cover of size $|M|$.*

Proof. Let $b \equiv (u, v)$ be the bridge edge of $L_{p,q}$. Suppose b be incident on an edge $e \in M$.

Without loss of generality, we can assume that u is the common endpoint of e and b. Let us pick the vertex u in the vertex cover and remove the edges covered by it. Let G' denote the resulting graph. Further, let M' denote a maximum matching of G'. Now, we claim that $|M'|$ = $|M| - 1$. For the sake of contradiction, assume that $|M'| \geq |M|$. Then the edge b and matching set M' would together form a matching of size $|M| + 1$ and this would contradict that F has a maximum matching M of size $|M|$. Now, suppose we choose $M' \equiv M \setminus \{e\}$ as the maximum matching of G' and let L' be the second maximum matching after M'. If we remove the edges of M' from G', the remaining graph would be a star graph. Therefore, the size of second maximum matching L' is exactly one. Now, using Lemma 61, we can cover G' using $|M'| + |L'| - 1$ vertices. Thus the vertex cover (including the vertex u) of the entire graph F has a size at most $|M'| + |L'| = |M|$. This proves the lemma. □

Lemma 68 (Extra Cost)**.** *If F' is a bridge graph $L_{p,q}$ for some $p, q \geq 1$, then the extra cost of F is at least $\left(\sqrt{2} - 1\right) \cdot |M| - 0.28$.*

Proof. We decompose F into two subgraphs: G_M and F'. It gives the following bound on the optimal cost of F.

$$\text{cost}^*(F) \geq \text{cost}^*(G_M) + \text{cost}^*(F') \tag{6.6}$$

We already know the bounds on the optimal costs of G_M and F'. That is,

- $\text{cost}^*(G_M) \overset{\text{Corollary 28}}{\geq} \sqrt{2} \cdot \sqrt{|M|\,(|M| - 1)} \overset{\text{(Lemma 44, } |M| \geq 3)}{\geq} \sqrt{2} \cdot \left(|M| - (3 - \sqrt{6})\right) \geq \sqrt{2} \cdot |M| - 0.78$.

- $\text{cost}^*(F') \overset{\text{Lemma 57}}{\geq} |F'| - 0.342$.

We substitute the above values in Equation (6.6). It gives the following inequality:

$$\begin{aligned}
\text{cost}^*(F) &\geq |F'| + |M| + \left(\sqrt{2}-1\right)|M| - 1.122 \\
&= |F| + \left(\sqrt{2}-1\right)|M| - 1.122 \\
&> \sqrt{|F|(|F|-1)} + 0.5 + \left(\sqrt{2}-1\right)|M| - 1.122 \qquad \text{(using Lemma 44)} \\
&> \sqrt{|F|(|F|-1)} + \left(\sqrt{2}-1\right)|M| - 0.63
\end{aligned}$$

This completes the proof. □

Corollary 35. *If F' is a bridge graph $L_{p,q}$ for some $p, q \geq 1$, then F has a vertex cover of size at most $1.53 + \left(\sqrt{2}+1\right)\delta(F)$.*

Proof. The proof follows from the following sequence of inequalities:
$1.53 + \left(\sqrt{2}+1\right)\delta(F) \overset{\text{Lemma 68}}{\geq} |M| + 1.53 - \left(\sqrt{2}+1\right)(0.63) > |M| \overset{\text{Lemma 67}}{=} |VC(F)|$. □

6.6.3.4 Case: $|L| = 2$ and F' is Non-Bridge Graph

Lemma 69 (Vertex Cover)**.** *If $|L| = 2$ and F' is a non-bridge graph, then F has a vertex cover of size at most $|M| + 1$.*

Proof. The proof simply follows from Lemma 61 for $|L| = 2$. □

Lemma 70 (Extra Cost)**.** *If $|L| = 2$ and F' is a non-bridge graph, the extra cost of F is at least $\left(\sqrt{2}-1\right) \cdot |M| + 0.5$*

Proof. We decompose F into two subgraphs: G_M and F'. It gives the following bound on the optimal cost of F.

$$\text{cost}^*(F) \geq \text{cost}^*(G_M) + \text{cost}^*(F') \tag{6.7}$$

We already know the bounds on the optimal costs of G_M and F'. That is,

- $\text{cost}^*(G_M) \overset{\text{Corollary 28}}{\geq} \sqrt{2} \cdot \sqrt{|M|\,(|M|-1)} \overset{\text{(Lemma 44, } |M| \geq 3)}{\geq} \sqrt{2} \cdot \left(|M| - (3-\sqrt{6})\right) \geq \sqrt{2} \cdot |M| - 0.78.$
- $\text{cost}^*(F') \overset{\text{Lemma 58}}{\geq} |F'|.$

We substitute the above values in Equation (6.7). It gives the following inequality:

$$\begin{aligned}
\text{cost}^*(F) &\geq |F'| + |M| + \left(\sqrt{2}-1\right)|M| - 0.78 \\
&= |F| + \left(\sqrt{2}-1\right)|M| - 0.78 && (\because |F| = |F'| + |M|) \\
&> \sqrt{|F|(|F|-1)} + 0.5 + \left(\sqrt{2}-1\right)|M| - 0.78 && \text{(using Lemma 44)} \\
&= \sqrt{|F|(|F|-1)} + \left(\sqrt{2}-1\right)|M| - 0.28
\end{aligned}$$

This completes the proof. □

Corollary 36. *If F' is a non-star non-bridge graph, then F has a vertex cover of size at most $1.68 + \left(\sqrt{2}+1\right)\delta(F)$.*

Proof. The proof follows from the following sequence of inequalities:
$1.68 + \left(\sqrt{2}+1\right)\delta(F) \overset{\text{Lemma 70}}{\geq} |M| + 1.68 - \left(\sqrt{2}+1\right)(0.28) > |M| + 1 \overset{\text{Lemma 69}}{=} |VC(F)|.$ □

6.6.3.5 Case: $|L| \geq 3$ and F'' is Bridge Graph

Since $|L| \geq 3$, Lemma 61 gives a vertex cover of size at most $|M| + |L| - 1$, which is at least $|M| + 2$. However, we can obtain a stronger bound than this if F'' is a bridge graph as shown in the following lemma.

Lemma 71. *If $|L| \geq 3$ and F'' is a bridge graph $L_{p,q}$ for some $p, q \geq 1$, then F has a vertex cover of size at most $|M| + 1$.*

Proof. We will incrementally construct a vertex cover S of size $|M| + 1$. Initially, S is empty, i.e., $S = \emptyset$. Let $b \equiv (u, v)$ be the bridge edge of $L_{p,q}$. We will add both vertices u and v to the set S, so that it covers all edges in F''. Now, we remove all the edges in the graph that are covered by u and v. Let M' and L' be the remaining sets corresponding to M and L, respectively. Let G' be the graph spanned by the edge set $M' \cup L'$. Now, observe that G' does not contain any odd cycles; otherwise there would be two adjacent edges in the cycle that would belong to the same set M' or L'. Moreover, G' has a maximum matching of size at most $|M| - 1$. This is because, the edge b is vertex-disjoint from every edge of G' and if G' has a matching of size at least $|M|$, then this matching together with b form a matching of size $|M| + 1$. This contradicts the fact that F has the maximum matching of size $|M|$. Since G' is bipartite and has a matching of size at most $|M| - 1$, it admits a vertex cover of size $|M| - 1$ (using the *Kőnig's Theorem* [31]). Thus, the vertex cover (including the vertices u and v) of the entire graph has a size at most $|M| + 1$. This completes the proof of the lemma. □

Lemma 72. *If $|L| \geq 3$ and F'' is a bridge graph $L_{p,q}$ for some $p, q \geq 1$, then the extra cost of F is at least $(\sqrt{2} - 1) \cdot (|M| + |L|) - 1.41$.*

Proof. We decompose F into three subgraphs: G_M, G_L, and F''. Then, it gives the following bound on the optimal cost of F.

$$\text{cost}^*(F) \geq \text{cost}^*(G_M) + \text{cost}^*(G_L) + \text{cost}^*(F'') \tag{6.8}$$

We already know the bounds on the optimal costs of G_M, G_L and F''. That is,

- $\text{cost}^*(G_M) \overset{\text{Corollary 28}}{\geq} \sqrt{2} \cdot \sqrt{|M|\,(|M| - 1)} \overset{\text{(Lemma 44, } |M| \geq 3)}{\geq} \sqrt{2} \cdot \big(|M| - (3 - \sqrt{6})\big) \geq \sqrt{2} \cdot |M| - 0.78.$
- $\text{cost}^*(G_L) \overset{\text{(using Corollary 28)}}{=} \sqrt{2} \cdot \sqrt{|L|\,(|L| - 1)} \overset{\text{(Lemma 44, } |L| \geq 3)}{\geq} \sqrt{2} \cdot \big(|L| - (3 - \sqrt{6})\big) \geq \sqrt{2} \cdot |L| - 0.78.$

- $\text{cost}^*(F'') \overset{\text{Lemma 57}}{\geq} |F''| - 0.342.$

We substitute the above values in Equation (6.8). It gives the following inequality:

$$\begin{aligned}
\text{cost}^*(F) &\geq |M| + |L| + |F''| + (\sqrt{2}-1)\cdot(|M|+|L|) - 1.902 \\
&= |F| + (\sqrt{2}-1)\cdot(|M|+|L|) - 1.902 \\
&> \sqrt{|F|(|F|-1)} + 0.5 + (\sqrt{2}-1)\cdot(|M|+|L|) - 1.902 \quad \text{(using Lemma 44)} \\
&> \sqrt{|F|(|F|-1)} + (\sqrt{2}-1)\cdot(|M|+|L|) - 1.402
\end{aligned}$$

This completes the proof. □

Corollary 37. *If $|L| \geq 3$ and F'' is a bridge graph $L_{p,q}$ for some $p, q \geq 1$, then F has a vertex cover of size at most $1.4 + \left(\sqrt{2}+1\right)\delta(F)$.*

Proof. The proof follows from the following sequence of inequalities:
$1.4 + \left(\sqrt{2}+1\right)\delta(F) \overset{\text{Lemma 72}}{\geq} |M| + |L| + 1.4 - \left(\sqrt{2}+1\right)(1.402) \overset{(|L| \geq 3)}{>} |M| + 1 \overset{\text{Lemma 71}}{=} |VC(F)|.$

□

This completes the analysis for all graph instances.

6.7 Bi-criteria Hardness of Approximation

In the previous section, we showed that the k-median problem cannot be approximated to any factor smaller than $(1+\varepsilon)$, where ε is some positive constant. The next step in the *beyond worst-case* discussion is to study the bi-criteria approximation algorithms. That is, we allow the algorithm to choose more than k centers and analyse whether it produces a solution that is close to the optimal solution with respect to k centers? Since the algorithm is allowed to output more than k centers we can hope to get a better approximate solution. An interesting question in this regard would be: *Does there exist a PTAS (polynomial time approximation scheme) for*

the k-median/k-means problem when the algorithm is allowed to choose βk centers for some constant $\beta > 1$? In other words, is there an $(1+\varepsilon, \beta)$-approximation algorithm? Note that here we compare the cost of βk centers with the optimal cost with respect to k centers. See Section 6.1 for the definition of (α, β) bi-criteria approximation algorithms.

In this section, we show that even with βk centers, the k-means/k-median problems cannot be approximated within any factor smaller than $(1+\varepsilon')$, for some constant $\varepsilon' > 0$.The following theorem state this result formally.

Theorem 68 (Bi-criteria Hardness: k-Median)**.** *For any constant $1 < \beta < 1.015$, there exists a constant $\varepsilon > 0$ such that there is no $(1+\varepsilon, \beta)$-approximation algorithm for the Euclidean k-median problem assuming the Unique Games Conjecture.*

Theorem 69 (Bi-criteria Hardness: k-Means)**.** *For any constant $1 < \beta < 1.28$, there exists a constant $\varepsilon > 0$ such that there is no $(1+\varepsilon, \beta)$-approximation algorithm for the Euclidean k-means problem assuming the Unique Games Conjecture. Moreover, the same result holds for any $1 < \beta < 1.1$ under the assumption that $\mathsf{P} \neq \mathsf{NP}$.*

First, let us prove the bi-criteria inapproximability result for the k-median problem.

6.7.1 Bi-criteria inapproximability: k-*Median*

In this subsection, we give a proof of Theorem 68. Let us define a few notations. Suppose $\mathcal{I} = (C, k)$ be some k-median instance. Then, OPT(C, k) denote the optimal k-median cost of C. Similarly, OPT$(C, \beta k)$ denote the optimal βk-median cost of C (or the optimal cost of C with βk centers). We use the same reduction as we used in the previous section for showing the hardness of approximation of the k-median problem. Based on the reduction, we establish the following theorem.

Theorem 70. *There is an efficient reduction from Vertex Cover on bounded degree triangle-free graphs G (with m edges) to Euclidean k-median instances $\mathcal{I} = (C, k)$ that satisfies the*

following properties:

1. *If G has a vertex cover of size k, then* $\text{OPT}(C, k) \leq m - k/2$
2. *For any constant $1 < \beta < 1.015$, there exists constants $\varepsilon, \delta > 0$ such that if G has no vertex cover of size $\leq (2 - \varepsilon) \cdot k$, then* $\text{OPT}(C, \beta k) \geq m - k/2 + \delta k$.

Proof. Since the reduction is the same as we discussed in Section 6.3 and 6.5, we keep all notations the same as before. Also, note that Property 1 in this theorem is the same as Property 1 of Theorem 67. Therefore, the proof is also the same as we did in Section 6.5.1. Now, we directly move to the proof of Property 2.

The proof is almost the same as we gave in Section 6.5.2. However, it has some minor differences since we consider the optimal cost with respect to βk centers instead of k centers. Now, we prove the following contrapositive statement: "For any constants $1 < \beta < 1.015$ and $\varepsilon > 0$, there exists constants $\varepsilon, \delta > 0$ such that if $\text{OPT}(C, \beta k) < (m - k/2 + \delta k)$ then G has a vertex cover of size at most $(2 - \varepsilon)k$". Let $\mathcal{C}$ denote an optimal clustering of C with βk centers. We classify its optimal clusters into two categories: (1) *star* and (2) *non-star*. Further, we sub-classify the star clusters into the following two sub-categories:

(a) Clusters composed of exactly one edge. Let these clusters be: $P_1, P_2, \ldots, P_{t_1}$.

(b) Clusters composed of at least two edges. Let these clusters be: $S_1, S_2, \ldots, S_{t_2}$.

Similarly, we sub-classify the non-star clusters into the following two sub-categories:

(i) Clusters with a maximum matching of size two. Let these clusters be: $W_1, W_2, \ldots, W_{t_3}$

(ii) Clusters with a maximum matching of size at least three. Let these clusters be: $Y_1, Y_2, \ldots, Y_{t_4}$

Note that $t_1+t_2+t_3+t_4$ equals βk. Suppose, we first compute a vertex cover of all the clusters except the single edge clusters: $P_1, \ldots, P_{t_1}$. Let that vertex cover be VC'. Now, some vertices in VC' might also cover the edges in $P_1, \ldots, P_{t_1}$. Suppose there are t'_1 single edge clusters that remain uncovered by VC'. Without loss of generality, we assume that these clusters are $P_1, \ldots, P_{t'_1}$. By Lemma 51, we can cover these clusters with $(\frac{2t'_1}{3} + 8\delta k) \leq (\frac{2t_1}{3} + 8\delta k)$ vertices; otherwise the graph would have a vertex cover of size at most $(2k - \delta k)$, and the proof of Property 2 would be complete. Now, we bound the vertex cover of the entire graph in the following manner.

$$
\begin{aligned}
|VC(G)| &\leq \sum_{i=1}^{t_1} |VC(P_i)| + \sum_{i=1}^{t_2} |VC(S_i)| + \sum_{i=1}^{t_3} |VC(W_i)| + \sum_{i=1}^{t_4} |VC(Y_i)| \\
&\leq \left(\frac{2t_1}{3} + 8\delta k\right) + t_2 + \sum_{i=1}^{t_3} \left(\left(\sqrt{2}+1\right)\delta(W_i) + 1.62\right) + \sum_{i=1}^{t_4} \left(\left(\sqrt{2}+1\right)\delta(Y_i) + 1.8\right), \\
&\qquad\qquad \text{(using Lemmas 49, 50, and 51)} \\
&= (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + \left(\sqrt{2}+1\right)\left(\sum_{i=1}^{t_3}\delta(W_i) + \sum_{i=1}^{t_4}\delta(Y_i)\right)
\end{aligned}
$$

Since the optimal cost $OPT(C, \beta k) = \sum_{j=1}^{\beta k}\sqrt{m_j(m_j-1)} + \sum_{i=1}^{t_3}\delta(W_i) + \sum_{i=1}^{t_4}\delta(Y_i) \leq m - k/2 + \delta k$, we get $\sum_{i=1}^{t_3}\delta(W_i) + \sum_{i=1}^{t_4}\delta(Y_i) \leq m - k/2 + \delta k - \sum_{j=1}^{\beta k}\sqrt{m_j(m_j-1)}$. We substitute this value in the previous equation, and get the following inequality:

$$
\begin{aligned}
|VC(G)| \leq (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + \\
\left(\sqrt{2}+1\right) \cdot \left(m - k/2 - \sum_{j=1}^{\beta k}\sqrt{m_j(m_j-1)} + \delta k\right)
\end{aligned}
$$

Using Lemma 44, we obtain the following inequalities:

1. For P_j, $\sqrt{m(P_j)\,(m(P_j)-1)} \geq m(P_j) - 1$ since $m(P_j) = 1$

2. For S_j, $\sqrt{m(S_j)\,(m(S_j)-1)} \geq m(S_j) - (2-\sqrt{2})$ since $m(S_j) \geq 2$

3. For W_j, $\sqrt{m(W_j)\,(m(W_j)-1)} \geq m(W_j) - (2-\sqrt{2})$ since $m(W_j) \geq 2$

4. For Y_j, $\sqrt{m(Y_j)\,(m(Y_j)-1)} \geq m(Y_j) - (3-\sqrt{6})$ since $m(Y_j) \geq 3$

We substitute these values in the previous equation, and get the following inequality:

$$|VC(G)| \leq (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + \left(\sqrt{2}+1\right)\cdot\Bigg(m - k/2 - \sum_{j=1}^{t_1}(m(P_j)-1) - \sum_{j=1}^{t_2}\left(m(S_j) - (2-\sqrt{2})\right) - \sum_{j=1}^{t_3}\left(m(W_j) - (2-\sqrt{2})\right) - \sum_{j=1}^{t_4}\left(m(Y_j) - (3-\sqrt{6})\right) + \delta k\Bigg)$$

Since $m = \sum_{j=1}^{t_1} m(P_j) + \sum_{j=1}^{t_2} m(S_j) + \sum_{j=1}^{t_3} m(W_j) + \sum_{j=1}^{t_4} m(Y_j)$, we get the following inequality:

$$\begin{aligned}|VC(G)| &\leq (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + \left(\sqrt{2}+1\right)\cdot\Bigg(-k/2 + t_1 + \\ &\qquad + t_2\cdot\left(2-\sqrt{2}\right) + \ t_3\cdot\left(2-\sqrt{2}\right) + t_4\cdot\left(3-\sqrt{6}\right) + \delta k\Bigg) \\ &= (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + \left(\sqrt{2}+1\right)\cdot\Bigg(\frac{(\beta-1)k}{2} - \frac{\beta k}{2} + t_1 + \\ &\qquad + t_2\cdot\left(2-\sqrt{2}\right) + \ t_3\cdot\left(2-\sqrt{2}\right) + t_4\cdot\left(3-\sqrt{6}\right) + \delta k\Bigg)\end{aligned}$$

Now, we substitute $\beta k = t_1 + t_2 + t_3 + t_4$, and obtain the following inequality:

$$\begin{aligned}
|VC(G)| &\leq (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 \\
&\qquad + \left(\sqrt{2}+1\right) \cdot \left(\frac{(\beta-1)k}{2} + \frac{t_1}{2} + \frac{t_2}{10} + \frac{t_3}{10} + \frac{3t_4}{50} + \delta k \right) \\
&= (1.88)t_1 + (1.25)t_2 + (1.87)t_3 + (1.95)t_4 + \left(\sqrt{2}+1\right) \cdot \frac{(\beta-1)k}{2} + \left(\sqrt{2}+9\right)\delta k \\
&< (1.95)\beta k + \left(\sqrt{2}+1\right) \cdot \frac{(\beta-1)k}{2} + \left(\sqrt{2}+9\right)\delta k \qquad \text{(using } t_3 + t_4 + t_1 + t_2 = \beta k\text{)} \\
&< (3.16)\beta k - (1.21)k + \left(\sqrt{2}+9\right)\delta k \\
&\leq (2-\varepsilon)k, \quad \text{for } \beta < 1.015 \text{ and appropriately small constants } \varepsilon, \delta > 0
\end{aligned}$$

This proves Property 2 and it completes the proof of Theorem 70. □

The following corollary states the main bi-criteria inapproximability result for the k-median problem.

Corollary 38. *There exists a constant $\varepsilon' > 0$ such that for any constant $1 < \beta < 1.015$, there is no $(1+\varepsilon', \beta)$-approximation algorithm for the k-median problem assuming the Unique Games Conjecture.*

Proof. In the proof of Corollary 26, we showed that $k \geq \frac{m}{2\Delta}$ for all the hard Vertex Cover instances. Therefore, the second property of Theorem 70, implies that OPT$(C, \beta k) \geq (m - \frac{k}{2}) + \delta k \geq (1 + \frac{\delta}{2\Delta}) \cdot (m - \frac{k}{2})$. Thus, the k-median problem can not be approximated within any factor smaller than $1 + \frac{\delta}{2\Delta} = 1 + \Omega(\varepsilon)$, with βk centers for any $\beta < 1.015$. □

Now, we prove the bi-criteria inapproximability result corresponding to the k-means objective.

6.7.2 Bi-criteria inapproximability: *k-Means*

Here, we again use the same reduction that we used earlier for the k-median problem in Sections 6.3, 6.5, and 6.7.1. Using this, we establish the following theorem.

Theorem 71. *There is an efficient reduction from* Vertex Cover *on bounded degree triangle-free graphs G (with m edges) to Euclidean k-means instances $\mathcal{I} = (C, k)$ that satisfies the following properties:*

1. *If G has a vertex cover of size k, then* $\text{OPT}(C, k) \leq m - k$
2. *For any $1 < \lambda \leq 2$ and $\beta < \frac{2}{7} \cdot \left(\lambda + \frac{5}{2}\right)$, there exists constants $\varepsilon, \delta > 0$ such that if G has no vertex cover of size $\leq (\lambda - \varepsilon) \cdot k$, then* $\text{OPT}(C, \beta k) \geq m - k + \delta k$.

This theorem is simply an extension of the result of Awasthi *et al.* [19] to the bi-criteria setting. Now, let us prove this theorem.

6.7.2.1 Completeness

Note that the proof of completeness is already given in [19]. Therefore, we just describe the main components of the proof for the sake of clarity. To understand the proof, let us define some notations used in [19]. Suppose F is a subgraph of G. For a vertex $v \in V(F)$, let $d_F(v)$ denote the number of edges in F that are incident on v. Note that, the optimal center for 1-means problem is simply the centroid of the point set. Therefore, we can compute the optimal 1-means cost of F. The following lemma states the optimal 1-means cost of F.

Lemma 73 (Claim 4.3 [19])**.** *Let F be a subgraph of G with r edges. Then, the optimal 1-means cost of F is $\sum_v d_F(v) \left(1 - \frac{d_F(v)}{r}\right)$*

The following corollary bounds the optimal 1-means cost of a star cluster. This corollary is implicitly stated in the proof of Claim 4.4 of [19].

Corollary 39. *The optimal* 1*-means cost of a star cluster with* r *edges is* $r - 1$.

Using the above corollary, we give the proof of completeness. Let $V = \{v_1, \ldots, v_k\}$ be a vertex cover of G. Let S_i denote the set of edges covered by v_i. If an edge is covered by two vertices i and j, then we arbitrarily keep the edge either in S_i or S_j. Let m_i denote the number of edges in S_i. We define $\{C(S_1), \ldots, C(S_k)\}$ as a clustering of the point set C. Now, we show that the cost of this clustering is at most $m - k$. Note that each S_i forms a star graph with its edges sharing the common vertex v_i. The following sequence of inequalities bound the optimal k-means cost of C.

$$\text{OPT}(C, k) \leq \sum_{i=1}^{k} \text{cost}^*(S_i) \overset{(Corollary\,39)}{=} \sum_{i=1}^{k} (m(S_i) - 1) = m - k.$$

6.7.2.2 Soundness

For the proof of soundness, we prove the following contrapositive statement: "For any constant $1 < \lambda \leq 2$ and $\beta < \frac{2}{7} \cdot \left(\lambda + \frac{5}{2}\right)$, there exists constants $\varepsilon, \delta > 0$ such that if $OPT(\beta k) \leq (m - k + \delta k)$ then G has a vertex cover of size at most $(\lambda - \varepsilon)k$, for $\varepsilon = \Omega(\delta)$." Let $\mathcal{C}$ denote an optimal clustering of C with βk centers. We classify its optimal clusters into two categories: (1) *star* and (2) *non-star*. Suppose there are t_1 star clusters: $S_1, \ldots, S_{t_1}$, and t_2 non-star clusters: $F_1, F_2, \ldots, F_{t_2}$. Note that $t_1 + t_2$ equals βk. The following lemma bounds the optimal 1-means cost of a non-star cluster.

Lemma 74 (Lemma 4.8 [19])**.** *The optimal* 1*-means cost of any non-star cluster* F *with* m *edges is at least* $m - 1 + \delta(F)$*, where* $\delta(F) \geq \frac{2}{3}$*. Furthermore, there is an edge* $(u, v) \in E(F)$ *such that* $d_F(u) + d_F(v) \geq m + 1 - \delta(F)$.

In the actual statement of the lemma in [19], the authors mentioned a weak bound of $\delta(F) > 1/2$. However, in the proof of their lemma they have shown $\delta(F) > 2/3 > 1/2$. This difference does not matter when we consider inapproximability of the k-means problem. However, this

difference improves the β value in bi-criteria inapproximability of the k-means problem.

Corollary 40 ([19]). *Any non-star cluster F has a vertex cover of size at most* $1 + \frac{5}{2} \cdot \delta(F)$.

Proof. Suppose (u, v) be an edge in F that satisfies the property: $d_F(u)+d_F(v) \geq m+1-\delta(F)$, by Lemma 74. This means that u and v covers at least $m(F) - \delta(F)$ edges of F. We pick u and v in the vertex cover, and for the remaining $\delta(F)$ edges we pick one vertex per edge. Therefore, F has a vertex cover of size at most $2 + \delta(F)$. Since $\delta(F) \geq \frac{2}{3}$, by Lemma 74, we get $2 + \delta(F) \leq 1 + \frac{5}{2} \cdot \delta(F)$. Hence, F has a vertex cover of size at most $1 + \frac{5}{2} \cdot \delta(F)$. This proves the corollary. □

Now, the following sequence of inequalities bound the vertex cover size of the enire graph G.

$$\begin{aligned}
|VC(G)| &\leq \sum_{i=1}^{t_1} |VC(S_i)| + \sum_{i=1}^{t_2} |VC(F_i)| \\
&\leq t_1 + \sum_{i=1}^{t_2} \left(1 + \frac{5}{2} \cdot \delta(F_i)\right) \quad \text{(using Corollary 40)} \\
&= t_1 + t_2 + \frac{5}{2} \cdot \sum_{i=1}^{t_2} \delta(F_i)
\end{aligned}$$

Since the optimal k-means cost $\text{OPT}(C, \beta k) = \sum_{i=1}^{t_1} (m(S_i) - 1) + \sum_{i=1}^{t_2} (m(F_i) - 1 + \delta(F_i)) \leq m - k + \delta k$, and $t_1 + t_2 = \beta k$. Therefore, $\sum_{i=1}^{t_2} \delta(F_i) \leq (\beta - 1)k + \delta k$. On substituting this value in the previous equation, we get the following inequality:

$$\begin{aligned}
|VC(G)| &\leq t_1 + t_2 + \frac{5}{2} \cdot (\beta - 1)k + \frac{5}{2} \cdot \delta k \\
&= \beta k + \frac{5}{2} \cdot (\beta - 1)k + \frac{5}{2} \cdot \delta k, \quad (\because t_1 + t_2 = \beta k) \\
&\leq (\lambda - \varepsilon)k, \quad \text{for } \beta < \frac{2}{7} \cdot \left(\lambda + \frac{5}{2}\right) \text{ and appropriately small constants } \varepsilon, \delta > 0
\end{aligned}$$

This proves the soundness condition and it completes the proof of Theorem 71. Based on this theorem, the following corollary states the main bi-criteria inapproximability result for the k-means problem.

Corollary 41. *For any constant $1 < \beta < 1.28$, there exists a constant $\varepsilon' > 0$ such that there is no $(1 + \varepsilon', \beta)$-approximation algorithm for the k-means problem assuming the Unique Games Conjecture. Moreover, the same result holds for any $1 < \beta < 1.1$ under the assumption that* $\mathsf{P} \neq \mathsf{NP}$.

Proof. Suppose Vertex Cover can not be approximated to any factor smaller than $\lambda - \varepsilon$, for some constant $\varepsilon, \lambda > 0$. In the proof of Corollary 26, we showed that $k \geq \frac{m}{2\Delta}$ for all the hard Vertex Cover instances. In that case, the second property of Theorem 71 implies that $\text{OPT}(C, \beta k) \geq (m - k) + \delta k \geq (1 + \frac{\delta}{2\Delta}) \cdot (m - k)$. Thus, the k-means problem can not be approximated within any factor smaller than $1 + \frac{\delta}{2\Delta} = 1 + \Omega(\varepsilon)$, with βk centers. Now, let us compute the value of β based on the value of λ. We know that $\beta < \frac{2}{7} \cdot \left(\lambda + \frac{5}{2}\right)$. Consider the following two cases:

- By Corollary 25, Vertex Cover is hard to approximate within any factor smaller than $2 - \varepsilon$ on bounded degree triangle-free graphs assuming the Unique Games Conjecture. Hence $\lambda = 2$ and thus $\beta < 1.28$ assuming the Unique Game Conjecture.
- By Theorem 64, Vertex Cover is hard to approximate within any factor smaller than 1.36 on bounded degree triangle-free graphs assuming $\mathsf{P} \neq \mathsf{NP}$. Hence $\lambda = 1.36$ and thus $\beta < 1.1$ assuming $\mathsf{P} \neq \mathsf{NP}$.

This completes the proof of the corollary. □

Chapter 7

Conclusion and Future Work

In this study, we designed FPT time constant-approximation algorithms for a range of constrained clustering problems, namely r-gather, r-capacity, balanced, ℓ-diversity, chromatic, strongly-private, fault-tolerant, and fair clustering problems. We designed these algorithms for the k-supplier, k-center, k-median, and k-means clustering objectives in general metric spaces. Moreover, we extended these algorithms to the outlier versions of the problems.

We used a simple technique to design the algorithms: obtaining a polynomial time bi-criteria approximation algorithm and converting it to an FPT time algorithm. However, we observed that this technique only gives a solution to the soft-assignment version of the problems. For the hard-assignment version, we used distance-based sampling techniques to design the algorithms. Unfortunately, the distance-based sampling technique only works with the k-median and k-means objectives. For the constrained k-supplier and k-center problems, we leave it as an open problem to design an FPT time approximation algorithm for the hard-assignment version of the problem. Moreover, we converted the algorithms that use distance-based sampling techniques to streaming algorithms. Therefore, we obtained constant-pass log-space streaming algorithms for the constrained k-median and k-means problems. We leave it as an open prob-

lem to design constant-pass log-space streaming algorithms for the constrained k-supplier and k-center problems.

We also studied the socially fair clustering problem, which generalizes the k-supplier and k-service problems. We designed an FPT time constant factor approximation algorithm for the problem. Moreover, we showed that the obtained approximation guarantees are tight, assuming FPT $\neq$ W[2]. For this problem, the hard-assignment and soft-assignment versions are equivalent. Therefore, we used a simple technique to design the algorithm: obtaining a polynomial time bi-criteria approximation algorithm and converting it to an FPT time algorithm.

We also studied the constrained k-median and k-means problems in continuous Euclidean space. There already exists FPT time $(1+\varepsilon)$-approximation algorithms for the constrained k-median and k-means problems in continuous Euclidean space. We extend the algorithms to the general distance function $\|.\|^z$, outlier setting, and streaming setting. Lastly, we provided the hardness of approximation result for the Euclidean k-median problem in $O(\log k)$ dimensional space. This solved an open question posed explicitly in the work of Awasthi *et al.* [19]. Furthermore, we gave hardness of approximation results for the bi-criteria versions of the Euclidean k-median and k-means problems. In our future work, we aim to study the Euclidean versions of the k-supplier and k-center problems.

Bibliography

[1] Mohsen Abbasi, Aditya Bhaskara, and Suresh Venkatasubramanian. Fair clustering via equitable group representations. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, page 504–514. Association for Computing Machinery, 2021. ISBN 9781450383097. doi: 10.1145/3442188.3445913. URL https://dl.acm.org/doi/10.1145/3442188.3445913.

[2] Marek Adamczyk, Jaroslaw Byrka, Jan Marcinkowski, Syed M. Meesum, and Michal Wlodarczyk. Constant-factor FPT approximation for capacitated k-median. In *27th Annual European Symposium on Algorithms (ESA)*, volume 144, pages 1:1–1:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. ISBN 978-3-95977-124-5. doi: 10.4230/LIPIcs.ESA.2019.1. URL https://drops.dagstuhl.de/opus/volltexte/2019/11122/.

[3] Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive sampling for k-means clustering. In Irit Dinur, Klaus Jansen, Joseph Naor, and José Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, page 15–28. Springer-Verlag, 2009. ISBN 9783642036842. doi: 10.1007/978-3-642-03685-9_2. URL https://dl.acm.org/doi/10.1007/978-3-642-03685-9_2.

[4] Gagan Aggarwal, Tomas Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. Approximation algorithms for k-anonymity. In *Proceedings of the International Conference on Database Theory (ICDT)*, 2005. URL http://ilpubs.stanford.edu:8090/645/.

[5] Gagan Aggarwal, Rina Panigrahy, Tomás Feder, Dilys Thomas, Krishnaram Kenthapadi, Samir Khuller, and An Zhu. Achieving anonymity via clustering. *ACM Trans. Algorithms*, 6, 2010. ISSN 1549-6325. doi: 10.1145/1798596.1798602. URL https://dl.acm.org/doi/10.1145/1798596.1798602.

[6] Akanksha Agrawal, Tanmay Inamdar, Saket Saurabh, and Jie Xue. Clustering what matters: Optimal

approximation for clustering with outliers. *CoRR*, abs/2212.00696, 2022. doi: 10.48550/arXiv.2212.00696. URL https://doi.org/10.48550/arXiv.2212.00696.

[7] Amir Ahmadi-Javid, Pardis Seyedi, and Siddhartha S. Syam. A survey of healthcare facility location. *Computers & Operations Research*, 79:223–263, 2017. ISSN 0305-0548. doi: 10.1016/j.cor.2016.05.018. URL https://doi.org/10.1016/j.cor.2016.05.018.

[8] S. Ahmadian, A. Norouzi-Fard, O. Svensson, and J. Ward. Better guarantees for k-means and Euclidean k-median by primal-dual algorithms. *SIAM J. Comp.*, 49(4):FOCS17–97–FOCS17–156, 2020. doi: 10.1137/18M1171321. URL https://doi.org/10.1137/18M1171321.

[9] Sara Ahmadian and Chaitanya Swamy. Improved approximation guarantees for lower-bounded facility location. In Thomas Erlebach and Giuseppe Persiano, editors, *Approximation and Online Algorithms*, pages 257–271. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-38016-7. doi: 10.1007/978-3-642-38016-7_21. URL https://doi.org/10.1007/978-3-642-38016-7_21.

[10] Sara Ahmadian and Chaitanya Swamy. Approximation algorithms for clustering problems with lower bounds and outliers. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 55, pages 69:1–69:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. ISBN 978-3-95977-013-2. doi: 10.4230/LIPIcs.ICALP.2016.69. URL https://drops.dagstuhl.de/opus/volltexte/2016/6215/.

[11] Sara Ahmadian, Alessandro Epasto, Ravi Kumar, and Mohammad Mahdian. Clustering without over-representation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 267–275. Association for Computing Machinery, 2019. ISBN 9781450362016. doi: 10.1145/3292500.3330987. URL https://dl.acm.org/doi/10.1145/3292500.3330987.

[12] Nir Ailon, Ragesh Jaiswal, and Claire Monteleoni. Streaming k-means approximation. In *Advances in Neural Information Processing Systems 22*, page 10–18. Curran Associates Inc., 2009. ISBN 9781615679119. doi: 10.5555/2984093.2984095. URL https://dl.acm.org/doi/10.5555/2984093.2984095.

[13] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. NP-hardness of Euclidean sum-of-squares clustering. *Mach. Learn.*, 75(2):245–248, may 2009. ISSN 0885-6125. doi: 10.1007/s10994-009-5103-0. URL https://dl.acm.org/doi/10.1007/s10994-009-5103-0.

[14] Hyung-Chan An, Aditya Bhaskara, Chandra Chekuri, Shalmoli Gupta, Vivek Madan, and Ola Svensson. Centrality of trees for capacitated k-center. *Math. Program.*, 154:29–53, 2015. doi: 10.1007/s10107-014-0857-y. URL https://doi.org/10.1007/s10107-014-0857-y.

[15] Barbara M. Anthony, Vineet Goyal, Anupam Gupta, and Viswanath Nagarajan. A plant location guide for the unsure. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, page 1164–1173. Society for Industrial and Applied Mathematics, 2008. doi: 10.5555/1347082.1347209. URL https://dl.acm.org/doi/10.5555/1347082.1347209.

[16] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, page 1027–1035. Society for Industrial and Applied Mathematics, 2007. ISBN 9780898716245. doi: 10.5555/1283383.1283494. URL https://dl.acm.org/doi/10.5555/1283383.1283494.

[17] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comp.*, page 21–29, 2004. doi: 10.1145/380752.380755. URL https://dl.acm.org/doi/10.1145/380752.380755.

[18] P. Austrin, S. Khot, and M. Safra. Inapproximability of vertex cover and independent set in bounded degree graphs. In *2009 24th Annual IEEE Conference on Computational Complexity (CCC)*, page 74–80. IEEE Computer Society, apr 2009. ISBN 9780769537177. doi: 10.1109/CCC.2009.38. URL https://dl.acm.org/doi/10.1109/CCC.2009.38.

[19] Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The hardness of approximation of Euclidean k-means. In *31st International Symposium on Computational Geometry (SoCG)*, volume 34, pages 754–767. Leibniz International Proceedings in Informatics (LIPIcs), 2015. ISBN 978-3-939897-83-5. doi: 10.4230/LIPIcs.SOCG.2015.754. URL https://drops.dagstuhl.de/opus/volltexte/2015/5117/.

[20] Sayan Bandyapadhyay and Kasturi Varadarajan. On variants of k-means clustering. In *32nd International Symposium on Computational Geometry (SoCG)*, volume 51, pages 14:1–14:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. ISBN 978-3-95977-009-5. doi: 10.4230/LIPIcs.SoCG.2016.14. URL https://drops.dagstuhl.de/opus/volltexte/2016/5906/.

[21] Sayan Bandyapadhyay, Fedor V. Fomin, and Kirill Simonov. On coresets for fair clustering in metric and Euclidean spaces and their applications. In *48th International Colloquium on Automata, Lan-*

guages, and Programming (ICALP), volume 198, pages 23:1–23:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. ISBN 978-3-95977-195-5. doi: 10.4230/LIPIcs.ICALP.2021.23. URL https://drops.dagstuhl.de/opus/volltexte/2021/14092/.

[22] Sayan Bandyapadhyay, Zachary Friggstad, and Ramin Mousavi. Parameterized approximation algorithms for k-center clustering and variants. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, (AAAI)*, pages 3895–3903, 2022. URL https://ojs.aaai.org/index.php/AAAI/article/view/20305.

[23] Nikhil Bansal. The circulation problem. https://www.win.tue.nl/~nikhil/courses/2013/2WO08/scribenotes26febv02.pdf, February 2012.

[24] J. Barilan, G. Kortsarz, and D. Peleg. How to allocate network centers. *J. Algorithms*, 15:385–415, 1993. ISSN 0196-6774. doi: 10.1006/jagm.1993.1047. URL https://doi.org/10.1006/jagm.1993.1047.

[25] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. URL http://www.fairmlbook.org.

[26] Suman Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. Fair algorithms for clustering. In *Advances in Neural Information Processing Systems*. Curran Associates Inc., 2019. doi: 10.5555/3454287.3454733. URL https://dl.acm.org/doi/10.5555/3454287.3454733.

[27] Ioana O. Bercea, Martin Groß, Samir Khuller, Aounon Kumar, Clemens Rösner, Daniel R. Schmidt, and Melanie Schmidt. On the cost of essentially fair clusterings. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 145, pages 1032–1041. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. doi: 10.4230/LIPIcs.APPROX-RANDOM.2019.18. URL https://drops.dagstuhl.de/opus/volltexte/2019/11233/.

[28] Anup Bhattacharya, Ragesh Jaiswal, and Amit Kumar. Faster algorithms for the constrained k-means problem. *Theor. Comp. Sys.*, 62(1):93–115, jan 2018. ISSN 1432-4350. doi: 10.1007/s00224-017-9820-7. URL https://dl.acm.org/doi/abs/10.1007/s00224-017-9820-7.

[29] Anup Bhattacharya, Dishant Goyal, Ragesh Jaiswal, and Amit Kumar. On Sampling Based Algorithms for k-Means. In *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020)*, volume 182, pages 13:1–13:17, Dagstuhl, Germany, 2020. Schloss

Dagstuhl–Leibniz-Zentrum für Informatik. ISBN 978-3-95977-174-0. doi: 10.4230/LIPIcs.FSTTCS.2020.13. URL https://drops.dagstuhl.de/opus/volltexte/2020/13254.

[30] Anup Bhattacharya, Dishant Goyal, and Ragesh Jaiswal. Hardness of Approximation for Euclidean k-Median. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*, volume 207, pages 4:1–4:23, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-207-5. doi: 10.4230/LIPIcs.APPROX/RANDOM.2021.4. URL https://drops.dagstuhl.de/opus/volltexte/2021/14697.

[31] John Adrian Bondy. *Graph Theory With Applications*. Elsevier Science Ltd., 1976. URL https://www.zib.de/groetschel/teaching/WS1314/BondyMurtyGTWA.pdf.

[32] Vladimir Braverman, Adam Meyerson, Rafail Ostrovsky, Alan Roytman, Michael Shindler, and Brian Tagiku. Streaming k-means on well-clusterable data. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, page 26–40. Society for Industrial and Applied Mathematics, 2011. doi: 10.5555/2133036.2133039. URL https://dl.acm.org/doi/10.5555/2133036.2133039.

[33] Mihai Bundefineddoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing (STOC)*, page 250–257. Association for Computing Machinery, 2002. ISBN 1581134959. doi: 10.1145/509907.509947. URL https://dl.acm.org/doi/10.1145/509907.509947.

[34] Jarosław Byrka, Bartosz Rybicki, and Sumedha Uniyal. An approximation algorithm for uniform capacitated k-median problem with $1 + \epsilon$ capacity violation. In *Integer Programming and Combinatorial Optimization (IPCO)*, pages 262–274. Springer International Publishing, 2016. ISBN 978-3-319-33461-5. doi: 10.1007/978-3-319-33461-5_22. URL https://doi.org/10.1007/978-3-319-33461-5_22.

[35] Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median and positive correlation in budgeted optimization. *ACM Trans. Algorithms*, 13(2), mar 2017. ISSN 1549-6325. doi: 10.1145/2981561. URL https://dl.acm.org/doi/10.1145/2981561.

[36] Jaroslaw Byrka, Piotr Skowron, and Krzysztof Sornat. Proportional approval voting, harmonic k-median, and negative association. In *45th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 107, pages 26:1–26:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. ISBN

978-3-95977-076-7. doi: 10.4230/LIPIcs.ICALP.2018.26. URL https://drops.dagstuhl.de/opus/volltexte/2018/9030/.

[37] Deeparnab Chakrabarty, Prachi Goyal, and Ravishankar Krishnaswamy. The non-uniform k-center problem. *ACM Trans. Algorithms*, 16(4), jun 2020. ISSN 1549-6325. doi: 10.1145/3392720. URL https://dl.acm.org/doi/10.1145/3392720.

[38] Ramaswamy Chandrasekaran and Arie Tamir. Open questions concerning Weiszfeld's algorithm for the Fermat-Weber location problem. *Math. Progr.*, 44:293–295, 1989. ISSN 1436-4646. doi: 10.1007/BF01587094. URL https://doi.org/10.1007/BF01587094.

[39] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem (extended abstract). In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing (STOC)*, page 1–10. Association for Computing Machinery, 1999. ISBN 1581130678. doi: 10.1145/301250.301257. URL https://dl.acm.org/doi/10.1145/301250.301257.

[40] Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, page 642–651. Society for Industrial and Applied Mathematics, 2001. ISBN 0898714907. doi: 10.5555/365411.365555. URL https://dl.acm.org/doi/10.5555/365411.365555.

[41] Chandra Chekuri (https://cstheory.stackexchange.com/users/176/chandra-chekuri). Is there any bi-criteria ptas for metric k-median? Theoretical Computer Science Stack Exchange, 2021. URL https://cstheory.stackexchange.com/q/49112. URL:https://cstheory.stackexchange.com/q/49112 (version: 2021-06-15).

[42] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006. ISSN 0022-0000. doi: 10.1016/j.jcss.2006.04.007. URL https://doi.org/10.1016/j.jcss.2006.04.007.

[43] Ke Chen. A constant factor approximation algorithm for k-median clustering with outliers. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, page 826–835. Society for Industrial and Applied Mathematics, 2008. doi: 10.5555/1347082.1347173. URL https://dl.acm.org/doi/10.5555/1347082.1347173.

[44] Ke Chen. On coresets for k-median and k-means clustering in metric and Euclidean spaces and their applications. *SIAM J. Comp.*, 39(3):923–947, 2009. doi: 10.1137/070699007. URL https://doi.org/10.1137/070699007.

[45] Xingyu Chen, Brandon Fain, Liang Lyu, and Kamesh Munagala. Proportionally fair clustering. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pages 1032–1041. PMLR, 2019. URL https://proceedings.mlr.press/v97/chen19d.html.

[46] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. In *Advances in Neural Information Processing Systems*, page 5036–5044, 2017. ISBN 9781510860964. doi: 10.5555/3295222.3295256. URL https://dl.acm.org/doi/10.5555/3295222.3295256.

[47] Eden Chlamtác, Yury Makarychev, and Ali Vakilian. Approximating fair clustering with cascaded norm objectives. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2664–2683, 2022. doi: 10.1137/1.9781611977073.104. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611977073.104.

[48] Alexandra Chouldechova and Aaron Roth. A snapshot of the frontiers of fairness in machine learning. *Commun. ACM*, 63(5):82–89, 2020. ISSN 0001-0782. doi: 10.1145/3376898. URL https://dl.acm.org/doi/10.1145/3376898.

[49] Michael B. Cohen, Yin Tat Lee, Gary Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing (STOC)*, page 9–21. Association for Computing Machinery, 2016. ISBN 9781450341325. doi: 10.1145/2897518.2897647. URL https://dl.acm.org/doi/10.1145/2897518.2897647.

[50] Vincent Cohen-Addad. A fast approximation scheme for low-dimensional k-means. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, page 430–440. Society for Industrial and Applied Mathematics, 2018. ISBN 9781611975031. doi: 10.5555/3174304.3175298. URL https://dl.acm.org/doi/abs/10.5555/3174304.3175298.

[51] Vincent Cohen-Addad and Karthik C.S. Inapproximability of clustering in Lp metrics. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 519–539, 2019. doi: 10.1109/FOCS.2019.00040. URL https://ieeexplore.ieee.org/document/8948668.

[52] Vincent Cohen-Addad and Jason Li. On the fixed-parameter tractability of capacitated clustering. In *46th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132,

pages 41:1–41:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. ISBN 978-3-95977-109-2. doi: 10.4230/LIPIcs.ICALP.2019.41. URL https://drops.dagstuhl.de/opus/volltexte/2019/10617/.

[53] Vincent Cohen-Addad, Anupam Gupta, Amit Kumar, Euiwoong Lee, and Jason Li. Tight FPT approximations for k-median and k-means. In *46th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132, pages 42:1–42:14, 2019. doi: 10.4230/LIPIcs.ICALP.2019.42. URL http://drops.dagstuhl.de/opus/volltexte/2019/10618.

[54] Vincent Cohen-Addad, Philip N. Klein, and Claire Mathieu. Local search yields approximation schemes for k-means and k-median in Euclidean and minor-free metrics. *SIAM J. Comp.*, 48(2):644–667, 2019. doi: 10.1137/17M112717X. URL https://doi.org/10.1137/17M112717X.

[55] Vincent Cohen-Addad, Karthik C. S., and Euiwoong Lee. On approximability of clustering problems without candidate centers. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, page 2635–2648. Society for Industrial and Applied Mathematics, 2021. ISBN 9781611976465. doi: 10.5555/3458064.3458220. URL https://dl.acm.org/doi/abs/10.5555/3458064.3458220.

[56] Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, page 169–182, 2021. ISBN 9781450380539. doi: 10.1145/3406325.3451022. URL https://dl.acm.org/doi/10.1145/3406325.3451022.

[57] Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. Improved coresets and sublinear algorithms for power means in Euclidean spaces. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, (NeurIPS)*, volume 34, pages 21085–21098. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/b035d6563a2adac9f822940c145263ce-Paper.pdf.

[58] Vincent Cohen-Addad, Karthik C. S., and Euiwoong Lee. Johnson coverage hypothesis: Inapproximability of k-means and k-median in ℓ_p-metrics. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1493–1530, 2022. doi: 10.1137/1.9781611977073.63. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611977073.63.

[59] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algo-*

rithms, 3rd Edition. MIT Press, 2009. ISBN 9780262533058. URL https://mitpress.mit.edu/9780262533058/introduction-to-algorithms/.

[60] Graham Cormode and Andrew McGregor. Approximation algorithms for clustering uncertain data. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, page 191–200. Association for Computing Machinery, 2008. ISBN 9781605581521. doi: 10.1145/1376916.1376944. URL https://dl.acm.org/doi/10.1145/1376916.1376944.

[61] M. Cygan, M. Hajiaghayi, and S. Khuller. LP rounding for k-centers with non-uniform hard capacities. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 273–282. IEEE Computer Society, 2012. doi: 10.1109/FOCS.2012.63. URL https://doi.org/10.1109/FOCS.2012.63.

[62] Marek Cygan and Tomasz Kociumaka. Constant factor approximation for capacitated k-center with outliers. In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 25, pages 251–262. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014. ISBN 978-3-939897-65-1. doi: 10.4230/LIPIcs.STACS.2014.251. URL https://drops.dagstuhl.de/opus/volltexte/2014/4462/.

[63] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer Cham, 2015. ISBN 978-3-319-21275-3. doi: 10.1007/978-3-319-21275-3. URL https://link.springer.com/book/10.1007/978-3-319-21275-3.

[64] Sanjoy Dasgupta. The hardness of k-means clustering. Technical Report CS2008-0916, Department of Computer Science and Engineering, University of California San Diego, 2008. URL https://cseweb.ucsd.edu/~dasgupta/papers/kmeans.pdf.

[65] Mark S. Daskin and Latoya K. Dean. *Location of Health Care Facilities*, volume 70, pages 43–76. Springer US, 2004. ISBN 978-1-4020-8066-1. doi: 10.1007/1-4020-8066-2_3. URL https://doi.org/10.1007/1-4020-8066-2_3.

[66] Erik D. Demaine, Fedor V. Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs. *ACM Trans. Algorithms*, 1(1): 33–47, jul 2005. ISSN 1549-6325. doi: 10.1145/1077464.1077468. URL https://dl.acm.org/doi/10.1145/1077464.1077468.

[67] Gökalp Demirci and Shi Li. Constant approximation for capacitated k-median with $(1+\epsilon)$-capacity violation. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 55, pages 73:1–73:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. ISBN 978-3-95977-013-2. doi: 10.4230/LIPIcs.ICALP.2016.73. URL https://doi.org/10.4230/LIPIcs.ICALP.2016.73.

[68] Hu Ding. Balanced k-center clustering when k is a constant. In *Proceedings of the 29th Canadian Conference on Computational Geometry (CCCG)*, pages 179–184, 2017. URL http://2017.cccg.ca/proceedings/Session5B-paper2.pdf.

[69] Hu Ding. Faster balanced clusterings in high dimension. *Theor. Comp. Sci.*, 842:28–40, 2020. ISSN 0304-3975. doi: 10.1016/j.tcs.2020.07.022. URL https://doi.org/10.1016/j.tcs.2020.07.022.

[70] Hu Ding and Jinhui Xu. Solving the chromatic cone clustering problem via minimum spanning sphere. In *Proceedings of the 38th International Colloquim Conference on Automata, Languages and Programming (ICALP)*, page 773–784. Springer-Verlag, 2011. ISBN 9783642220050. doi: 10.5555/2027127.2027209. URL https://dl.acm.org/doi/abs/10.5555/2027127.2027209.

[71] Hu Ding and Jinhui Xu. Chromatic k-mean clustering in high dimensional space. *CoRR*, abs/1204.6699, 2012. URL http://arxiv.org/abs/1204.6699.

[72] Hu Ding and Jinhui Xu. A unified framework for clustering constrained data without locality property. *Algorithmica*, 82(4):808–852, apr 2020. ISSN 0178-4617. doi: 10.1007/s00453-019-00616-2. URL https://dl.acm.org/doi/abs/10.1007/s00453-019-00616-2.

[73] Hu Ding, Lunjia Hu, Lingxiao Huang, and Jian Li. Capacitated center problems with two-sided bounds and outliers. In *Algorithms and Data Structures - 15th International Symposium, WADS 2017, St. John's, NL, Canada, July 31 - August 2, 2017, Proceedings*, volume 10389, pages 325–336. Springer International Publishing, 2017. ISBN 978-3-319-62127-2. doi: 10.1007/978-3-319-62127-2_28. URL https://doi.org/10.1007/978-3-319-62127-2_28.

[74] Rod G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for $W[1]$. *Theor. Comp. Sci.*, 141(1-2):109–131, 1995. ISSN 0304-3975. doi: 10.1016/0304-3975(94)00097-3. URL https://doi.org/10.1016/0304-3975(94)00097-3.

[75] Zvi Drezner and Horst W Hamacher. *Facility location: applications and theory*. Springer Science &

Business Media, 2001. ISBN 978-3-540-21345-1. URL https://link.springer.com/book/9783540421726.

[76] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, 1972. ISSN 0004-5411. doi: 10.1145/321694.321699. URL https://dl.acm.org/doi/10.1145/321694.321699.

[77] Dan Feldman, Morteza Monemizadeh, and Christian Sohler. A PTAS for k-means clustering based on weak coresets. In *Proceedings of the twenty-third annual symposium on Computational geometry (SoCG)*, page 11–18. Association for Computing Machinery, 2007. ISBN 9781595937056. doi: 10.1145/1247069.1247072. URL https://dl.acm.org/doi/10.1145/1247069.1247072.

[78] Andreas Emil Feldmann. Fixed-parameter approximations for k-center problems in low highway dimension graphs. *Algorithmica*, 81:1031–1052, 2019. doi: 10.1007/s00453-018-0455-0. URL https://doi.org/10.1007/s00453-018-0455-0.

[79] Andreas Emil Feldmann and Dániel Marx. The parameterized hardness of the k-center problem in transportation networks. In *16th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2018)*, volume 101, pages 19:1–19:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. ISBN 978-3-95977-068-2. doi: 10.4230/LIPIcs.SWAT.2018.19. URL https://drops.dagstuhl.de/opus/volltexte/2018/8845/.

[80] Qilong Feng, Zhen Zhang, Ziyun Huang, Jinhui Xu, and Jianxin Wang. Improved algorithms for clustering with outliers. In *30th International Symposium on Algorithms and Computation (ISAAC)*, volume 149, pages 61:1–61:12. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. ISBN 978-3-95977-130-6. doi: 10.4230/LIPIcs.ISAAC.2019.61. URL https://drops.dagstuhl.de/opus/volltexte/2019/11557/.

[81] Zachary Friggstad, Kamyar Khodamoradi, Mohsen Rezapour, and Mohammad R. Salavatipour. Approximation schemes for clustering with outliers. *ACM Trans. Algorithms*, 15(2), feb 2019. ISSN 1549-6325. doi: 10.1145/3301446. URL https://dl.acm.org/doi/10.1145/3301446.

[82] Zachary Friggstad, Mohsen Rezapour, and Mohammad R. Salavatipour. Local search yields a PTAS for k-means in doubling metrics. *SIAM J. Comp.*, 48(2):452–480, 2019. doi: 10.1137/17M1127181. URL https://doi.org/10.1137/17M1127181.

[83] Mehrdad Ghadiri, Samira Samadi, and Santosh Vempala. Socially fair k-means clustering. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, page 438–448. Association for Computing Machinery, 2021. ISBN 9781450383097. doi: 10.1145/3442188.3445906. URL https://dl.acm.org/doi/10.1145/3442188.3445906.

[84] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comp. Sci.*, 38: 293–306, 1985. ISSN 0304-3975. doi: 10.1016/0304-3975(85)90224-5. URL https://doi.org/10.1016/0304-3975(85)90224-5.

[85] Dishant Goyal and Ragesh Jaiswal. Tight fpt approximation for socially fair clustering. *Information Processing Letters*, page 106383, 2023. ISSN 0020-0190. doi: https://doi.org/10.1016/j.ipl.2023.106383. URL https://www.sciencedirect.com/science/article/pii/S0020019023000261.

[86] Dishant Goyal and Ragesh Jaiswal. Tight fpt approximation for constrained k-center and k-supplier. *Theoretical Computer Science*, 940:190–208, 2023. ISSN 0304-3975. doi: https://doi.org/10.1016/j.tcs.2022.11.001. URL https://www.sciencedirect.com/science/article/pii/S0304397522006594.

[87] Dishant Goyal, Ragesh Jaiswal, and Amit Kumar. Streaming PTAS for constrained k-means. *CoRR*, abs/1909.07511, 2019. URL http://arxiv.org/abs/1909.07511.

[88] Dishant Goyal, Ragesh Jaiswal, and Amit Kumar. FPT Approximation for Constrained Metric k-Median/Means. In *15th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 180, pages 14:1–14:19. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020. ISBN 978-3-95977-172-6. doi: 10.4230/LIPIcs.IPEC.2020.14. URL https://drops.dagstuhl.de/opus/volltexte/2020/13317.

[89] Fabrizio Grandoni, Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, and Rakesh Venkat. A refined approximation for Euclidean k-means. *Inf. Process. Lett.*, 176(C), 2022. ISSN 0020-0190. doi: 10.1016/j.ipl.2022.106251. URL https://dl.acm.org/doi/10.1016/j.ipl.2022.106251.

[90] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 359–366. IEEE Computer Society, 2000. ISBN 0769508502. doi: 10.5555/795666.796588. URL https://dl.acm.org/doi/10.5555/795666.796588.

[91] Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *J. Algorithms*, 31(1):228–248, 1999. ISSN 0196-6774. doi: 10.1006/jagm.1998.0993. URL https://doi.org/10.1006/jagm.1998.0993.

[92] Mohammadtaghi Hajiaghayi, Wei Hu, Jian Li, Shi Li, and Barna Saha. A constant factor approximation algorithm for fault-tolerant k-median. *ACM Trans. Algorithms*, 12, 2016. ISSN 1549-6325. doi: 10.1145/2854153. URL https://dl.acm.org/doi/10.1145/2854153.

[93] Elfarouk Harb and Ho Shan Lam. KFC: A scalable approximation algorithm for k-center fair clustering. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 33, pages 14509–14519. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/a6d259bfbfa2062843ef543e21d7ec8e-Paper.pdf.

[94] Dorit S. Hochbaum and David B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *J. ACM*, 33(3):533–550, may 1986. ISSN 0004-5411. doi: 10.1145/5925.5933. URL https://doi.org/10.1145/5925.5933.

[95] Tanmay Inamdar and Kasturi Varadarajan. Fault tolerant clustering with outliers. In *Approximation and Online Algorithms*, page 188–201. Springer-Verlag, 2020. ISBN 978-3-030-39478-3. doi: 10.1007/978-3-030-39479-0_13. URL https://dl.acm.org/doi/abs/10.1007/978-3-030-39479-0_13.

[96] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, sep 1999. ISSN 0360-0300. doi: 10.1145/331499.331504. URL https://dl.acm.org/doi/10.1145/331499.331504.

[97] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing (STOC)*, page 731–740, 2002. ISBN 1581134959. doi: 10.1145/509907.510012. URL https://dl.acm.org/doi/10.1145/509907.510012.

[98] Ragesh Jaiswal, Amit Kumar, and Sandeep Sen. A simple D^2-sampling based PTAS for k-means and other clustering problems. *Algorithmica*, 70(1):22–46, sep 2014. ISSN 0178-4617. doi: 10.1007/s00453-013-9833-9. URL https://dl.acm.org/doi/abs/10.1007/s00453-013-9833-9.

[99] Viggo Kann. On the approximability of NP-complete optimization problems. 1992. URL https://www.csc.kth.se/~viggo/papers/phdthesis.pdf.

[100] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. In *Proceedings of the Eighteenth Annual Symposium on Computational Geometry (SoCG)*, page 10–18, 2002. ISBN 1581135041. doi: 10.1145/513400.513402. URL https://dl.acm.org/doi/10.1145/513400.513402.

[101] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103. Springer, 1972. ISBN 978-1-4684-2001-2. doi: 10.1007/978-1-4684-2001-2_9. URL https://doi.org/10.1007/978-1-4684-2001-2_9.

[102] Samir Khuller and Yoram J. Sussmann. The capacitated k-center problem. *SIAM J. Discret. Math.*, 13: 403–418, 2000. ISSN 0895-4801. doi: doi/10.1137/S0895480197329776. URL https://dl.acm.org/doi/10.1137/S0895480197329776.

[103] Samir Khuller, Robert Pless, and Yoram J. Sussmann. Fault tolerant k-center problems. *Theor. Comp. Sci.*, 242:237–245, 2000. ISSN 0304-3975. doi: 10.1016/S0304-3975(98)00222-9. URL https://doi.org/10.1016/S0304-3975(98)00222-9.

[104] Matthäus Kleindessner, Pranjal Awasthi, and Jamie Morgenstern. Fair k-center clustering for data summarization. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 162, pages 669–702. PMLR, 2019. URL https://proceedings.mlr.press/v162/angelidakis22a.html.

[105] J. Krarup and S. Vajda. On Torricelli's geometrical solution to a problem of Fermat. *IMA J. Manag. Math.*, 8(3):215–224, 1997. doi: 10.1093/imaman/8.3.215. URL https://doi.org/10.1093/imaman/8.3.215.

[106] Ravishankar Krishnaswamy, Shi Li, and Sai Sandeep. Constant approximation for k-median and k-means with outliers via iterative rounding. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, page 646–659. Association for Computing Machinery, 2018. ISBN 9781450355599. doi: 10.1145/3188745.3188882. URL https://dl.acm.org/doi/10.1145/3188745.3188882.

[107] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering

problems in any dimensions. *J. ACM*, 57(2):22–46, feb 2010. ISSN 0004-5411. doi: 10.1145/1667053.1667054. URL https://dl.acm.org/doi/abs/10.1145/1667053.1667054.

[108] Christiane Lammersen, Melanie Schmidt, and Christian Sohler. Probabilistic k-median clustering in data streams. *Theor. Comp. Sys.*, 56(1):251–290, jan 2015. ISSN 1432-4350. doi: 10.1007/s00224-014-9539-7. URL https://doi.org/10.1007/s00224-014-9539-7.

[109] Euiwoong Lee, Melanie Schmidt, and John Wright. Improved and simplified inapproximability for k-means. *Inf. Process. Lett.*, 120:40–43, 2017. ISSN 0020-0190. doi: 10.1016/j.ipl.2016.11.009. URL https://doi.org/10.1016/j.ipl.2016.11.009.

[110] Jian Li, Ke Yi, and Qin Zhang. Clustering with diversity. In *Proceedings of the 37th International Colloquium Conference on Automata, Languages and Programming (ICALP)*, pages 188–200, 2010. ISBN 978-3-642-14165-2. doi: 10.1007/978-3-642-14165-2_17. URL https://doi.org/10.1007/978-3-642-14165-2_17.

[111] Shi Li. Approximating capacitated k-median with $(1+\varepsilon)k$ open facilities. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, page 786–796. Society for Industrial and Applied Mathematics, 2016. ISBN 9781611974331. doi: 10.5555/2884435.2884491. URL https://dl.acm.org/doi/10.5555/2884435.2884491.

[112] Shi Li. On uniform capacitated k-median beyond the natural LP relaxation. *ACM Trans. Algorithms*, 13, 2017. ISSN 1549-6325. doi: 10.1145/2983633. URL https://dl.acm.org/doi/abs/10.1145/2983633.

[113] Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing (STOC)*, page 901–910. Association for Computing Machinery, 2013. ISBN 9781450320290. doi: 10.1145/2488608.2488723. URL https://dl.acm.org/doi/10.1145/2488608.2488723.

[114] S. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theor.*, 28(2):129–137, 1982. ISSN 1557-9654. doi: 10.1109/TIT.1982.1056489. URL https://doi.org/10.1109/TIT.1982.1056489.

[115] Mario Lucic, Matthew Faulkner, Andreas Krause, and Dan Feldman. Training gaussian mixture models at scale via coresets. *J. Mach. Learn. Res.*, 18(160):1–25, 2017. URL http://jmlr.org/papers/v18/15-506.html.

[116] Sepideh Mahabadi and Ali Vakilian. Individual fairness for k-clustering. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pages 6586–6596. JMLR.org, 2020. doi: 10.5555/3524938.3525549. URL https://dl.acm.org/doi/abs/10.5555/3524938.3525549.

[117] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is NP-hard. *Theor. Comp. Sci.*, 442:13–21, 2012. ISSN 0304-3975. doi: 10.1016/j.tcs.2010.05.034. URL https://doi.org/10.1016/j.tcs.2010.05.034.

[118] Konstantin Makarychev, Yury Makarychev, Maxim Sviridenko, and Justin Ward. A bi-criteria approximation algorithm for k-means. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 60, pages 14:1–14:20. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. ISBN 978-3-95977-018-7. doi: 10.4230/LIPIcs.APPROX-RANDOM.2016.14. URL https://drops.dagstuhl.de/opus/volltexte/2016/6637/.

[119] Konstantin Makarychev, Yury Makarychev, and Ilya Razenshteyn. Performance of Johnson-Lindenstrauss transform for k-means and k-medians clustering. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, page 1027–1038. Association for Computing Machinery, 2019. ISBN 9781450367059. doi: 10.1145/3313276.3316350. URL https://dl.acm.org/doi/10.1145/3313276.3316350.

[120] Yury Makarychev and Ali Vakilian. Approximation algorithms for socially fair clustering. In *Conference on Learning Theory, (COLT)*, volume 134, pages 3246–3264. PMLR, 2021. URL https://proceedings.mlr.press/v134/makarychev21a.html.

[121] Pasin Manurangsi. Tight running time lower bounds for strong inapproximability of maximum k-coverage, unique set cover and related problems (via t-wise agreement testing theorem). In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, page 62–81, 2020. doi: 10.5555/3381089.3381094. URL https://dl.acm.org/doi/10.5555/3381089.3381094.

[122] Jirı Matoušek. On approximate geometric k-clustering. *Discrete & Computational Geometry*, 24:61–84, 2000. ISSN 1432-0444. doi: 10.1007/s004540010019. URL https://doi.org/10.1007/s004540010019.

[123] Nimrod Megiddo and Kenneth J. Supowit. On the complexity of some common geometric location problems. *SIAM J. Comp.*, 13(1):182–196, 1984. doi: 10.1137/0213014. URL https://doi.org/10.1137/0213014.

[124] P. Milasevic and G. R. Ducharme. Uniqueness of the spatial median. *Ann. Statist.*, 15(3):1332 – 1333, 1987. doi: 10.1214/aos/1176350511. URL https://doi.org/10.1214/aos/1176350511.

[125] Stanislav Minsker. Geometric median and robust estimation in Banach spaces. *Bernoulli*, 21(4):2308–2335, 2015. doi: 10.3150/14-BEJ645. URL https://doi.org/10.3150/14-BEJ645.

[126] Pitu B. Mirchandani and Richard L. Francis. *Discrete Location Theory*. Wiley, 1990. ISBN 978-0-471-89233-5. URL https://www.wiley.com/en-us/exportProduct/pdf/9780471892335.

[127] Clemens Rösner and Melanie Schmidt. Privacy preserving clustering with constraints. In *45th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 107, pages 96:1–96:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. ISBN 978-3-95977-076-7. doi: 10.4230/LIPIcs.ICALP.2018.96. URL https://drops.dagstuhl.de/opus/volltexte/2018/9100/.

[128] Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. *J. ACM*, 66(5), aug 2019. ISSN 0004-5411. doi: 10.1145/3325116. URL https://doi.org/10.1145/3325116.

[129] Zoya Svitkina. Lower-bounded facility location. *ACM Trans. Algorithms*, 6, 2010. ISSN 1549-6325. doi: 10.1145/1824777.1824789. URL https://dl.acm.org/doi/10.1145/1824777.1824789.

[130] Chaitanya Swamy and David B. Shmoys. Fault-tolerant facility location. *ACM Trans. Algorithms*, 4, 2008. ISSN 1549-6325. doi: 10.1145/1383369.1383382. URL https://dl.acm.org/doi/10.1145/1383369.1383382.

[131] Latanya Sweeney. k-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002. ISSN 0218-4885. doi: 10.1142/S0218488502001648. URL https://dl.acm.org/doi/10.1142/S0218488502001648.

[132] Éva Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5:247–255, 1985. ISSN 1439-6912. doi: 10.1007/BF02579369. URL https://doi.org/10.1007/BF02579369.

[133] Andrea Vattani. The hardness of k-means clustering in the plane. Technical report, Department of Computer Science and Engineering, University of California San Diego, 2009. URL https://cseweb.ucsd.edu/~avattani/papers/kmeans_hardness.pdf.

[134] Sundar Vishwanathan. An o(log*n) approximation algorithm for the asymmetric p-center problem. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, page 1–5,

USA, 1996. Society for Industrial and Applied Mathematics. ISBN 0898713668. doi: 10.5555/313852.313861. URL https://dl.acm.org/doi/10.5555/313852.313861.

[135] J S Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, mar 1985. ISSN 0098-3500. doi: 10.1145/3147.3165. URL https://dl.acm.org/doi/10.1145/3147.3165.

[136] Dennis Wei. A constant-factor bi-criteria approximation guarantee for k-means++. In *Advances in Neural Information Processing Systems 29*, page 604–612. Curran Associates Inc., 2016. ISBN 9781510838819. doi: 10.5555/3157096.3157164. URL https://dl.acm.org/doi/10.5555/3157096.3157164.

[137] E. Weiszfeld. Sur le point pour lequel la somme des distances de n points donnes est minimum. *Tohoku Math. J., First Series*, 43:355–386, 1937. ISSN 1881-2015. URL https://www.jstage.jst.go.jp/article/tmj1911/43/0/43_0_355/_article/-char/en.

[138] Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Ann. Data. Sci.*, 2:165–193, 2015. doi: 10.1007/s40745-015-0040-1. URL https://link.springer.com/article/10.1007/s40745-015-0040-1.

[139] Yicheng Xu, Rolf H Möhring, Dachuan Xu, Yong Zhang, and Yifei Zou. A constant FPT approximation algorithm for hard-capacitated k-means. *Optimization and Engineering*, 21:709–722, 2020. ISSN 1573-2924. doi: 10.1007/s11081-020-09503-0. URL https://doi.org/10.1007/s11081-020-09503-0.

[140] Neal Young (https://cstheory.stackexchange.com/users/8237/neal-young). Is there any bi-criteria PTAS for metric k-median? Theoretical Computer Science Stack Exchange, 2021. URL https://cstheory.stackexchange.com/q/49113. URL:https://cstheory.stackexchange.com/q/49113 (version: 2021-06-16).

List of Publications

- Dishant Goyal and Ragesh Jaiswal

  "Tight FPT Approximation for Constrained k-Supplier and k-Center ", In *Theoretical Computer Science*, 2023

- Dishant Goyal and Ragesh Jaiswal

  "Tight FPT Approximation for Socially Fair Clustering ", In *Information Processing Letters*, 2023

- Anup Bhattacharya, Dishant Goyal, Ragesh Jaiswal

  "Hardness of Approximation for Euclidean k-Median ", In *International Conference on Approximation Algorithms or Combinatorial Optimization Problems (APPROX),* 2021

- Dishant Goyal, Ragesh Jaiswal, Amit Kumar

  "FPT Approximation for Constrained Metric k-Median and k-Means ", In *International Symposium on Parameterized and Exact Computation (IPEC),* 2020

- Anup Bhattacharya, Dishant Goyal, Ragesh Jaiswal, Amit Kumar

  "On Sampling Based Algorithms for k-Means ", In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS),* 2020