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

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
renchunying@emails.bjut.edu.cn, xudc@bjut.edu.cn
[2] Faculty of Management, University of New Brunswick, Fredericton, NB Canada E3B 5A3. ddu@unb.ca
[3] School of Mathematics and Statistics, Shandong Normal University, Jinan 250014, P.R. China. liminemily@sdnu.edu.cn

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 select a set $S \subseteq \mathcal{R}^\ell$ of size at most k and a punished subset $\mathcal{D}_p \subseteq \mathcal{D}$ to minimize the sum of the total squared distance of connected points and the total penalty cost of non-connected points, namely $\sum_{j\in\mathcal{D}\setminus\mathcal{D}_p} c(j,S) + \sum_{j\in\mathcal{D}_p} p_j$. For the general case, $P = \max_{j\in\mathcal{D}_p} p_j$ is a polynomial about n. We propose a pseudo-polynomial time $(6.357+\varepsilon)$-approximation algorithm for the k-means problem with penalty via the primal-dual technique, improving the previous best approximation ratio of $19.849+\epsilon$ in [8]. When P is a constant value, the pseudo-polynomial time can be transformed into polynomial time.

Keywords: Approximation algorithm · k-means problem with penalties · linear programming · dual programming

1 Introduction

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

The K-means problem is the most representative and widely investigated problem among clustering problems. Therefore, it is of great significance to learn

* 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.

** Corresponding author.

the k-means problem. In particular, the classic k-means problem has been under extensive study in operations research and computer science [4,9,11,16,17,18]. 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 goal is to select a subset S of at most k center points in $\mathcal{R}^\ell$ such that each point in $\mathcal{D}$ is connected to the closest cluster in S with minimum total squared distances, namely to minimize $\sum_{j\in\mathcal{D}} c(j,S)$, where $c(j,S)$ is the squared Euclidean distances from j to the nearest point in S.

In general, the problem is NP-hard [2,6], 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 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 [15], is called the "k-means" algorithm. It is widely used due to its simple and practicable features, but its theoretical analysis has poor boundaries, and the approximate ratio can reach infinity. In order to improve the performance guarantee, Arthur and Vassilvitskii [3] proposed a modified "k-means" algorithm, called k-means++, by randomly selecting the first initial k centers with a specific probability and demonstrate that this new algorithm has approximation ratio $O(\ln k)$. Additionally, to gain better results of performance guarantee, some researchers have made their contributions. The first constant polynomial time approximation algorithm for the k-means problem is given in [11] with an approximation ratio $9+\varepsilon$ based on local search technique. Currently, the best approximation factor for the k-means problem is $6.357+\varepsilon$ [1] by the primal-dual method.

Given that the classic k-means problem has been studied very thoroughly, research on this field has diverted its attention to the deformation of the k-means problem, special examples of the k-means problem, and how to improve the running speed of the algorithm in various problems. In practice, it is more desirable and beneficial to pay a penalty for not connecting some outliers in $\mathcal{D}$ that are too far away from most points. Therefore, the main focus of this work is to consider the k-means problem with penalties. Formally, we are given an integer k, and a data set $\mathcal{D} \subseteq \mathcal{R}^\ell$ of n points where each point $j \in \mathcal{D}$ is associated with a penalty cost p_j. The objective is to select a set $S \subseteq \mathcal{R}^\ell$ of size k and a subset $\mathcal{D}_p \subseteq \mathcal{D}$ such that the sum of the total squared distance of connected points and the total penalty cost of non-connected point, $\sum_{j\in\mathcal{D}\setminus\mathcal{D}_p} c(j,S) + \sum_{j\in\mathcal{D}_p} p_j$, is minimized. In the past two decades, tremendous efforts have been made on the issue of the clustering with penalties. For the facility location problem with penalties, Charikar et al.[5] gave a $(3+\varepsilon)$-approximation by using a primal-dual approach, and the current best approximation ratio ratio $(1.5148+\varepsilon)$ via an LP-rounding algorithm [14]. As to the k-means problem with penalty, the first constant approximation algorithm is $25+\varepsilon$ via local search technique [19], and the current best approximation ratio is $19.849+\epsilon$ by primal-dual technique [8]. Moreover, there are also other faster algorithms such as seeding algorithm designed for k-means problem with penalties [12,13].

Our principal contribution in this article is to give a quasi-polynomial time algorithm with the best $(6.357+\varepsilon)$-approximation ratio so far by adopting the primal-dual method, improving the previous best result.

1.1 Organizations

The outline of this article is as follows. Preliminaries are described in Section 2. Section 3 gives a detailed description and the corresponding conclusions of the JV-P(δ) algorithm. In section 4, we give a pseudo-polynomial time algorithm as well as the anlysis. Some discussions are stated in Section 5.

2 Preliminaries

Given an instance of the $k-$means problem with penalty, we can use the standard discrete technique in [7] to transform the original problem into a discrete problem at a small cost, that is, a discrete $k-$means problem with penalty, 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 original problem is not higher than $(1+\varepsilon)$ times the optimal value of the discrete problem, where ε is an arbitrarily small constant.

Definition 1. *(The discrete $k-$means problem with penalty) Given an instance of a discrete $k-$means problem with penalty $I=(\mathcal{D},\mathcal{F},\mathcal{P},d,k)$, where $\mathcal{D}$ and $\mathcal{F}$ represent the set of clients and facilities, respectively, let $c(j,i)=d^2(j,i)$ denote the connection cost of client j and facility 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 S of facilities and a subset $\mathcal{P}_s$ of clients, such that*

$$\sum_{j\in\mathcal{D}_s} p_j+\sum_{j\in\mathcal{D}\setminus\mathcal{D}_s} c(j,S)$$

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

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

With reference to the facility location problem, 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 sets of indicator variables: a variable y_i for each facility $i\in\mathcal{F}$, a variable x_{ij} for each facility-client pair $i\in\mathcal{F}$, $j\in\mathcal{D}$ and a variable r_j for each client $j\in\mathcal{D}$. $y_i\in\{0,1\}$ indicates whether the facility i is opened, $x_{ij}\in\{0,1\}$ represents whether client j is connected to facility i and $r_j\in\{0,1\}$ denotes whether or not client j is connected to a facility.

We will relax the indicator variables to the nonnegative real space, leading to the standard LP-P relaxation as follows:

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

The set of constraints (1) represents each client should be connected to at least one facility or be punished, the set of constraints (2) says that clients can only be connected to opened facilities, and the set of constraints (3) illustrates that at most k facilities can be opened.

In order to solve the difficulty caused by the constraint (3) to solve LP-P, we adopt the Lagrangian multiplier method [10], which introduces the Lagrangian multipliers λ for each constraint in (3), resulting in a new linear program LP-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 LP-P and LP-P(λ) respectively. Then we have $\mathrm{OPT}_{\mathrm{L}}(\lambda) \leq \mathrm{OPT}_k$ for $\lambda \geq 0$. In the following we give the dual programming DUAL-P(λ) of LP-P(λ).

$$\begin{aligned}
\max \quad & \sum_{j\in\mathcal{D}} \alpha_j - \lambda \cdot k \\
\text{s.t} \quad & \sum_{j\in\mathcal{D}} [\alpha_j - c(j,i)]^+ \leq \lambda && \forall i \in \mathcal{F} && (8)\\
& \alpha_j \leq p_j && \forall j \in \mathcal{D} && (9)\\
& \alpha_j \geq 0 && \forall j \in \mathcal{D}, && (10)
\end{aligned}$$

where $[\alpha_j - c(j,i)]^+ = \max\{\alpha_j - c(j,i), 0\}$. Let $\mathrm{OPT}_{\mathrm{D}}(\lambda)$ denote the optimal values of DUAL-P(λ). Hence we have $\mathrm{OPT}_{\mathrm{D}}(\lambda) \leq \mathrm{OPT}_{\mathrm{L}}(\lambda) \leq \mathrm{OPT}_k$ for $\lambda \geq 0$.

Observing LP-P(λ) and DUAL-P(λ), we find that if the $\lambda \cdot k$ constant term in the objective function is ignored, they are the standard linear programming relaxation and the dual of the uniform facility location problem with penalty, in which the facility opening cost is λ. Therefore, we can give the improved JV-P(δ) algorithm to solve DUAL-P(λ) and get a constant approximate ratio of $6,357+\varepsilon$. If we can search a suitable λ in the polynomial so that the number of facilities opened is not greater than k, we will look for a feasible solution to the original problem, where the target value corresponding to this solution does not exceed the optimal value OPT_k of the original problem $6,357+\varepsilon$ times.

3 The JV-P(δ) algorithm and the main conclusions

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

JV-P(δ) is a modified primal-dual algorithm based on the JV-P algorithm in [4] to solve the facility location problem with penalties where the opening cost of each facility is equal to λ. This algorithm contains two phases: the dual-growth phase (Algorithm 1) and the pruning phase (Algorithm 2).

In the dual-growth phase, we construct a feasible dual solution (α, β) for DUAL-P(λ).

Algorithm 1 JV-P(δ): The dual-growth phase

1. Initially, let $\alpha := 0$ and set $\beta_{ij} := [\alpha_j - c(i,j)]^+ = \max\{\alpha_j - c(i,j), 0\}$. Let $A := D$ denote the set of active clients and let θ represent time. Starting from zero, $\forall j \in A$, α_j and θ rise at the same time per unit speed. When $\alpha_j = p_j$, we say that j is frozen, and α_j does not increased. If one of the following events occurs, j will be removed from A.
   - **Event 1.** A dual constraint $\sum_{j\in\mathcal{D}}[\alpha_j - c(j,i)]^+ = \lambda$ for a facility $i \in \mathcal{F}$. In this case, we say that facility i is tight or temporarily opened. We update A by removing $j \in A$ if $\alpha_j \geq c(i,j)$. We say that facility i is the witness of these removed clients, denoted as $w(j) = i$.
   - **Event 2.** An active client $j \in A$ gets a tight edge, i.e., $\alpha_j - c(j,i) = 0$, to some already tight facility i. In this case, we remove j from A and let i be its witness.
2. When $A = \emptyset$ or j is frozen $\forall j \in A$, the dual-growth phase stops.

After the dual-growth phase, we obtain a set of temporarily opened facilities $\mathcal{F}_y$, in which there may be a client j contributing to the opening of multiple facilities, that is $\{i \in \mathcal{F}_y : \alpha_j > c(j,i)\}$. These facilities are relatively close, and the opening cost of these facilities can be reduced by closing some of them. For this, we introduce the pruning phase, where it will select a subset of temporarily opened facilities to open.

We first construct the client-facility graph G and the conflict graph H. Before that, we introduce some notations.

1. $\mathcal{F}_y$: the set of tight facilities during the dual-growth phase.

2. $N(j) := \{i \in \mathcal{F} : \alpha_j - c(j,i) > 0\}$, $\forall j \in \mathcal{D}$.
3. $N(i) := \{j \in \mathcal{D} : \alpha_j - c(j,i) > 0\}$, $\forall i \in \mathcal{F}$.
4. For a tight facility i, we let $t_i := \max_{j \in N(i)} \alpha_j$, and $t_i := 0$ if $N(i) = \emptyset$. Hence, we have $t_i \geq \alpha_j$, $\forall j \in N(i)$ and for a client j and its witness $w(j)$, $\alpha_j \geq min\{t_{w(j)}, p_j\}$.

– the client-facility graph $G = (\mathcal{D} \cup \mathcal{F}_y, E)$: if $i \in N(j)$, then we have $(j,i) \in E$.

– the conflict graph $H = (\mathcal{F}_y, E)$: if $j \in N(i) \cap N(i^{'})$ and $c(i,i^{'}) \leq \delta \min(t_i, t_{i^{'}})$, where δ is a constant, then we have $(i,i^{'}) \in E$.

Algorithm 2 JV-P(δ): The pruning phase

Find a maximal independent set IS of H and open these facilities. Define the penalized set of clients $\mathcal{D}_p$,

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

The remaining clients are connected to the closest facility in IS.

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

Lemma 1. *Given an instant I of DKPP, we assume that d is a Euclidean metric on $\mathcal{D} \cup \mathcal{F}$. Then, for any $\lambda \geq 0$, Algorithm 1 constructs a solution α to DUAL-P(λ) and Algorithm 2 outputs a set IS of opened facilities and a set $\mathcal{D}_p$ of punished clients such that*

$$\sum_{j \in \mathcal{D} \setminus \mathcal{D}_p} d(j, IS)^2 + \sum_{j \in \mathcal{D}_p} p_j \leq \rho \cdot \left(\sum_{j \in \mathcal{D}} \alpha_j - \lambda |IS| \right), \tag{11}$$

where $\delta \approx 2.315$ and $\rho \approx 6.357$.

Proof. From Algorithm 1, for any $\lambda \geq 0$, α is a feasible solution to DUAL-P(λ). Now we just need to prove (11). If we have

$$\begin{aligned} \frac{c(j, \text{IS})}{\rho} &\leq \alpha_j - \sum_{i \in N(j) \cap \text{IS}} (\alpha_j - c(j,i)) \\ &= \alpha_j - \sum_{i \in \text{IS}} [\alpha_j - c(j,i)]^+, \quad \forall j \in \mathcal{D} \setminus \mathcal{D}_p, \qquad (12) \\ \frac{p_j}{\rho} &\leq \alpha_j - \sum_{i \in \text{IS}} [\alpha_j - c(j,i)]^+, \quad \forall j \in \mathcal{D}_p. \qquad (13) \end{aligned}$$

Then by adding up all clients and $[\alpha_j - c(j,i)]^+ \geq 0$ for any $i \in \mathrm{IS}, j \in \mathcal{D}_p$, we have

$$\frac{1}{\rho}\left(\sum_{j\in\mathcal{D}\setminus\mathcal{D}_p} d(j,\mathrm{IS})^2 + \sum_{j\in\mathcal{D}_p} p_j\right) \leq \sum_{j\in\mathcal{D}} \alpha_j - \sum_{j\in\mathcal{D}\setminus\mathcal{D}_p}\sum_{i\in\mathrm{IS}} [\alpha_j - c(j,i)]^+.$$

Moreover, Note that any facility in IS is opened and no client in $\mathcal{D}_p$ contributes positively to any facility in IS. So we obtain that $\sum_{j\in\mathcal{D}\setminus\mathcal{D}_p}[\alpha_j - c(j,i)]^+ = \lambda$, for any $i \in \mathrm{IS}$. Therefore, (11) is satisfied.

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

We also let $\beta_{ij} = [\alpha_j - d(j,i)^2]^+$ and similarly $\beta_{ij}^{\lambda} = [\alpha_j^{\lambda} - d(j,i)^2]^+$ and $\beta_{ij}^{\lambda+\varepsilon_z} = [\alpha_j^{\lambda+\varepsilon_z} - d(j,i)^2]^+$. Now, We will discuss each case separately.

Case 1. $s_j = 0$. $\forall j \in \mathcal{D}_0$, there exists a $w(j) \in \mathcal{F}$ such that $\alpha_j \geq t_{w(j)}$, and $\alpha_j \geq c(j,w(j))$. Furthermore, if $w(j) \in \mathrm{IS}$, then $d(j,\mathrm{IS}) \leq d(j,w(j))$; otherwise, there exists an $i \in \mathcal{F}$ such that $c(w(j),i) \leq \delta \min\{t_{w(j)}, t_i\} \leq \delta\alpha_j$, in which case

$$d(j,\mathrm{IS}) \leq d(j,i) \leq d(w(j),j) + d(w(j),i) \leq \sqrt{\alpha_j} + \sqrt{\delta\alpha_j} = (1+\sqrt{\delta})\sqrt{\alpha_j}.$$

Moreover, $c(j,\mathrm{IS}) \leq (1+\sqrt{\delta})^2\alpha_j$ and $\sum_{i\in\mathrm{IS}}[\alpha_j - c(j,\mathrm{IS})]^+ = 0$ together imply that

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

When $\rho \geq (1+\sqrt{\delta})^2$, we have

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

Case 2. $s_j = 1$. $\forall j \in \mathcal{D}_1$, there is only one facility $i^* \in \mathrm{IS}$ such that $[\alpha_j - c(j,i^*)]^+ > 0$. So, we have

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

Case 3. $s_j > 1$. In an ℓ-dimensional metric space, the following holds: $\forall j \in \mathcal{D}_{>1}$,

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

In the above, $\mu = \frac{1}{s_j}\sum_{i\in S_j} i$ is the centroid point of the facility in S_j, and the fourth inequality follows from $c(i,i^{'}) > \delta \min\{t_i, t_{i'}\} \geq \delta\alpha_j$, $\forall i \in S_j$. Hence,

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

For $s_j > 1$, $s_j(1-\frac{\delta}{2})+\frac{\delta}{2}$ is a non-increasing function of s_j because $\delta \geq 2$. We also know that $c(j,\mathrm{IS}) \leq c(j,i) \leq \alpha_j$, $\forall i \in S_j$.
So,

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

When

$$\rho \geq \frac{1}{\frac{\delta}{2}-1},$$

we have

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

Combining all the above cases, let

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

where $\delta \geq 2$. Setting $\delta \approx 2.315$, we have $\rho \approx 6.357$, and hence (12) follows.

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

As the algorithm shows, unless some information about λ is given, $|\mathrm{IS}| \leq k$ may not be guaranteed. When λ is small, the number of facilities in IS tends to $|F|$. When λ is large, the number of facilities in IS approaches 1. Hence, we hope to find a suitable λ to make $|\mathrm{IS}| \leq k$. Similar to that in [1], we give a pseudo-polynomial algorithm for DPKP by enumerating λ and finding a k-cardinality solution, such that the obtained solution satisfies the constraint that at most k facilities are opened and Lemma 1.

4 The pseudo-polynomial algorithm as well as our main results

In [1], by enumerating λ, a (6.357+ε)-approximation algorithm of pseudo-polynomial time is obtained for the $k-$means problem. Based on the same idea, we presnet a (6.357+ε)-pseudo polynomial time approximation algorithm for the $k-$means

problem with penalty. For convenience, we abbreviate this algorithm as PPTA, which contains two sub-algorithms: Eλ and Ck.

Eλ enumerates all the possibilities of $\lambda = 0, 1 \cdot \varepsilon_z, \cdots, L \cdot \varepsilon_z$, in which ε_z is a small step size and L is large.

$$n \gg 1/\varepsilon, \varepsilon_z = n^{-3-6log_{1+\varepsilon}P}, L = 4n^7 \cdot \varepsilon_z^{-1}, P = \max_{j\in\mathcal{D}}\{p_j\}.$$

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

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

At the same time, it can be guaranteed that when $\lambda = L \cdot \varepsilon_z$, $|\mathrm{IS}| \leq 1$. In order to simplify the algorithm, we preprocess any instance of DPKP, 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 [1].

Algorithm 3 Eλ

1. Initially, $\lambda = 0$ and set $\alpha_j^{in} = \min\{\min_{i\in\mathcal{F}} d(j,i)^2, p_j\}$, and $\mathrm{IS} = \mathcal{F}$.
2. $\alpha = \alpha^{in}$, $\lambda = \lambda + \varepsilon_z$, $A = \emptyset$. θ indicates time. Starting from zero, when $\theta = \alpha_j$,$\forall j \in \mathcal{D}$, let $A = A \cup j$. α_j and θ increases at the same rate. When $\alpha_j = p_j$, we say that j is frozen and α_j is not increased. If the following event occurs, j is removed from A.
   Event: j has a tight edge to a tight facility i with $B(\alpha_j) \geq B(t_i)$. Let i be the witness of j, denoted as $w(j) = i$.
3. While Step 2 is happening, we decrease α_j with $B(\alpha_j) > B(\theta)$ by $|A|$ times the growth rate of θ.
4. When $A = \emptyset$ or any client in A is frozen, output $\alpha^{out} = \alpha$
5. When $\lambda = L \cdot \varepsilon_z$, the algorithm stops; Otherwise $\alpha^{in} = \alpha^{out}$, and go back to Step 2.

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

Lemma 2. *[1] Any two 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. In the above assumption, we have $1 \leq d(j,i)^2 \leq n^6$ and $P = max_{j\in\mathcal{D}}\{p_j\}$. We notice that at any time the largest $\alpha-$value is at most

$$\begin{aligned}\min\{\max\{\lambda\} + n^6, P\} &= \min\{L\cdot\varepsilon_z + n^6, P\}\\ &= \min\{4n^7 + n^6, P\} \leq \min\{5n^7, P\}\\ &\leq P.\end{aligned}$$

Hence, $B(\alpha_j) \leq 1+\lfloor\log_{1+\varepsilon} P\rfloor \leq 2\log_{1+\varepsilon} P$ for any client j and the dual solution α. Similar to [1], we can still get the following claim.

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

Proof. We prove it by induction on $b = 0, 1, \cdots, 2\log_{1+\varepsilon} P$.

$b = 0$**.** It is trivially true that there is no clients' $\alpha-$value increases or decreases with $B(\alpha_j) = 0$.

$b > 0$**.** We now assume that α_j can increase by at most $\varepsilon_z n^{3b-3}$ when $B(\theta) \leq b-1$ is correct, and 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 in the case of $B(\theta) < B(\alpha_j)$, 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 was decreased 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}. 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 $j^{'} \in N^{in}(i)$ was decreased by Algorithm 3 only while $B(\theta) < b$ and $\alpha_{j'}$ was 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 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 $\varepsilon_z = n^{-3-6log_{1+\varepsilon}P}$. 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 G^{λ}, H^{λ} and IS^{λ}, following the same method as in Section 2 (See Algorithm 4 for specific operations.).

Through the above analysis and the value of L, we know that when $\lambda = 0$, $|\mathrm{IS}| = n$, and when $\lambda = L\cdot\varepsilon_z$, $|\mathrm{IS}| = 1$. 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.

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

1. Initially, $\lambda = 0$.
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, $H^{\lambda+\varepsilon_z}$, IS^λ.
3. Construct a client-facility graph $G^{(\lambda,1)}$ with bipartition $\mathcal{D}$ and $\mathcal{F}_y^\lambda \cup \mathcal{F}_y^{\lambda+\varepsilon_z}$ that has an edge from client j to facility $i \in \mathcal{F}_y^\lambda$ if (j,i) is present in G^λ and to $i \in \mathcal{F}_y^{\lambda+\varepsilon_z}$ if (j,i) is present in $\mathrm{G}^{\lambda+\varepsilon_z}$. The opening time t_i of facility i is now naturally 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. Generate the conflict graph $H^{(\lambda,1)}$ from $G^{(\lambda,1)}$ and t, and output a maximal independent set $\mathrm{IS}^{(\lambda,1)}$ of $H^{(\lambda,1)}$ by greedily extending IS^λ.
5. Let $p = 2$.
   while $p \neq |F_y^\lambda| + 2$ **do**
   Removing a facility $i \in \mathcal{F}_y^\lambda$ from $G^{(\lambda,1)}$, we construct and output a new conflict graph $H^{(\lambda,p)}$ from $G^{(\lambda,p)}$ and its maximal independent set $\mathrm{IS}^{(\lambda,p)}$ by greedily extending $\mathrm{IS}^{(\lambda,p-1)} \backslash \{i\}$.
   if $|\mathrm{IS}^{(\lambda,p)}| = k$, **then**
   stop, and output $\mathrm{IS}^{(\lambda,p)}$.
   else
   $p = p + 1$.
   end if
   end while
6. At the end of the procedure (after $|F_y^\lambda|$ many steps), we have $G^{\lambda+\varepsilon_z} = G^{(\lambda,|F_y^\lambda|+1)}$. Set $H^{\lambda+\varepsilon_z} = H^{(\lambda,|F_y^\lambda|+1)}$, $\mathrm{IS}^{\lambda+\varepsilon_z} = \mathrm{IS}^{(\lambda,|F_y^\lambda|+1)}$ and $\lambda = \lambda + \varepsilon_z$. Go to Step 2.

4.1 Approximate ratio analysis

Without loss of generality, we assume $\mathrm{IS} = \mathrm{IS}^{(\lambda,l)}$, $G = G^{(\lambda,l)}$ and $H = H^{(\lambda,l)}$, where $\mathrm{IS}^{(\lambda,l)}$ is the solution output by the algorithm 4, $G^{(\lambda,l)}$ and $H^{(\lambda,l)}$ are corresponding the client-facility graph and conflict graph. Now, we give the definition of the client set $\mathcal{D}_p$ that is ultimately penalized,

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

where $j \notin N(i)$ for each $i \in \mathrm{IS}$ 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 clients connect to the nearest facility in IS.

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 DUAL-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, IS satisfies the following lemma and theorem.

For each client j, we denote S_j as the set of facilities that j contributes in IS, namely $S_j := \{i \in \mathrm{IS} : \alpha_j > c(j,i)\}$. 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 $\beta_{ij} = [\alpha_j - d(j,i)^2]^+$ and similarly $\beta_{ij}^{\lambda} = [\alpha_j^{\lambda} - d(j,i)^2]^+$ and $\beta_{ij}^{\lambda+\varepsilon_z} = [\alpha_j^{\lambda+\varepsilon_z} - d(j,i)^2]^+$. Now, We discuss each case separately.

Lemma 3. *For any $j \in \mathcal{D}_{>0}$, we have $d(j,IS)^2 \leq \rho \cdot (\alpha_j - \sum_{i\in S_j} \beta_{ij})$.*

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

Lemma 4. *For every $j \in \mathcal{D}_0$, we have $d(j,IS)^2 \leq (1+5\varepsilon)\rho \cdot \alpha_j$.*

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

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

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

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

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

If $w(j) \in$IS, $d(j,$IS$) \leq d(j,w(j))$; otherwise, there is an $i \in$ IS such that $c(w(j),i) \leq \delta \cdot t_{w(j)}^{\lambda+\varepsilon_z}$. So

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

Hence

$$\frac{c(j,\text{IS})}{(1+\sqrt{\delta})^2} \leq (1+2\varepsilon)^2\alpha_j \leq (1+5\varepsilon)\alpha_j.$$

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

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

Lemma 5. *For every $j \in \mathcal{D}_p$ and $i \in IS$, we have $p_j = \alpha_j$ and $\beta_{ij} = 0$.*

Lemma 6. *For any $i \in IS$, we have $\sum_{j\in\mathcal{D}\backslash\mathcal{D}_p} \beta_{ij} \geq \lambda - \frac{1}{n}$.*

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

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

The second inequality follows because for any $i \in$ IS, if $i \in \mathcal{F}_y^{\lambda+\varepsilon_z}$, $\sum_{j\in\mathcal{D}\backslash\mathcal{D}_p} \beta_{ij}^{\lambda+\varepsilon_z} = \lambda + \varepsilon_z$; otherwise, $i \in \mathcal{F}_y^{\lambda}$, $\sum_{j\in\mathcal{D}\backslash\mathcal{D}_p} \beta_{ij}^{\lambda+\varepsilon_z} = \lambda$. □

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

$$\sum_{j\in\mathcal{D}\setminus\mathcal{D}_p} d(j,IS)^2 + \sum_{j\in\mathcal{D}_p} p_j \le (\rho + O(\varepsilon))\cdot OPT_k,$$

where $\rho \approx 6.357$.

Proof. By Lemmas 3-5, we get

$$\sum_{j\in\mathcal{D}\setminus\mathcal{D}_p} d(j,\mathrm{IS})^2 \le (1+5\varepsilon)\rho \sum_{j\in\mathcal{D}\setminus\mathcal{D}_p}\left(\alpha_j - \sum_{i\in S_j}\beta_{ij}\right),$$
$$\sum_{j\in\mathcal{D}_p} p_j \le (1+5\varepsilon)\rho \sum_{j\in\mathcal{D}_p}\alpha_j.$$

Therefore,

$$\sum_{j\in\mathcal{D}\setminus\mathcal{D}_p} d(j,\mathrm{IS})^2 + \sum_{j\in\mathcal{D}_p} p_j \le (1+5\varepsilon)\rho\left(\sum_{j\in\mathcal{D}}\alpha_j - \sum_{j\in\mathcal{D}\setminus\mathcal{D}_p}\sum_{i\in S_j}\beta_{ij}\right).$$

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

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

The penultimate inequality stems from

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

4.2 Algorithm running time analysis

Note that in Algorithm 3: E(λ), the total number of iteration is at most L, and the running time in each iteration is at most $O(m)$, where $m = |\mathcal{F}|$. Thus the running time of E(λ) is $n^{O(\varepsilon^{-1}\log P)}$. 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 $n = |\mathcal{D}|$. Hence the running time of Ck is $n^{O(\varepsilon^{-1}\log P)}$. So the total running time is also $n^{O(\varepsilon^{-1}\log P)}$.

According to the P value, we divide $n^{O(\varepsilon^{-1}\log P)}$ into two categories. For the general case, P is a polynomial about n. Then $n^{O(\varepsilon^{-1}\log P)} = n^{O(\varepsilon^{-1}\log n)}$, which implies a pseudo-polynomial time algorithm. When P is a constant value, $n^{O(\varepsilon^{-1}\log P)} = n^{O(\varepsilon^{-1})}$, which implies a polynomial time algorithm.

5 Discussion

In this article, we study the k-means problem with penalties. In the Euclidean space, when the penalty value of each client is general, a quasi-polynomial time (6.357+ε)-approximation algorithm is given via the primal-dual technique. When the maximum penalty value P is a constant value, the pseudo-polynomial time can be transformed into polynomial time. We note that the same algorithm can be used to obtain an approximate algorithm with ratio of $2.633 + \varepsilon$ for the penalized k-median problem in the Euclidean space.

We believe that our algorithm can be improved to obtain a polynomial time algorithm with the same approximate ratio. Intuitively, we only need to transform ε_z into polynomial time. Inspired by [1], we 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 with 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).

References

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

2. Aloise D, Deshpande A, Hansen P, Popat P. NP-hardness of Euclidean sum-of-squares clustering. Machine Learning, 2009, 75: 245-248.
3. Arthur D, Vassilvitskii S. k-means++: the advantages of careful seeding. In: Proceedings of SODA, 2007, pp. 1027-1035.
4. Charikar, M, Khuller, S, Mount, D.M, Narasimhan, G. Algorithms for facility location problems with outliers. In: Proceedings of 12th ACM-SIAM Symposium on Discrete Algorithms. 2001, pp: 642-651
5. Cohen-Addad V, Klein P N, Mathieu C. Local search yields approximation schemes for k-means and k-median in Euclidean and minor-free metrics. SIAM Journal on Computing, 2019, 48: 644-667.
6. Drineas P, Frieze A, Kannan R, Vempala S, Vinay, V. Clustering large graphs via the singular value decomposition. Machine Learning, 2004, 56: 9-33.
7. Feldman D, Monemizadeh M, Sohler C. A PTAS for $k-$means clustering based on weak coresets. In: Proceedings of the twenty-third annual symposium on Computational geometry. 2007: 11-18.
8. Feng Q, Zhang Z, Shi F, Wang J. An improved approximation algorithm for the k-means problem with penalties. In: Proceedings of FAW, 2019, pp. 170-181.
9. Friggstad Z, Rezapour M, Salavatipour M R. Local search yields a PTAS for k-means in doubling metrics. SIAM Journal on Computing, 2019, 48: 452-480.
10. Jain K, Vazirani V V. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. Journal of the ACM, 2001, 48: 274-296.
11. Kanungo T, Mount D, Netanyahu N, Piatko C, Silverma R. A local search approximation algorithm for k-means clustering. Computational Geometry, 2004, 28: 89-112.
12. Li M. The bi-criteria seeding algorithms for two variants of k-means problem. Journal of Combinatorial Optimization, 2020, DOI: 10.1007/s10878-020-00537-9.
13. Li M, Xu D, Yue J, Zhang D, Zhang P. The seeding algorithm for k-means problem with penalties. Journal of Combinatorial Optimization, 2020, 39: 15-32.
14. Li, Y, Du, D, Xiu, N, Xu, D. Improved approximation algorithms for the facility location problems with linear/submodular penalties. Algorithmica, 2015, 73: 460-482.
15. Lloyd S. Least squares quantization in PCM. IEEE Transactions on Information Theory, 1982, 28: 129-137.
16. Makarychev K, Makarychev Y, Razenshteyn I. Performance of Johnson-Lindenstrauss transform for k-means and k-medians clustering. In: Proceedings of STOC, 2019, pp. 1027-1038.
17. Vazirani V V. Approximation Algorithms. Springer-Verlag Berlin, Heidelberg, 2001.
18. Williamson D P, Shmoys D B. The Design of Approximation Algorithms. Cambridge University Press, 2011.
19. Zhang D, Hao C, Wu C, Xu D, Zhang Z. Local search approximation algorithms for the k-means problem with penalties. Journal of Combinatorial Optimization, 2019, 37: 439-453.