Mathematical Structures in Computer Science (2022), **32**, pp. 151–163
doi:10.1017/S0960129521000104

CAMBRIDGE UNIVERSITY PRESS

PAPER

An improved primal-dual approximation algorithm for the *k*-means problem with penalties†

Chunying Ren[1], Dachuan Xu[1], Donglei Du[2] and Min Li[3,*]

[1]Department of Operations Research and Information Engineering, Beijing University of Technology, Beijing 100124, P.R. China, [2]Faculty of Management, University of New Brunswick, Fredericton, NB E3B 5A3, Canada and [3]School of Mathematics and Statistics, Shandong Normal University, Jinan 250014, P.R. China
*Corresponding author. Email: liminemily@sdnu.edu.cn

(Received 18 December 2020; revised 15 March 2021; accepted 2 June 2021; first published online 16 August 2021)

Abstract

In the k-means problem with penalties, we are given a data set $\mathcal{D} \subseteq \mathbb{R}^\ell$ of n points where each point $j \in \mathcal{D}$ is associated with a penalty cost p_j and an integer k. The goal is to choose a set $CS \subseteq \mathcal{R}^\ell$ with $|CS| \leq k$ and a penalized subset $\mathcal{D}_p \subseteq \mathcal{D}$ to minimize the sum of the total squared distance from the points in $D \setminus D_p$ to CS and the total penalty cost of points in D_p, namely $\sum_{j \in \mathcal{D} \setminus \mathcal{D}_p} d^2(j, CS) + \sum_{j \in \mathcal{D}_p} p_j$. We employ the primal-dual technique to give a pseudo-polynomial time algorithm with an approximation ratio of $(6.357 + \varepsilon)$ for the k-means problem with penalties, improving the previous best approximation ratio $19.849 + \epsilon$ for this problem given by Feng et al. in Proceedings of FAW (2019).

Keywords: K-means problem with penalties; linear program; approximation algorithm; JV algorithm

1. Introduction

Machine learning is one of the fastest growing branches in the field of artificial intelligence, and within it, clustering problems gain great popularity in the unsupervised learning. Modern civilization has witnessed the extensive applications of the algorithms related to clustering problems in finance, people's livelihood, medical care, transportation, and the Internet, etc.

The k-means problem is the most representative problem in clustering. It has been widely studied in the fields of operations research, management, and computer science (Charikar et al., 2001; Friggstad et al., 2019; Ji et al., 2020; Kanungo et al., 2004; Makarychev et al., 2019; Vazirani, 2001; Williamson and Shmoys, 2011). A standard k-means problem is described as follows. Given an integer k and a data set $\mathcal{D} \subseteq \mathcal{R}^\ell$ of n points, the purpose is to select a central point set CS in $\mathcal{R}^\ell$ such that the size of CS cannot exceed k. The customers in $\mathcal{D}$ are divided into $|CS|$ categories. The connection cost of each cluster is the square of the Euclidean distance between all customers in the category and its center point. Our target is to minimize the connection cost of all categories.

In general, the k-means problem is NP-hard (Aloise et al., 2009; Drineas et al., 2004), which means one cannot obtain an optimal solution in polynomial time unless P = NP. The earliest algorithm for the k-means problem was proposed around 1950, and over the next 70 years, hundreds

†A preliminary version of this paper appeared in Proceedings of the 16th Annual Conference on Theory and Applications of Models of Computation, pp. 377–389, 2020.

© The Author(s), 2021. Published by Cambridge University Press

https://doi.org/10.1017/S0960129521000104 Published online by Cambridge University Press

of articles have studied this problem and proposed many algorithms and techniques. The main techniques include local search, primal-dual, linear rounding, among others. One popular algorithm, put forward by Lloyd–Forgy (Lloyd, 1982), is called the "k-means" algorithm. It is widely used due to its simple and practicable features, but its theoretical performance is poor and the approximate ratio can reach infinity. Subsequently, Arthur and Vassilvitskii (2007) proposed the "k-means++" algorithm by randomly selecting the initial k central points with certain probability in the "k-means" algorithm. This initial randomization technique results in an approximation ratio of $O(\ln k)$. For the classic k-means problem, Kanungo et al. (2004) used the local search technique to give the first constant $(9+\varepsilon)$-approximation algorithm. The current best approximation algorithm is $6.357+\varepsilon$, which is given by Ahmadian et al. (2017) through the primal-dual method.

Given that the k-means problem has been investigated thoroughly, attention on this field has been focusing on the deformation of the k-means problem, special examples of the k-means problem, and approaches to improve the running time of the algorithm in various problems. In actual scenarios, we may refuse to serve some customers who are too far away by bearing certain penalties. Incoordinating this scenario into the mathematical model is a variant of the k-means problem, that is, the k-means problem with penalties.

A formal k-means problem with penalties is described as follows. We are given a constant k and a data set $\mathcal{D} \subseteq \mathcal{R}^{\ell}$ with n clients, where each client is assigned a penalty fee p_j. The intent is to pick a central point set $\mathrm{CS} \subseteq \mathcal{R}^{\ell}$ of size k and an unserved customer subset $\mathcal{D}_p \subseteq \mathcal{D}$ such that the sum of the total squared Euclidean distances from all customers in $\mathcal{D}\backslash\mathcal{D}_p$ to the cluster center and the total of penalties for all unserved customers, that is, $\sum_{j\in\mathcal{D}\backslash\mathcal{D}_p} c(j,\mathrm{CS}) + \sum_{j\in\mathcal{D}_p} p_j$, is minimized, where $c(j,\mathrm{CS})$ represent the square of the Euclidean distance from client j to the nearest center point in CS.

In recent decades, researchers have made great contributions on the clustering problem with penalties. For example, for the penalized facility location problem, Charikar et al. (Cohen-Addad et al., 2019) utilize the primal-dual approach to give a $(3+\varepsilon)$-approximation algorithm. And the current best approximation ratio of $1.5148+\varepsilon$ is given by Li et al. (2015) through an LP-rounding algorithm. As to the penalized k-means problem, the first constant approximation algorithm is $25+\varepsilon$ by adopting the local search method (Zhang et al., 2019), and the current best approximation guarantee is $19.849+\epsilon$ by using the primal-dual method (Feng et al., 2019). Moreover, there are also other faster algorithms such as the seeding algorithm designed for k-means problem with penalties (Li, 2020; Li et al., 2020).

The main contribution of this paper is to apply the primal-dual method to give a quasi-polynomial time algorithm with an approximate ratio $6.357+\varepsilon$, which improves the best ratio in the past.

1.1 Organizations

Some preliminary knowledge is described in Section 2. We then give a detailed description and the corresponding conclusions of the JV-P(δ) algorithm in Section 3. In Section 4, we present a pseudo-polynomial time algorithm as well as the anlysis. Finally, some discussions are stated in Section 5.

2. Preliminaries

Given an example of the k-means problem with penalties, we can use the standard discrete technique in Feldman et al. (2007) to transform the original problem into a discrete problem at a small cost, that is, a discrete k-means problem with penalties, where the center point is selected in a set $\mathcal{F}$ instead of selecting in the entire space $\mathcal{R}^{l}$, and the optimal value of the discrete problem is at most $(1+\varepsilon)$ times higher than the optimal value of the original problem, where $\varepsilon > 0$ is an arbitrarily small constant.

https://doi.org/10.1017/S0960129521000104 Published online by Cambridge University Press

Definition 1. *(The discrete k-means problem with penalties) Assume we are given an instance of the discrete k-means problem with penalties* $R = (\mathcal{D}, \mathcal{F}, d, \mathcal{P}, k)$*, where* $\mathcal{D}$ *and* $\mathcal{F}$ *represent the set of customers and facilities, respectively. Let* $c(j, i) = d^2(j, i)$ *be the connection cost from customers* j *to facility* i *where* $d(j, i)$ *is the Euclidean distance from* j *to* i*. If client* j *is not connected to any facility* $i \in \mathcal{F}$*,* j *will pay a penalty fee* p_j*. Let* $|\mathcal{D}| = n$*,* $|\mathcal{F}| = m$ *and* $\mathcal{P} = \{p_1, \cdots, p_n\}$*. The goal is to choose a subset* CS *of facilities with* $|\mathrm{CS}| \leq k$ *and a subset* $\mathcal{D}_p$ *of clients, such that*

$$\sum_{j \in \mathcal{D} \setminus \mathcal{D}_p} c(j, \mathrm{CS}) + \sum_{j \in \mathcal{D}_p} p_j$$

is minimized, where $c(j, \mathrm{CS}) = \min_{i \in \mathrm{CS}} c(j, i)$*.*

In the future, for the convenience of description, the discrete k-means problem with penalties is abbreviated as DKPP.

With reference to the facility location problem with penalties, DKPP's integer linear programming formulation can be easily interpreted as selecting k facilities to set up in the facility set $\mathcal{F}$ with zero cost. To formulate the integer linear program, we introduce three indicator vectors, $y \in \mathcal{R}^{|\mathcal{F}|}$, $x \in \mathcal{M}^{|\mathcal{F}| \times |\mathcal{D}|}$, and $r \in \mathcal{R}^{|D|}$, where $y_i \in \{0, 1\}$ indicates whether the facility i is offered $x_{ij} \in \{0, 1\}$ represents whether customer j is clustered to facility i and $r_j \in \{0, 1\}$ denotes whether or not customer j is clustered to a facility.

We relax the three indicator vectors to the entire nonnegative real number space to receive the following standard relaxation program KM-P:

$$\begin{aligned}
\min \quad & \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{D}} x_{ij} c(j, i) + \sum_{j \in \mathcal{D}} p_j \cdot r_j & & \\
\text{s.t} \quad & \sum_{i \in \mathcal{F}} x_{ij} + r_j \geq 1 & \forall j \in \mathcal{D} & (1) \\
& x_{ij} \leq y_i & \forall j \in \mathcal{D}, \forall i \in \mathcal{F} & (2) \\
& \sum_{i \in \mathcal{F}} y_i \leq k & & (3) \\
& x_{ij}, y_i, r_j \geq 0 & \forall j \in \mathcal{D}, \forall i \in \mathcal{F}. & (4)
\end{aligned}$$

Constraints (1) represent each customer is either clustered to a facility or punished, constraints (2) say that customers can only be clustered to established facilities, and constraint (3) illustrates that at most k facilities are established.

In order to resolve the difficulty caused by the constraint (3) to solve KM-P, we adopt the Lagrangian multiplier method (Jain and Vazirani, 2001), which introduces the Lagrangian multipliers λ for the constraint in (3), resulting in a new linear program KM-P(λ):

$$\begin{aligned}
\min \quad & \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{D}} x_{ij} c(j, i) + \sum_{j \in \mathcal{D}} p_j \cdot r_j + \lambda \left(\sum_{i \in \mathcal{F}} y_i - k \right) & & \\
\text{s.t} \quad & \sum_{i \in \mathcal{F}} x_{ij} + r_j \geq 1 & \forall j \in \mathcal{D} & (5) \\
& x_{ij} \leq y_i & \forall j \in \mathcal{D}, \forall i \in \mathcal{F} & (6) \\
& x_{ij}, y_i, r_j \geq 0 & \forall j \in \mathcal{D}, \forall i \in \mathcal{F}. & (7)
\end{aligned}$$

Let OPT_k and $\mathrm{OPT}_{\mathrm{L}}(\lambda)$ denote the optimal values of KM-P and KM-P(λ), respectively. Then we have $\mathrm{OPT}_{\mathrm{L}}(\lambda) \leq \mathrm{OPT}_k$ for $\lambda \geq 0$. Next we give the dual programming DKM-P(λ) of KM-P(λ).

https://doi.org/10.1017/S0960129521000104 Published online by Cambridge University Press

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

$$\text{s.t} \quad \sum_{j\in\mathcal{D}} [\alpha_j - c(j,i)]^+ \le \lambda \qquad \forall i \in \mathcal{F} \tag{8}$$

$$\alpha_j \le p_j \qquad \forall j \in \mathcal{D} \tag{9}$$

$$\alpha_j \ge 0 \qquad \forall j \in \mathcal{D}, \tag{10}$$

where $[\alpha_j - c(j,i)]^+ = \max\{\alpha_j - c(j,i), 0\}$. Let $\mathrm{OPT_D}(\lambda)$ denote the optimal values of DKM-P(λ). Hence we have $\mathrm{OPT_D}(\lambda) \le \mathrm{OPT_L}(\lambda) \le \mathrm{OPT}_k$ for $\lambda \ge 0$.

Observing KM-P(λ) and DKM-P(λ), if we omit the constant term $\lambda \cdot k$ in the objective function, they are the standard linear programming relaxation of the facility location problem with penalties and the corresponding dual programming, in which the facility opening cost is λ. Therefore, we can give the improved JV-P(δ) algorithm to solve DKM-P(λ) and get a constant approximate ratio of $6.357 + \varepsilon$. If the number of solutions output by the algorithm is no more than k, we will look for a feasible solution to the original problem, where the target value corresponding to this solution is at most $6.357 + \varepsilon$ times the optimal value OPT_k of the original problem.

3. The JV-P(δ) Algorithm and the Main Conclusions

Next we recall the JV-P(δ) algorithm and its main conclusions.

The JV-P algorithm is put forward by Charikar et al. (2001) when solving the facility location problem with penalties. We give the JV-P(δ) algorithm based on the JV-P algorithm. The algorithm includes two stages, generating a dual solution(GD) and selecting a center set(SS).

In the GD stage, the algorithm outputs a feasible solution α of DKM-P(λ).

After the GD stage, we obtain a set of temporarily opened facilities $\mathcal{F}_y$, in which there may be a customer j contributing to the opening of multiple facilities, that is $|\{i \in \mathcal{F}_y : \alpha_j > c(j,i)\}| > 1$, and these facilities are relatively close. The opening cost of these facilities can be reduced by closing some of them. For this, we introduce the SS stage, where it will opt for a subset of temporarily opened facilities to open.

We first construct two graphs, the customer-facility graph G and the pick-facility graph H. Before that, we introduce some notations.

A. $\mathcal{F}_y$: the set of facilities temporarily opened during the GD stage.

B. $\forall j \in \mathcal{D}$, the neighbor of customer j is represented as
$$N(j) := \{i \in \mathcal{F} : \alpha_j - c(j,i) > 0\}.$$

C. $\forall i \in \mathcal{F}$, the neighbor of facility i is represented as
$$N(i) := \{j \in \mathcal{D} : \alpha_j - c(j,i) > 0\}.$$

D. For a temporarily opened facility i, we let $t_i := \max_{j\in N(i)} \alpha_j$, and $t_i := 0$ if $N(i) = \emptyset$. Therefore we obtain $t_i \ge \alpha_j$, $\forall j \in N(i)$.

E. For a customer j and his corresponding witness $w(j)$, $\alpha_j \ge \min\{t_{w(j)}, p_j\}$.

- the customer-facility graph $G = (\mathcal{D} \cup \mathcal{F}_y, E)$: if $i \in N(j)$, we have $(j,i) \in E$.
- the pick-facility graph $H = (\mathcal{F}_y, E)$: $\forall j \in \mathcal{D}$, if $j \in N(i) \cap N(i')$ with $c(i,i') \le \delta \min\{t_i, t_{i'}\}$, where δ is a constant and $\delta \approx 2.315$, we have $(i,i') \in E$.

From the JV-P(δ) algorithm, we obtain the following conclusions.

https://doi.org/10.1017/S0960129521000104 Published online by Cambridge University Press

Algorithm 1 GD

1. Initially, set $\alpha := 0$, $\beta_{ij} := \max\{\alpha_j - c(j,i), 0\} = [\alpha_j - c(j,i)]^+$. Let $A := D$ indicate the set of active customers, and θ denote time. Starting from zero, $\forall j \in A$, α_j and θ rise at the same time at a unit speed. Once $\alpha_j = p_j$, we say that j is frozen, and α_j stops growing. If customer j causes one of the following cases to occur, we will update A by deleting j with $\alpha_j \geq c(j,i)$ from A.

   Case 1. There is a facility i such that the dual constraint

   $$\sum_{j \in \mathcal{D}} [\alpha_j - c(j,i)]^+ = \lambda$$

   is satisfied. Then we suppose that the facility i is temporarily opened. And we claim that the facility i is the witness of these customers with $\alpha_j \geq c(i,j), \forall j \in A$, expressed as $w(j) = i$.

   Case 2. There exists an active customer $j \in A$ reaching a tight edge, i.e. $\alpha_j - c(j,i) = 0$, to some already temporarily opened facility i. So we assume that the facility i is the witness of this customer.

2. When either $A = \emptyset$ or j is frozen $\forall j \in A$, the GD stage is completed.

Algorithm 2 SS

Step 1 We opt for a maximal independent set CS in H.

Step 2 We open all facilities of CS.

Step 3 We define the set of customers who are not served, that is, the set of customers who are punished, as follows:

$$\mathcal{D}_p := \{j \in \mathcal{D} : j \notin N(i), \forall i \in \text{CS and } \alpha_j = p_j\}.$$

Step 4 The rest of the customers gather to the nearest opened facility.

Lemma 1. *Given an instant R of DKPP, we assume that d is a Euclidean metric on $\mathcal{D} \cup \mathcal{F}$. Then, for any $\lambda \geq 0$, Algorithm 1 outputs a feasible solution α to DKM-P(λ) and Algorithm 2 outputs a set* CS *of opened facilities and a set $\mathcal{D}_p$ of punished clients by requesting $\delta \approx 2.315$ and $\rho \approx 6.357$ such that*

$$\sum_{j \in \mathcal{D} \setminus \mathcal{D}_p} c(j, \text{CS}) + \sum_{j \in \mathcal{D}_p} p_j \leq \rho \cdot \left(\sum_{j \in \mathcal{D}} \alpha_j - \lambda |\text{CS}| \right). \tag{11}$$

Proof. According to Algorithm 1, we can easily get that α is a feasible solution for DKM-P(λ), $\forall \lambda \geq 0$. For the rest, we prove that α and CS satisfy inequality (11).

Now if we obtain

$$\frac{c(j, \text{CS})}{\rho} \leq \alpha_j - \sum_{i \in N(j) \cap \text{CS}} (\alpha_j - c(j,i)) = \alpha_j - \sum_{i \in \text{CS}} [\alpha_j - c(j,i)]^+ \forall j \in \mathcal{D} \setminus \mathcal{D}_p \tag{12}$$

$$\frac{p_j}{\rho} \leq \alpha_j - \sum_{i \in \text{CS}} [\alpha_j - c(j,i)]^+ \quad \forall j \in \mathcal{D}_p, \tag{13}$$

https://doi.org/10.1017/S0960129521000104 Published online by Cambridge University Press

by adding up the $\frac{c(j,\mathrm{CS})}{\rho}$ of all clients and $[\alpha_j - c(j,i)]^+ = 0, \forall i \in \mathrm{CS}, j \in \mathcal{D}_p$, we get

$$\frac{1}{\rho}\left(\sum_{j\in\mathcal{D}\setminus\mathcal{D}_p} c(j,\mathrm{CS}) + \sum_{j\in\mathcal{D}_p} p_j\right) \le \sum_{j\in\mathcal{D}} \alpha_j - \sum_{j\in\mathcal{D}\setminus\mathcal{D}_p}\sum_{i\in\mathrm{CS}} [\alpha_j - c(j,i)]^+.$$

Moreover, note that all the facilities opened in CS and all customers who have made positive contributions to the opening of these facilities belong to $\mathcal{D}\setminus\mathcal{D}_p$. So we have $\sum_{j\in\mathcal{D}\setminus\mathcal{D}_p}[\alpha_j - c(j,i)]^+ = \lambda, \forall i \in \mathrm{CS}$. Therefore, (11) is satisfied.

For each client j, let S_j indicate the set of facilities to which j contributes in CS, that is $S_j := \{i \in \mathrm{CS} : \alpha_j > c(j,i)\}$. Then, according to the members of S_j, we divide clients into three cases, $\mathcal{D}_0 = \{j \in \mathcal{D}\setminus\mathcal{D}_p : |S_j| = 0\}$, $\mathcal{D}_1 = \{j \in \mathcal{D}\setminus\mathcal{D}_p : |S_j| = 1\}$, and $\mathcal{D}_{>1} = \mathcal{D}\setminus\mathcal{D}_1\setminus\mathcal{D}_p = \{j \in \mathcal{D}\setminus\mathcal{D}_p : |S_j| > 1\}$.

Denote $s_j = |S_j|$ and $\beta_{ij} = [\alpha_j - c(j,i)]^+$. Now, we will discuss each case separately.

Case 1: $s_j = 0$. Based on the definition of $\mathcal{D}_0$, we know that for any $j \in \mathcal{D}_0$, there is an $w(j) \in \mathcal{F}$ such that $\alpha_j \ge t_{w(j)}$ and $\alpha_j \ge c(j, w(j))$.
Further, we will discuss the following two events separately for $w(j)$.

Event 1. For $w(j) \in \mathrm{CS}$, obviously there is, $c(j,\mathrm{CS}) \le c(j, w(j))$.

Event 2. For $w(j) \notin \mathrm{CS}$, according to SS stage, there is an facility $i \in \mathrm{CS}$ which satisfies,

$$c(w(j), i) \le \delta \min\{t_{w(j)}, t_i\} \le \delta\alpha_j.$$

In this event, we have

$$\begin{aligned} d(j,\mathrm{CS}) &\le d(j,i) \\ &\le d(j,w(j)) + d(w(j),i) \\ &\le \sqrt{\alpha_j} + \sqrt{\delta\alpha_j} = (1+\sqrt{\delta})\sqrt{\alpha_j} \\ \Rightarrow c(j,\mathrm{CS}) &\le (1+\sqrt{\delta})^2\alpha_j. \end{aligned} \tag{14}$$

Recall the definition of $\mathcal{D}_0$, that is, $\sum_{i\in\mathrm{CS}}[\alpha_j - c(j,\mathrm{CS})]^+ = 0$. Together with formula (14), we have that

$$c(j,\mathrm{CS}) \le (1+\sqrt{\delta})^2\left(\alpha_j - \sum_{i\in\mathrm{CS}}[\alpha_j - c(j,\mathrm{CS})]^+\right).$$

If $\rho \ge (1+\sqrt{\delta})^2$, we are able to obtain

$$\frac{c(j,\mathrm{CS})}{\rho} \le \alpha_j - \sum_{i\in\mathrm{CS}}[\alpha_j - c(j,\mathrm{CS})]^+.$$

Case 2: $s_j = 1$. In this case, we can only find one facility i^* in IS that satisfies $[\alpha_j - c(j,i^*)]^+ > 0$. Hence, we obtain

$$c(j,\mathrm{CS}) \le c(j,i^*) = \alpha_j - (\alpha_j - c(j,i^*)) = \alpha_j - \sum_{i\in\mathrm{CS}}[(\alpha_j - c(j,i)]^+.$$

Case 3: $s_j \ge 2$. Based on the ℓ-dimensional Euclidean space, $\forall j \in \mathcal{D}_{>1}$,

$$\sum_{i\in S_j} c(j,i) \ge \sum_{i\in S_j} c(i,\mu) = \frac{\sum_{i\in S_j}\sum_{\tilde{i}\in S_j} c(i,\tilde{i})}{2s_j} \tag{15}$$

https://doi.org/10.1017/S0960129521000104 Published online by Cambridge University Press

is established, where $\mu = \frac{1}{s_j}\sum_{i\in S_j} i$ represents the centroid of S_j. And because $c(i,\tilde{i}) > \delta \min\{t_i, t_{\tilde{i}}\} \geq \delta\alpha_j$, $\forall i \in S_j$, we deduce that

$$\begin{aligned}\frac{\sum_{i\in S_j}\sum_{\tilde{i}\in S_j} c(i,\tilde{i})}{2s_j} &= \frac{\sum_{i\in S_j}\sum_{\tilde{i}\in S_j, i\neq\tilde{i}} c(i,\tilde{i})}{2s_j} \\ &\geq \frac{\sum_{i\in S_j}\sum_{\tilde{i}\in S_j, i\neq\tilde{i}} \delta\alpha_j}{2s_j} \\ &= \frac{s_j-1}{2}\delta\cdot\alpha_j. \end{aligned} \tag{16}$$

Combining (15) and (16), for $\delta \geq 2$, we have

$$\begin{aligned}\sum_{i\in S_j}(\alpha_j - c(j,i)) &\leq \left(s_j - \frac{s_j-1}{2}\delta\right)\alpha_j \\ &\leq \left(s_j\left(1-\frac{\delta}{2}\right)+\frac{\delta}{2}\right)\alpha_j \\ &\leq \left(2-\frac{\delta}{2}\right)\alpha_j. \end{aligned} \tag{17}$$

Among them, $s_j > 1$, and $s_j(1-\frac{\delta}{2})+\frac{\delta}{2}$ is a nonincreasing discrete function. $\forall j \in \mathcal{D}_{>1}$, $i \in S_j$, we have $c(j, \mathrm{CS}) \leq c(j,i) \leq \alpha_j$. So according to (17), we get

$$\begin{aligned}\left(\frac{\delta}{2}-1\right)c(j,\mathrm{CS}) &= \left(1-\left(2-\frac{\delta}{2}\right)\right)c(j,\mathrm{CS}) \leq \left(1-\left(2-\frac{\delta}{2}\right)\right)\alpha_j \\ &= \alpha_j - \left(2-\frac{\delta}{2}\right)\alpha_j \leq \alpha_j - \sum_{i\in S_j}(\alpha_j - c(j,i)).\end{aligned}$$

If $\rho \geq \frac{1}{\frac{\delta}{2}-1}$, we are able to receive

$$\frac{c(j,\mathrm{CS})}{\rho} \leq \alpha_j - \sum_{i\in \mathrm{IS}}[\alpha_j - c(j,\mathrm{CS})]^+.$$

Gathering the above three cases, we take

$$\rho(\delta) := \max\left\{(1+\sqrt{\delta})^2, \frac{1}{\frac{\delta}{2}-1}\right\},$$

where $\delta \geq 2$.

Finally, by setting $\delta \approx 2.315$, $\rho \approx 6.357$, (12) is satisfied.

For (13), we have $\alpha_j = p_j$, and $\alpha_j \leq c_{ij}$, $\forall j \in \mathcal{D}_p$, $\forall i \in \mathrm{CS}$. So $\sum_{i\in\mathrm{CS}}[\alpha_j - c(j,\mathrm{CS})]^+ = 0$. Hence we only need to ensure that $\rho > 1$, which implies (13). □

Lemma 1 shows that once $|\mathrm{CS}| \leq k$, we get a 6.357-approximation algorithm for DKPP. But as the JV-P(δ) algorithm shows, unless some information about λ is given, $|\mathrm{CS}| \leq k$ may not be guaranteed. Intuitively, we observe that λ represents the cost of opening a facility. If the cost of setting up facilities is small enough, we tend to open all the facilities. On the contrary, once the cost of setting up facilities is high, we will open as few facilities as possible. Therefore, we wish to give an appropriate λ such that $|\mathrm{CS}| = k$. Inspired by Ahmadian et al. (2017), we enumerate λ in a pseudo-polynomial time algorithm to obtain a number of corresponding solution CS as k.

https://doi.org/10.1017/S0960129521000104 Published online by Cambridge University Press

Algorithm 3 Eλ

1. Initially, assume $\lambda = 0$, $\alpha_j^{in} = \min\{\min_{i\in\mathcal{F}} c(j,i), p_j\}$, and $CS = \mathcal{F}$.
2. Set $\alpha = \alpha^{in}$, and $\lambda = \lambda + \varepsilon_z$. Let $A = \emptyset$ be the set of active customers and let θ denote time. Starting from zero, once $\theta = \alpha_j, \forall j \in \mathcal{D}$, uptade $A := A \cup \{j\}$. Both α_j and θ rise at the same time at a unit speed. When $\alpha_j = p_j$, we say that j is frozen, and α_j stops growing. If customer j causes the following case to occur, we replace A by deleting j with $\alpha_j \geq c(j,i)$ from A.

   Case: j has a tight edge to a temporarily opened facility i with $B(\alpha_j) \geq B(t_i)$. Assume that the facility i is the witness of this customer, expressed as $w(j) = i$.

3. When Step 2 runs, we decrease α_j with $B(\alpha_j) > B(\theta)$ by $|A|$ times the growth rate of θ.
4. When either $A = \emptyset$ or j is frozen $\forall j \in A$, output $\alpha^{out} = \alpha$.
5. For $\lambda = L \cdot \varepsilon_z$, the algorithm ends; otherwise $\alpha^{in} = \alpha^{out}$, and go back to Step 2.

4. The Pseudo-Polynomial Algorithm as Well as Our Main Results

In Ahmadian et al. (2017), by enumerating λ, a (6.357+ε)-approximation algorithm of pseudo-polynomial time is obtained for the k-means problem. Similarly, we present a (6.357+ε)-pseudo polynomial time approximation algorithm for DKPP. For convenience, we abbreviate this algorithm as PPTA, which contains two sub-algorithms: Eλ and Ck.

Eλ enumerates all possible values of λ, namely $\lambda = 0, 1 \cdot \varepsilon_z, \cdots, L \cdot \varepsilon_z$, where $\varepsilon > 0$ is a sufficiently small constant.

$$n \gg 1/\varepsilon, \varepsilon_z = n^{-3-30log_{1+\varepsilon}n}, L = 4n^7 \cdot \varepsilon_z^{-1}.$$

When $\lambda = 0$ and $\alpha_j^0 = \min\{\min_{i\in\mathcal{F}} d^2(j,i), p_j\}$, it is obviously a feasible solution of DKM-P(0) and $CS^0 = \mathcal{F}$, which satisfies Lemma 1. For each $\lambda > 0$, the Eλ algorithm, with input $\alpha_j^{\lambda-\varepsilon_z}$, outputs a feasible solution α_j^{λ} of DKM-P(λ). In addition, for the simplicity of algorithm, we introduce a piecewise function B to divide the α values of clients.

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

Simultaneously, when $\lambda = L \cdot \varepsilon_z$, $|CS| \leq 1$ can be guaranteed. Before the algorithm is described, we preprocess any instance of DKPP, so that with a small loss, the cost of any client j to any facility j satisfies: $1 \leq c(j,i) \leq n^6$ and $p_j \geq 1, \forall j \in \mathcal{D}$, where $n = |\mathcal{D}|$. The proof can be referred to Ahmadian et al. (2017).

From the Eλ algorithm, we know that α^{λ} is a feasible solution for DKM-P(λ), either j has a tight edge to a temporarily opened facility i or $\alpha_j = p_j$ where α_j stops increasing. In each solution $\alpha = \alpha^{\lambda}$ ($\lambda = 0, 1 \cdot \varepsilon_z, \cdots, L \cdot \varepsilon_z$), α^{λ} and $\alpha^{\lambda+\varepsilon_z}$ are close.

Lemma 2. *Ahmadian et al. (2017) The corresponding solutions α^{in} and α^{out} in Eλ are close, that is $|\alpha_j^{in} - \alpha_j^{out}| \leq \frac{1}{n^2}$ for all $j \in \mathcal{D}$.*

Proof. From the above assumption, we have $1 \leq d^2(j,i) \leq n^6$. We notice that at any time the largest $\alpha-$value is at most

$$\max\{\lambda\} + n^6 = L \cdot \varepsilon_z + n^6 = 4n^7 + n^6 \leq 5n^7. \tag{18}$$

Hence, $\forall j \in \mathcal{D}$, $B(\alpha_j) \leq 1 + \lfloor \log_{1+\varepsilon} 5n^7 \rfloor \leq 10 \log_{1+\varepsilon} n$. Similar to Ahmadian et al. (2017), we can still get the following claim.

https://doi.org/10.1017/S0960129521000104 Published online by Cambridge University Press

Claim. Any α can increase $\varepsilon_z n^{3b}$ at most while $B(\theta) \leq b$.

Proof. We prove it by using the induction on $b = 0, 1, \cdots, 10 \log_{1+\varepsilon} n$.

For $b = 0$, it is trivially true that there is no client whose $\alpha-$value increases or decreases with $B(\alpha_j) = 0$.

For $b > 0$, assume that α_j can increase by at most $\varepsilon_z n^{3b-3}$ when $B(\theta) \leq b-1$ is correct. We shall prove that α_j can increase by at most $\varepsilon_z n^{3b}$ while $B(\theta) \leq b$ is true.

Before the proof, we assume that some α_j is increasing and has at least one increased ε_z while $B(\theta) \leq b$. So we know that $j \in A$ and $\alpha_j = \theta$.

First, suppose that $\alpha_j < \alpha_j^{in}$. Then we have that α_j was decreased while $B(\alpha_j) \leq b$. Through Algorithm 3, we know that α_j only decreases whenever $B(\theta) < B(\alpha_j)$, and hence α_j was decreased only while $B(\theta) < b$. Because we assume that $\alpha_{j'}$ can increase by at most $\varepsilon_z n^{3b-3}$ when $B(\theta) \leq b-1$, therefore α_j decreases by at most $n \cdot \varepsilon_z n^{3b-3}$ while $B(\theta) \leq b-1$. Thus, we get $\alpha_j = \alpha_j^{in}$ again after α_j has increased by at most $\varepsilon_z n^{3b-2}$.

Second, we start with $\alpha_j = \alpha_j^{in}$ and now consider how much α_j can be increased while $B(\theta) \leq b$. Let i be a witness of j in α^{in}, and we have $B(\alpha_{j'}^{in}) \leq B(\alpha_j^{in}) \leq B(\alpha_j) = B(\theta) \leq b, \forall j' \in N^{in}(i)$. Thus $j' \in N^{in}(i)$ are decreased by Algorithm 3 only while $B(\theta) < b$ and $\alpha_{j'}$ is decreased at most $\varepsilon_z n^{3b-2}$. Hence, with the increase of α_j at most $\varepsilon_z n^{3b-2} \cdot (n-1) + \varepsilon_z$, i will again be tight, or $\alpha_j = p_j$, and j is frozen.

In either case, the total amount α_j can increase by at most $\varepsilon_z n^{3b-2} + \varepsilon_z n^{3b-2} \cdot (n-1) + \varepsilon_z \leq \varepsilon_z \cdot n^{3b}$ while $B(\theta) \leq b$. □

Let $\varepsilon_z = n^{-3-30log_{1+\varepsilon} n}$. Then from the above claim, we have that $\alpha_j^{out} - \alpha_j^{in} \leq \frac{1}{n^3}$, and $\alpha_j^{in} - \alpha_j^{out} \leq \frac{1}{n^2}$ since α_j decreases by no more than n times the maximum increase in the $\alpha-$value of any client. □

The purpose of Ck is to find a feasible solution with number k of DKPP between $\alpha_j^{\lambda-\varepsilon_z}$ and $\alpha_j^{\lambda}, \forall \lambda = 1 \cdot \varepsilon_z, \cdots, L \cdot \varepsilon_z$. For each solution α^{λ}, we construct the customer-facility graph G^{λ}, the pick-facility graph H^{λ} and the maximal independent set CS^{λ} in the same way as those in Section 3 (See Algorithm 4 for specific operations).

Through the above analysis and the value of L, we know that when $\lambda = 0$, $|\mathrm{CS}| = m$, and when $\lambda = L \cdot \varepsilon_z$, $|\mathrm{CS}| = 1$. Then from the structure of Algorithm 4, whenever the number of solutions is reduced by at most one, there must be a λ, and the number of solutions output by Algorithm 4 is k.

4.1 Approximation ratio analysis

Without loss of generality, we suppose $\mathrm{CS} = \mathrm{CS}^{(\lambda,l)}$, $G = G^{(\lambda,l)}$, and $H = H^{(\lambda,l)}$, where $\mathrm{CS}^{(\lambda,l)}$ is the solution output by the Algorithm 4, and $G^{(\lambda,l)}$ and $H^{(\lambda,l)}$ correspond to the customer-facility graph and the pick-facility graph, respectively. Now, we first set up the facilities in CS, and second give the definition of the client set $\mathcal{D}_p$ that is ultimately penalized,

$$\mathcal{D}_p := \left\{ j \in \mathcal{D} : j \notin N(i), \forall i \in \mathrm{CS} \text{ and } \alpha_j^{\lambda+\varepsilon_z} = p_j \right\},$$

where $j \notin N(i)$ for each $i \in \mathrm{CS}$ is expressed as $\alpha_j^{\lambda} \leq c(j,i)$ if $i \in H^{\lambda}$, or $\alpha_j^{\lambda+\varepsilon_z} \leq c(j,i)$ if $i \in H^{\lambda+\varepsilon_z}$. The remaining customers are assigned to the nearest opened facility.

https://doi.org/10.1017/S0960129521000104 Published online by Cambridge University Press

Algorithm 4 Ck:(G^{λ}, $G^{\lambda+\varepsilon_z}$, H^{λ},CS^{λ})

1. Initially, we set $\lambda = 0$.
2. Input $G^{\lambda} = (\mathcal{D} \cup \mathcal{F}_y^{\lambda}, E^{\lambda})$, $G^{\lambda+\varepsilon_z} = (\mathcal{D} \cup \mathcal{F}_y^{\lambda+\varepsilon_z}, E^{\lambda+\varepsilon_z})$, H^{λ}, $H^{\lambda+\varepsilon_z}$, and CS^{λ}.
3. We establish a customer-facility graph $G^{(\lambda,1)}$, which is also a bipartite graph of $\mathcal{D}$ and $\mathcal{F}_y^{\lambda} \cup \mathcal{F}_y^{\lambda+\varepsilon_z}$. There is an edge from client j to facility $i \in \mathcal{F}_y^{\lambda}$ if (j, i) belongs to G^{λ} and to $i \in \mathcal{F}_y^{\lambda+\varepsilon_z}$ if (j, i) belongs to $\mathrm{G}^{\lambda+\varepsilon_z}$. The opening time t_i of facility i is now trivially set to t_i^{λ} if $i \in \mathcal{F}_y^{\lambda}$ and to $t_i^{\lambda+\varepsilon_z}$ if $i \in \mathcal{F}_y^{\lambda+\varepsilon_z}$.
4. Based on $G^{(\lambda,1)}$ and t, we produce the pick-facility graph $H^{(\lambda,1)}$ and output a maximal independent set $\mathrm{CS}^{(\lambda,1)}$ of $H^{(\lambda,1)}$ by greedily extending CS^{λ}.
5. Let $p = 2$.
   while $p \neq |\mathcal{F}_y^{\lambda}| + 2$ **do**
   Form $G^{(\lambda,p)}$ by deleting a facility $i \in \mathcal{F}_y^{\lambda}$ from $G^{(\lambda,p-1)}$. Then we build a new pick-facility graph $H^{(\lambda,p)}$ from $G^{(\lambda,p)}$ and output its maximal independent set $\mathrm{CS}^{(\lambda,p)}$ by greedily extending $\mathrm{CS}^{(\lambda,p-1)} \backslash \{i\}$.
   if $|\mathrm{CS}^{(\lambda,p)}| = k$, **then**
   the algorithm stops, and outputs $\mathrm{CS}^{(\lambda,p)}$.
   else
   $p = p + 1$.
   end if
   end while
6. At the end of the procedure (after $|\mathcal{F}_y^{\lambda}|$ many steps), $G^{\lambda+\varepsilon_z} = G^{(\lambda,|\mathcal{F}_y^{\lambda}|+1)}$, and $H^{\lambda+\varepsilon_z} = H^{(\lambda,|\mathcal{F}_y^{\lambda}|+1)}$. We set $\mathrm{CS}^{\lambda+\varepsilon_z} = \mathrm{CS}^{(\lambda,|\mathcal{F}_y^{\lambda}|+1)}$ and $\lambda = \lambda + \varepsilon_z$. Go to Step 2.

Let

$$\alpha_j := \begin{cases} \min\left\{\alpha_j^{\lambda}, \alpha_j^{\lambda+\varepsilon_z}\right\}, & j \in \mathcal{D} \backslash \mathcal{D}_p, \\ p_j, & j \in \mathcal{D}_p. \end{cases}$$

Note that $\alpha \leq \alpha^{\lambda+\varepsilon_z}$. Hence α is a feasible solution of DKM-P($\lambda + \varepsilon_z$). Since α^{λ} and $\alpha^{\lambda+\varepsilon_z}$ are close, we get $\alpha_j \geq \alpha^{\lambda} - 1/n^2$ and $\alpha_j \geq \alpha^{\lambda+\varepsilon_z} - 1/n^2$ for each $j \in \mathcal{D}$. Therefore, CS satisfies the following Lemmas 3–6 and Theorem 1.

For each client j, we denote S_j as the set of facilities that j contributes in CS, namely $S_j := \{i \in \mathrm{CS} : \alpha_j > c(j, i)\}$. Then, according to the members of S_j, we divide clients into two cases.

Case 1: $\mathcal{D}_0 = \{j \in \mathcal{D} \backslash \mathcal{D}_p : S_j = \emptyset\}$.

Case 2: $\mathcal{D}_{>0} = \mathcal{D} \backslash \mathcal{D}_0 \backslash \mathcal{D}_p = \{j \in \mathcal{D} \backslash \mathcal{D}_p : S_j \neq \emptyset\}$.

We also let $\beta_{ij} = [\alpha_j - c(j, i)]^+$ and similarly $\beta_{ij}^{\lambda} = [\alpha_j^{\lambda} - c(j, i)]^+$ and $\beta_{ij}^{\lambda+\varepsilon_z} = [\alpha_j^{\lambda+\varepsilon_z} - c(j, i)]^+$. Now, we discuss each case separately.

Lemma 3. *$\forall j \in \mathcal{D}_{>0}$, we acquire* $c(j,\mathrm{CS}) \leq \rho \cdot (\alpha_j - \sum_{i \in S_j} \beta_{ij})$.

Please refer to Lemma 1 for the proof, and we will omit the details here.

https://doi.org/10.1017/S0960129521000104 Published online by Cambridge University Press

Lemma 4. $\forall j \in \mathcal{D}_0$, *we obtain* $c(j, \mathrm{CS}) \le (1+5\varepsilon)\rho \cdot \alpha_j$.

Proof. From Algorithm 4, we know that $\forall j \in \mathcal{D}_0$, there exists a facility $w(j) \in \mathcal{F}_y^{\lambda+\varepsilon_z}$ such that

$$B(\alpha_j) \ge B(t_{w(j)}) \Rightarrow (1+\varepsilon)\alpha_j^{\lambda+\varepsilon_z} \ge t_{w(j)}^{\lambda+\varepsilon_z},$$

$$\alpha_j^{\lambda+\varepsilon_z} \ge c(j, w(j)).$$

Because $\alpha_j = \min\left\{\alpha_j^{\lambda}, \alpha_j^{\lambda+\varepsilon_z}\right\}$, α^{λ} and $\alpha^{\lambda+\varepsilon_z}$ are close and $\alpha_j \ge 1$. From Lemma 2, we obtain

$$\alpha_j^{\lambda+\varepsilon_z} \le \alpha_j + \frac{1}{n^2} \le \left(1+\frac{1}{n^2}\right)\alpha_j.$$

If $w(j) \in \mathrm{CS}$, obviously there is, $d(j, \mathrm{CS}) \le d(j, w(j))$. If $w(j) \notin \mathrm{CS}$, there exists a facility $i^* \in \mathrm{CS}$, which satisfies, $c(w(j), i^*) \le \delta \cdot t_{w(j)}^{\lambda+\varepsilon_z} \le \delta \cdot (1+\varepsilon)\alpha_j^{\lambda+\varepsilon_z}$. In this case, we have

$$\begin{aligned} d(j, \mathrm{CS}) &\le d(j, i^*) \le d(j, w(j)) + d(w(j), i^*) \\ &\le \sqrt{\alpha_j^{\lambda+\varepsilon_z}} + \sqrt{\delta(1+\varepsilon)\alpha_j^{\lambda+\varepsilon_z}} \\ &\le (1+\sqrt{\delta}+\varepsilon)\sqrt{\alpha_j^{\lambda+\varepsilon_z}} \\ &\le (1+\sqrt{\delta}+\varepsilon)\sqrt{1+\frac{1}{n^2}}\sqrt{\alpha_j}. \\ &\le (1+2\varepsilon)(1+\sqrt{\delta})\sqrt{\alpha_j}. \end{aligned}$$

Hence

$$\frac{c(j, \mathrm{CS})}{(1+\sqrt{\delta})^2} \le (1+2\varepsilon)^2\alpha_j \le (1+5\varepsilon)\alpha_j.$$

This lemma is satisfied if $\rho \ge (1+\sqrt{\delta})^2$. □

Based on Algorithms 1–4 we get the following results.

Lemma 5. *We have* $p_j = \alpha_j$ *and* $\beta_{ij} = 0$, $\forall j \in \mathcal{D}_p$, $\forall i \in CS$.

Lemma 6. $\forall i \in$CS, *we obtain* $\sum_{j\in\mathcal{D}\setminus\mathcal{D}_p} \beta_{ij} \ge \lambda - \frac{1}{n}$.

Proof. Since $\alpha_j \ge \max\{\alpha_j^{\lambda+\varepsilon_z}, \alpha_j^{\lambda}\} - \frac{1}{n^2}$ for every client $j \in \mathcal{D} \setminus \mathcal{D}_p$, we have

$$\sum_{j\in\mathcal{D}\setminus\mathcal{D}_p} \beta_{ij} \ge \sum_{j\in\mathcal{D}\setminus\mathcal{D}_p} \left(\max\left\{\beta_j^{\lambda+\varepsilon_z}, \beta_j^{\lambda}\right\} - \frac{1}{n^2}\right) \ge \lambda - \frac{1}{n}.$$

The last inequality is from the follwoing argument: $\forall i \in \mathrm{CS}$, if $i \in \mathcal{F}_y^{\lambda+\varepsilon_z}$, $\sum_{j\in\mathcal{D}\setminus\mathcal{D}_p} \beta_{ij}^{\lambda+\varepsilon_z} = \lambda + \varepsilon_z$; otherwise, $i \in \mathcal{F}_y^{\lambda}$, $\sum_{j\in\mathcal{D}\setminus\mathcal{D}_p} \beta_{ij}^{\lambda+\varepsilon_z} = \lambda$. □

Theorem 1. *The solution* CS *output by Algorithm 4 satisfies*

$$\sum_{j\in\mathcal{D}\setminus\mathcal{D}_p} c(j, \mathrm{CS}) + \sum_{j\in\mathcal{D}_p} p_j \le (\rho + O(\varepsilon)) \cdot OPT_k,$$

where $\rho \approx 6.357$.

https://doi.org/10.1017/S0960129521000104 Published online by Cambridge University Press

Proof. By Lemmas 3–5, we get

$$\sum_{j\in\mathcal{D}\setminus\mathcal{D}_p} c(j,\mathrm{CS}) \le (1+5\varepsilon)\rho \sum_{j\in\mathcal{D}\setminus\mathcal{D}_p}\left(\alpha_j - \sum_{i\in S_j}\beta_{ij}\right),$$

$$\sum_{j\in\mathcal{D}_p} p_j \le (1+5\varepsilon)\rho \sum_{j\in\mathcal{D}_p}\alpha_j.$$

Therefore,

$$\sum_{j\in\mathcal{D}\setminus\mathcal{D}_p} c(j,\mathrm{CS}) + \sum_{j\in\mathcal{D}_p} p_j \le (1+5\varepsilon)\rho\left(\sum_{j\in\mathcal{D}}\alpha_j - \sum_{j\in\mathcal{D}\setminus\mathcal{D}_p}\sum_{i\in S_j}\beta_{ij}\right).$$

Because α is a feasible solution of DKM-P($\lambda+\varepsilon_z$), by Lemma 6, and $\sum_{i\in\mathrm{CS}}\beta_{ij} = \sum_{i\in S_j}\beta_{ij}$, we have

$$\begin{aligned}
&\sum_{j\in\mathcal{D}\setminus\mathcal{D}_p} c(j,\mathrm{CS}) + \sum_{j\in\mathcal{D}_p} p_j \\
&\le (1+5\varepsilon)\rho\left(\sum_{j\in\mathcal{D}}\alpha_j - \sum_{j\in\mathcal{D}\setminus\mathcal{D}_p}\sum_{i\in S_j}\beta_{ij}\right) \le (1+5\varepsilon)\rho\left(\sum_{j\in\mathcal{D}}\alpha_j - |\mathrm{IS}|\left(\lambda-\frac{1}{n}\right)\right) \\
&= (1+5\varepsilon)\rho\left(\sum_{j\in\mathcal{D}}\alpha_j - k\cdot(\lambda+\varepsilon_z) + \frac{k}{n} + k\cdot\varepsilon_z\right) \\
&= (1+5\varepsilon)\rho\left(\sum_{j\in\mathcal{D}}\alpha_j - k\cdot(\lambda+\varepsilon_z) + \frac{k}{n} + k\cdot n^{-3-30\log_{1+\varepsilon} n}\right) \\
&\le (1+5\varepsilon)\,\rho\left(\mathrm{OPT}_k + 1 + \frac{1}{n^3}\right) \le (1+5\varepsilon)\,\rho(\mathrm{OPT}_k+2) \\
&\le (1+5\varepsilon)\,\rho(1+2\varepsilon)\mathrm{OPT}_k \\
&= (\rho+O(\varepsilon))\mathrm{OPT}_k.
\end{aligned}$$

The penultimate inequality is due to

$$1 \le \varepsilon\cdot n \le \varepsilon\cdot\sum_{j\in\mathcal{D}}\min\left\{\min_{i\in\mathcal{F}} c(j,i), p_j\right\} \le \varepsilon\cdot\mathrm{OPT}_k.$$

□

4.2 Algorithm running time analysis

In Algorithm 3, E(λ), the total number of iterations, is at most L, and the running time in each iteration is a polynomial function of $m = |\mathcal{F}|$. Thus, the running time of E(λ) is $n^{O(\varepsilon^{-1}\log n)}$. In Algorithm 4, Ck, the enumeration of λ, iterates at most L times, and at each iteration, the running time of the algorithm is $O(n^2)$, where $n = |\mathcal{D}|$. Therefore, the running time of Ck is $n^{O(\varepsilon^{-1}\log n)}$ and so is the total running time.

5. Discussion

In this article, we discuss the penalized k-means problem. We employ the primal-dual technique and give a pseudo-polynomial time algorithm with an approximation ration of $(6.357+\varepsilon)$ in the

https://doi.org/10.1017/S0960129521000104 Published online by Cambridge University Press

Euclidean space. We note that the same algorithm is also applicable to the penalized k-median problem in the same Euclidean space to obtain a $(2.633 + \varepsilon)$-approximation algorithm for the problem.

We think that our algorithm can be enhanced to obtain a polynomial time algorithm with the same approximate ratio. Intuitively, we only need to transform ε_z into polynomial time. Inspired by Ahmadian et al. (2017), we may be able to obtain a polynomial time algorithm by increasing the opening cost of one facility ε_z instead of increasing the cost of all facilities ε_z. But compared to the pseudo-polynomial time algorithm, the algorithm and analysis are more complicated.

Acknowledgements. The first two authors are supported by Natural Science Foundation of China (No. 11871081) and Beijing Natural Science Foundation Project No. Z200002. The third author is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) grant 06446, and Natural Science Foundation of China (Nos. 11771386, 11728104). The fourth author is supported by Higher Educational Science and Technology Program of Shandong Province (No. J17KA171) and Natural Science Foundation of Shandong Province (No. ZR2020MA029) of China.

References

Ahmadian, S., Norouzi-Fard, A., Svensson, O. and Ward, J. (2017). Better guarantees for k-means and Euclidean k-median by primal-dual algorithms. In: *Proceedings of FOCS*, 61–72.

Aloise, D., Deshpande, A., Hansen, P. and Popat, P. (2009). NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning* **75** 245–248.

Arthur, D. and Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. In: *Proceedings of SODA*, 1027–1035.

Charikar, M., Khuller, S., Mount, D. M. and Narasimhan, G. (2001). Algorithms for facility location problems with outliers. In: *Proceedings of 12th ACM-SIAM Symposium on Discrete Algorithms*, 642–651.

Cohen-Addad, V., Klein, P. N. and Mathieu, C. (2019). Local search yields approximation schemes for k-means and k-median in Euclidean and minor-free metrics. *SIAM Journal on Computing* **48** 644–667.

Drineas, P., Frieze, A., Kannan, R., Vempala, S. and Vinay, V. (2004). Clustering large graphs via the singular value decomposition. *Machine Learning* **56** 9–33.

Feldman, D., Monemizadeh, M. and Sohler, C. (2007). A PTAS for $k-$means clustering based on weak coresets. In: *Proceedings of the twenty-Third Annual Symposium on Computational Geometry*, 11–18.

Feng, Q., Zhang, Z., Shi, F. and Wang, J. (2019). An improved approximation algorithm for the k-means problem with penalties. In: *Proceedings of FAW*, 170–181.

Friggstad, Z., Rezapour, M. and Salavatipour, M. R. (2019). Local search yields a PTAS for k-means in doubling metrics. *SIAM Journal on Computing* **48** 452–480.

Jain, K. and Vazirani, V. V. (2001). Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM* **48** 274–296.

Ji, S., Xu, D., Guo, L., Li, M. and Zhang, D. (2020). The seeding algorithm for spherical k-means clustering with penalties. *Journal of Combinatorial Optimization*. doi:10.1007/s10878-020-00569-1.

Kanungo, T., Mount, D., Netanyahu, N., Piatko, C. and Silverma, R. (2004). A local search approximation algorithm for k-means clustering. *Computational Geometry* **28** 89–112.

Li, M. (2020). The bi-criteria seeding algorithms for two variants of k-means problem. *Journal of Combinatorial Optimization*. doi:10.1007/s10878-020-00537-9.

Li, M., Xu, D., Yue, J., Zhang, D. and Zhang, P. (2020). The seeding algorithm for k-means problem with penalties. *Journal of Combinatorial Optimization* **39** 15–32.

Li, Y., Du, D., Xiu, N. and Xu, D. (2015). Improved approximation algorithms for the facility location problems with linear/submodular penalties. *Algorithmica* **73** 460–482.

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory* **28** 129–137.

Makarychev, K., Makarychev, Y. and Razenshteyn, I. (2019). Performance of Johnson-Lindenstrauss transform for k-means and k-medians clustering. In: *Proceedings of STOC*, 1027–1038.

Vazirani, V. V. (2001). *Approximation Algorithms*, Springer-Verlag, Berlin, Heidelberg.

Williamson, D. P. and Shmoys, D. B. (2011). *The Design of Approximation Algorithms*, Cambridge University Press, Cambridge.

Zhang, D., Hao, C., Wu, C., Xu, D. and Zhang, Z. (2019). Local search approximation algorithms for the k-means problem with penalties. *Journal of Combinatorial Optimization* **37** 439–453.

Cite this article: Ren C, Xu D, Du D and Li M (2022). An improved primal-dual approximation algorithm for the k-means problem with penalties. *Mathematical Structures in Computer Science* **32**, 151–163. https://doi.org/10.1017/S0960129521000104

https://doi.org/10.1017/S0960129521000104 Published online by Cambridge University Press