APPROXIMATION ALGORITHM FOR SPHERICAL $k$-MEANS PROBLEM WITH PENALTY

CHENCHEN WU
School of Business, Nankai University, Tianjin, 300071, China
College of Science, Tianjin University of Technology, Tianjin, 300384, China

WEI LV*
Computer Science and Technology Department
Tianjin Renai College, Tianjin, 306136, China

YUJIE WANG
Department of Operations Research and Information Engineering
Beijing University of Technology, Beijing, 100124, China

DACHUAN XU
Department of Operations Research and Information Engineering
Beijing University of Technology, Beijing, 100124, China

(Communicated by Wenxun Xing)

Abstract. The $k$-means problem is a classical combinatorial optimization problem which has lots of applications in many fields such as machine learning, data mining, etc. We consider a variant of $k$-means problem in the spherical space, that is, spherical $k$-means problem with penalties. In the problem, it is allowable that some nodes in the spherical space can not be clustered by paying some penalty costs. Based on local search scheme, we propose a $\left(4(11 + 4\sqrt{7}) + \epsilon\right)$-approximation algorithm using single-swap operation, where $\epsilon$ is a positive constant.

1. Introduction. In the real world, it is important to cluster the data with similar structure into several parts. This leads to clustering problem. The $k$-means problem is one of the most widely used problems (c.f. [9, 10]). In the $k$-means problem, we are given a set of discrete nodes $\mathcal{N}$ in continuous $d$-dimension space $\mathcal{R}^d$ (c.f. Figure 1). The cost of each pair of nodes in $\mathcal{R}^d$ is measured by the distance between them in $\mathcal{R}^d$. We shall choose $k$ centers in $\mathcal{R}^d$ such that the total cost from each node in $\mathcal{N}$ to the closest center is minimized. That is,

$$\min_{C \subseteq \mathcal{R}^d : |C| \leq k} \sum_{j \in \mathcal{N}} \min_{i \in C} c(i, j),$$

where $c(i, j)$ is the cost between the nodes $i$ and $j$.

The $k$-means problem has lots of applications in many fields such as machine learning, data mining, etc, that is NP-hard [2]. One of the efficient methods for NP-hard problems is to design a heuristic algorithm. Lloyd’s algorithm [15] is the
most creative achievement which is used in various engineer fields (c.f. [13, 16]). It alternately finds centers and clusters in the algorithm. The algorithm is one of the TOP 10 algorithms in data mining [18]. Although the algorithm is successfully applied to the field of engineering, the theoretical worst-case performance guarantee (i.e. approximation ratio) of the algorithm is unbounded.

One way to obtain an approximation algorithm for $k$-means problem is a kind of seeding algorithm which chooses appropriate initial solution in Lloyd’s algorithm. Arthur and Vassilvitskii [4] obtain a $O(\ln k)$-approximation algorithm by finding an efficient initial solution in Lloyd’s algorithm. This kind of algorithm is called $k$-means++. Another scheme to obtain approximation algorithm for the $k$-means problem is local search. Based on the scheme, Kanungo et al. [12] propose a constant approximation algorithm for the problem. Moreover, Ahmadian et al. [3] give a $6.375$-approximation algorithm based on primal-dual scheme. Although the algorithm is complex, the approximation ratio is the best until now. There are also some works on the bi-criteria, that is, it is allowable to violate the upper bound of the number of the centers. Wei [17] gives an approximation algorithm by violating the upper bound of $k$ based on seeding algorithm.

There are also some interesting extensions for the $k$-means problem such as $k$-means with penalties (c.f. [7, 8, 13]). In the $k$-means problem, all data should be divided into some group. Actually, some data are far from the others such as, the node in the lower right corner in Figure 1. This leads to a penalty version. In the $k$-means problem with penalties, it is allowable that some nodes are not clustered by paying some penalty costs. Zhang et al. [20] propose a $(81 + \epsilon)$-approximation algorithm based on local search scheme which is further improved to $(25 + \epsilon)$, where $\epsilon$ is a positive constant. Feng et al. [6] give a better $(19.849 + \epsilon)$-approximation algorithm. Li et al. [14] propose a seeding algorithm for the problem.

Figure 1. $k$-means problem
In the classical k-means problem, all nodes are all in $\mathbb{R}^d$ (c.f. Figure 1). In fact, the nodes needed to be clustered are on the sphere in some practical problems. This induces an extension of the k-means problem, spherical k-means problem. In this problem, all nodes in $\mathcal{N}$ are on the sphere with unit radius (Figure 5). Zhang et al. [19] obtain a $(2(4 + \sqrt{7}) + \epsilon)$-approximation algorithm based on the local search scheme. Ji et al. [11] consider spherical k-means problem with penalties by seeding algorithm in which there are penalty costs based on the spherical k-means problem.

In this work, we study spherical k-means problem with penalties using local search technique. Combining the special structure and the technique in [19], we obtain $(4(11 + \sqrt{7}) + \epsilon)$-approximation algorithm for the problem. In the rest of the paper, we introduce the problem in detail in Section 2. Then, we give the standard local search algorithm in Section 3. In Section 4. we analyze the single-swap algorithm. At last, we will propose some conclusions in Section 5.

2. Preliminary. In the spherical k-means problem with penalties (Sph-k-MP), we are given a node set $\mathcal{N} = \{j_1, j_2, \cdots, j_n\} \subseteq \mathcal{S}^d$, where $\mathcal{S}^d$ is a unit sphere in $d$-dimension space, that is, $\mathcal{S}^d = \{x \in \mathbb{R}^d : \|x\| = 1\}$. Each pair nodes $a, b \in \mathcal{S}^d$, let $d(a, b)$ denote the spherical distance between them. Also, the distance $d(a, b)$ is the cost between $a$ and $b$. There is a penalty cost $p_j$ for each node $j \in \mathcal{N}$. We need to choose at most $k$ center in $\mathcal{S}^d$ and a penalized subset of $\mathcal{N}$ such that the total cost is minimized. The total cost includes the cost for the un-penalized nodes to the corresponding closest center and the penalty cost for the penalized nodes. Thus, we have

$$\min_{P \subseteq \mathcal{N}, S \subseteq \mathcal{S}^d, |S| \leq k} \sum_{j \in P} p_j + \sum_{j \in \mathcal{N}\setminus P} d(j, S),$$

where $d(j, S) := \min_{c \in S} d(j, c)$. Actually, we have $d(j, i) = 1 - c(j, i)$ where $i, j$ are two distinct nodes in $\mathcal{S}^d$.

Note that the distance in the spherical space $\mathcal{S}^d$ is different with the distance in the Euclid space $\mathbb{R}^d$. Though $d(a, b)$ for each $a, b \in \mathcal{S}^d$ is positive and symmetric,
the triangle inequalities do not hold anymore. Indeed,\[ d(a, b) = 1 - c(a, b) = 1 - \frac{a^T b}{\|a\|\|b\|} = 1 - a^T b = 1 - \frac{1}{2} (\|a\|^2 + \|b\|^2 - a^T b) = 1 - 2\|a - b\|^2. \] (1)
The third equality holds since $\|a\| = \|b\| = 1$. Then, for each $a, b, c \in S^d$, the following holds.\[ \sqrt{d(a, c)} \leq \sqrt{d(a, b)} + \sqrt{d(b, c)}. \] (2)

3. **Single-swap algorithm.** In this section, we will give an approximation algorithm for the Sph-$k$-MP based on a single-swap operation. Since the penalized set can be ascertained if the center set is given, we just use the center set to represent the corresponding solution. Let $\text{Cost}(S)$ be the total cost if the center set is $S$. Under the framework of the local search technique, we need to choose a feasible solution for the Sph-$k$-MP firstly. Then, we swap a center in the current solution and a node not in the current solution. In the discrete space, the procedure is well-defined. However, in the continuous space, we need to choose a node in $S^d$ to replace a center in the current solution. There are infinite nodes in $\mathbb{R}^d$. Thus, it is necessary to find a discrete alternate nodes in the framework of the local search framework. In [19], they use the $N$ as the discrete candidates. We follow this idea, and obtain our algorithm.

**Algorithm 1:**

Step 0 Initially, we arbitrarily choose a subset of $N$ with $k$ elements which is denote as $S_{cur}$.

Step 1 Find the minimum cost of the solution $S'$ in\[ \{S_{cur} \setminus \{s\} \cup \{j\} : s \in S_{cur}, j \in N \setminus S_{cur}\}. \]
That is,\[ S' := \arg \min_{s \in S_{cur}, j \in N \setminus S_{cur}} \text{Cost}(S_{cur} \setminus \{s\} \cup \{j\}). \]

Step 2 If $\text{Cost}(S') \geq \text{Cost}(S_{cur})$, output local optimal solution $S := S_{cur}$. Otherwise, update $S_{cur} := S'$ and go back Step 1.

In the above algorithm, we use the framework of local search technique based on swapping a pair of nodes (see Figure 3). We use a node in $N$ to replace a node in the current solution. When the solution can not be improved anymore, the algorithm terminates. In fact, the condition of $\text{Cost}(S') \geq \text{Cost}(S_{cur})$ in Step 2 needs to be changed as $\text{Cost}(S') - \text{Cost}(S_{cur}) > \epsilon$ to guarantee the polynomial time of the algorithm, where $\epsilon$ is a positive constant. This will lead to an additional $\epsilon$ in approximation ratio. For easy to understand, we omit $\epsilon$ in the procedure of the proofs.
4. Analysis. Though the algorithm is standard, we need to analyze the efficiency of the algorithm for the problem of the penalty version. For this, we define some notation for the analysis.

- **S**: the local optimal solution.
- **O**: the global optimal solution.
- **P**: the penalized node set in local optimal solution, that is, \( P = \{ j \in \mathcal{N}, p_j < d(j, S) \} \).
- **P^***: the penalized node set in global optimal solution, that is, \( P^* = \{ j \in \mathcal{N}, p_j < d(j, O) \} \).
- **N^S(s)**: the node subset of \( \mathcal{N} \) whose cluster center is \( s \) in local optimal solution, that is, \( N^S(s) = \{ j \in \mathcal{N}, d(s, j) < d(s', j), \text{ for all } s' \neq s, s' \in S \} \).
- **N^O(o)**: the node subset of \( \mathcal{N} \) whose cluster center is \( o \) in global optimal solution, that is, \( N^O(o) = \{ j \in \mathcal{N}, d(o, j) < d(o', j), \text{ for all } o' \neq o, o' \in O \} \).
- **s_j**: the center of the node \( j \) in local optimal solution, that is, \( s_j := \arg \min_{s \in S} \{ d(j, s) : s \in S \} \).
- **o_j**: the center of the node \( j \) in global optimal solution, that is, \( o_j := \arg \min_{o \in O} \{ d(j, o) : o \in O \} \).
- **C_j(S)**: the distance for \( j \) in the solution of \( S \), that is, \( C_j(S) = d(j, s_j) \).
- **C_j(O)**: the distance for \( j \) in the solution of \( O \), \( C_j(O) = d(j, o_j) \).
- **j_o**: the closest node in \( \mathcal{N} \) for the center \( o \) in \( O \), that is, \( j_o := \arg \min_{j \in \mathcal{N}} d(j, o) \).

In the analysis of local search algorithm, it is important to find appropriate operations. The solution can not improve the local optimal solution by these operations. In our analysis, there are two relative mappings (c.f. Figure 4). In fact, we find the closest \( s \) in \( S \) for each node in \( o \in O \). This is a mapping \( g \) from \( O \) to \( S \). Moreover, we find the closest node \( j_o \) in \( \mathcal{N} \) for each \( o \in O \). This is also a mapping \( h \) from \( O \) to \( \mathcal{N} \). With no loss of generality, we assume \( h(o) \neq h(o') \) for \( o \neq o' \). Then we construct a mapping \( f \) from \( \{ h(o) : o \in O \} \subseteq \mathcal{N} \) to \( S \). In the previous works on local search technique, it is always to consider the swap operation of a node in \( S \) and a node in \( O \). In our algorithm, the analysis does not work since we use the node in \( \mathcal{N} \) to replace one of the current center. We swap \( s \) in \( S \) and \( j_o \) for some \( o \) in \( O \). We abuse the notation of \( g^{-1}(.) \) to be the set of original image for mapping \( g \). We consider the following three cases.

Case 1: If \( |g^{-1}(g(o))| = 1 \) (see the first part of Figure 4), we map the node \( j_o \in \mathcal{N} \) to the corresponding \( h(o) \in S \), that is, \( f(j_o) = g(o) \).
Case 2: If $|g^{-1}(g(o))| = 2$ (see the second part of Figure 4), we find an other node $s_1$ in $S$ which is not the closest node for any $o \in O$. Then, we map the both the nodes to the node $s_1 \in S$, that is $f(j_o,j) = f(j_o,j') = s_1$ where $h(g^{-1}(g(o))) = \{j_o,j_o\}$, where we also abuse the notation $h(O')$ for $O' \subseteq O$ to denote the set of the nodes $h(o)$ for all $o \in O'$.  

Case 3: If $|g^{-1}(g(o))| > 2$ (see the third part of Figure 4), we find $l - 1$ nodes $s_1,s_2,\ldots,s_{l-1}(l = |g^{-1}(g(o))|)$ in $S$ which is not the closest node for any $o \in O$. Also, these nodes can be different with the nodes found in Case 2. Then, we map two nodes in $h(g^{-1}(g(o)))$ to the same $s_1$, the left nodes to $s_2,s_3,\ldots,s_l$ one by one. That is, $f(j_o,j) = f(j_o,j') = s_1,f(j_o,j') = s_2,\ldots,f(j_o) = s_{l-1}$, where $h(g^{-1}(g(o))) = \{j_o,j_o,\ldots,j_o\}$.

Then, we use the operation of swapping the nodes in $j_o \in \{h(o) : o \in O\}$ and $f(j_o)$ in our analysis. Note that there are some node in $S$ will be swapped twice (Case 2 and Case 3). When we swap $j_o \in \{h(o) : o \in O\}$ and $f(j_o)$, we need to estimate the cost changes induced by the swap operation. Since the candidate centers in our algorithm is not in $\mathcal{S}^d$ but in $\mathcal{N}$, we set the center of $N_{O}(o)$ is $j_o$ for each $o \in O$. Then, let $o = h^{-1}(j_o)$ and $s = f(j_o)$ for easy to understand. We consider the following cases (c.f. Figure 5).

Case I The node $j \notin N^O(o) \cup N^S(s) \setminus (P \cap P^*)$ keeps the original clustering. The cost does not change.

Case II The node $j \in P \cap P^*$ keeps to be penalize. The cost does not change.

Case III The node $j \in P \cap N^O(o)$ is clustered into the cluster with center $j_o$ which is penalized in local optimal solution. The cost changes to $d(j_o,j)$ from $p_j$.

Case IV The node $j \in N^S(s) \cap P^*$ is penalized which is clustered into the cluster with center $s$ in local optimal solution. The cost changes to $p_j$ from $d(s,j)$.

Case V The node $j \in N^O(o) \setminus P$ is clustered into the cluster with center $j_o = h(o)$. The cost changes to $d(j_o,j)$ from $d(s,j)$.

Case VI The node $j \in N^S(s) \setminus (N^O(o) \cup P^*)$ is clustered into the cluster with center $g(o_j)$. The cost changes to $d(g(o_j),j)$ from $d(s,j)$. Note that $o_j$ is the center which $j$ is assigned to and $g(o_j) \in S$ is the closest center in local optimal solution to $o_j$.

Now, we are ready to analyze Algorithm 1. From the above, we need to estimate the bound of $d(j_o,j)$ and $d(g(o_j),j)$ to obtain the approximation ratio. Thus, we
have the following two lemmas. Lemma 4.1 is similar as [19]. We also rewrite the lemma and proof for the completeness of the paper.

**Lemma 4.1.** For each node $j \in N^O(o)$, we have

$$d(j_o, j) \leq 4d(o, j).$$

**Proof.** By (2), we obtain

$$d(j_o, o) \leq \left( \sqrt{d(o, j)} + \sqrt{d(o, j_o)} \right)^2$$

$$= d(o, j) + d(o, j_o) + 2\sqrt{d(o, j)d(o, j_o)}$$

$$\leq 2(d(o, j) + d(o, j_o))$$

$$\leq 4d(o, j).$$

The third inequality holds since the average inequality. The last inequality holds since the definition of $j_o$ which is the closest node in $N$ for $o$. \qed

Next, we consider the bound for Case VI.

**Lemma 4.2.** For a node $j \in N$,

$$d(g(o_j), j) \leq 4C_j(O) + C_j(S) + 4\sqrt{C_j(S)\sqrt{C_j(O)}}.$$

**Proof.** By (2),

$$d(g(o_j), j) \leq \left( \sqrt{d(o_j, j)} + \sqrt{d(g(o_j), o_j)} \right)^2.$$ 

Then,

$$d(g(o_j), j) \leq d(o_j, j) + d(g(o_j), o_j) + 2\sqrt{d(o_j, j)d(g(o_j), o_j)}.$$ 

Since $g(o_j)$ is the closest node in $S$ for $o_j$ and $s_j \in S$, we have

$$d(g(o_j), j) \leq d(o_j, j) + d(s_j, o_j) + 2\sqrt{d(o_j, j)d(s_j, o_j)}.$$
By (2) again,
\[
d(g(a_j), j) \leq d(o_j, j) + \left(\sqrt{d(s_j, j)} + \sqrt{d(o_j, j)}\right)^2 \\
+ 2\sqrt{d(o_j, j)} \left(\sqrt{d(s_j, j)} + \sqrt{d(o_j, j)}\right).
\]
By the definition of \( C_j(O) \) and \( C_j(S) \), it is easy to see that
\[
d(g(a_j), j) \leq 4C_j(O) + C_j(S) + 4\sqrt{C_j(O)}\sqrt{C_j(S)}.
\]
\[\square\]

In our analysis, we need to swap all \( j_o \) and the corresponding \( f(j_o) \) (c.f. Figure 6). Then, we obtain our approximation ratio.

**Theorem 4.3.** There is a \((4(11 + \sqrt{7}) + \epsilon)\)-approximation algorithm for Sph-k-MP.

**Proof.** Since \( S \) is local optimal, the solution of \( S \setminus \{s\} \cup j_o \) can not improve the solution of \( S \) (set \( s = f(j_o) \) for easy).

\[
Cost(S \setminus \{s\} \cup \{j_o\}) - Cost(S) \geq 0.
\]

Analyzing Case I-VI, we have
\[
\sum_{j \in N^S(s) \cap P^*} (p_j - d(s_j, j)) + \sum_{j \in N^S(s) \setminus (N^O(o) \cup P^*)} (d(g(o_j), j) - d(s_j, j)) \\
+ \sum_{j \in N^O(o) \setminus P} (d(j_o, j) - d(s_j, j)) + \sum_{j \in N^O(o) \cap P} (d(j_o, j) - p_j) \geq 0
\]
By Lemma 4.1,
\[
\sum_{j \in N^S(s) \cap P^*} (p_j - d(s_j, j)) + \sum_{j \in N^S(s) \setminus (N^O(o) \cup P^*)} (d(g(o_j), j) - d(s_j, j)) \\
+ \sum_{j \in N^O(o) \setminus P} (4d(o_j, j) - d(s_j, j)) + \sum_{j \in N^O(o) \cap P} (4d(o_j, j) - p_j) \geq 0.
\]

Considering all pair \( j_o \) and the corresponding \( f(j_o) \in S \) for all \( o \in O \), we sum up all inequalities. Note that the are some \( s \in S \) swapped twice.

\[
2 \sum_{s \in S} \sum_{j \in N^S(s) \cap P^*} (p_j - d(s_j, j)) + 2 \sum_{s \in S} \sum_{j \in N^S(s) \setminus (N^O(o) \cup P^*)} (d(g(o_j), j) - d(s_j, j)) \\
+ \sum_{o \in O} \sum_{j \in N^O(o) \setminus P} (4d(o_j, j) - d(s_j, j)) + \sum_{o \in O} \sum_{j \in N^O(o) \cap P} (4C_j(O) - p_j) \geq 0.
\]
By Lemma 4.2, we have
\[
2 \sum_{j \in (N \setminus P) \cap P^*} (p_j - C_j(S)) \\
+ 2 \sum_{j \in (N \setminus P) \setminus P^*} \left( C_j(S) + 4C_j(O) + 4\sqrt{C_j(S)C_j(O) - C_j(S)} \right) \\
+ \sum_{j \in (N \setminus P^*) \setminus P} (4C_j(O) - C_j(S)) + \sum_{j \in (N \setminus P^*) \cap P} (4C_j(O) - p_j) \geq 0.
\]
APPROXIMATION ALGORITHM FOR SPHERICAL $k$-MEANS PROBLEM WITH PENALTY 9

Figure 6. Partition of $\mathcal{N}$

That is,

$\sum_{j \in (\mathcal{N} \setminus P) \cap P^*} p_j + 2 \sum_{j \in (\mathcal{N} \setminus P^*) \setminus P} C_j(O) + 4 \sum_{j \in (\mathcal{N} \setminus P^*) \cap P} C_j(O)$

$- \left( 2 \sum_{j \in (\mathcal{N} \setminus P) \cap P^*} C_j(S) + \sum_{j \in (\mathcal{N} \setminus P^*) \setminus P} C_j(S) + \sum_{j \in (\mathcal{N} \setminus P^*) \cap P} p_j \right)$

$+ 8 \sum_{j \in (\mathcal{N} \setminus P^*) \setminus P} \sqrt{C_j(S)C_j(O)} \geq 0.$

Then, we obtain

$2 \sum_{j \in (\mathcal{N} \setminus P) \cap P^*} p_j + 12 \sum_{j \in (\mathcal{N} \setminus P^*) \setminus P} C_j(O) + 4 \sum_{j \in (\mathcal{N} \setminus P^*) \cap P} C_j(O) + \sum_{j \in P \cap P^*} p_j$

$- \left( 2 \sum_{j \in (\mathcal{N} \setminus P) \cap P^*} C_j(S) + \sum_{j \in (\mathcal{N} \setminus P^*) \setminus P} C_j(S) + \sum_{j \in (\mathcal{N} \setminus P^*) \cap P} p_j + \sum_{j \in P \cap P^*} p_j \right)$

$+ 8 \sqrt{\sum_{j \in (\mathcal{N} \setminus P) \cap P^*} C_j(S)} \sqrt{\sum_{j \in (\mathcal{N} \setminus P) \setminus P^*} C_j(O)} \geq 0.$

It is easy to see that,

$12\text{Cost}(O) - \text{Cost}(S) + 8\sqrt{\text{Cost}(S)}\sqrt{\text{Cost}(O)} \geq 0.$

Then, we have,

$\text{Cost}(S) \leq 4(11 + \sqrt{7})\text{Cost}(O).$

Thus, Algorithm 1 is $(4(11 + \sqrt{7}) + \epsilon)$-approximation algorithm.
5. **Conclusions.** We consider an extension of the classical $k$-means problem, i.e., spherical $k$-means problem with penalties. Based on local search scheme, we give a $(4(1 + 4\sqrt{7}) + \epsilon)$-approximation algorithm, where $\epsilon$ is a positive constant. In our algorithm, we use simple sing-swap operation. In fact, the operation can be instead by multi-swap operation. This will improve the approximation ratio as $58 + \epsilon$ roughly using the simple analysis technique, where $\epsilon$ is a positive constant. It is interesting to consider $k$-means problem with penalties using primal-dual scheme in the future.

**Acknowledgments.** The first author is supported by National Natural Science Foundation of China (No. 11971349). The third author is supported by National Natural Science Foundation of China (No. 11871081) and Beijing Natural Science Foundation Project No. Z200002.

**REFERENCES**

[1] S. Ahmadian, A. Norouzi-Fard, O. Svensson and J. Ward, Better guarantees for $k$-means and Euclidean $k$-median by primal-dual algorithms, *SIAM Journal on Computing*, 49 (2019), FOCS17-97–FOCS17-156.

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

[3] S. Ahmadian, A. Norouzi-Fard, O. Svensson and J. Ward, Better guarantees for $k$-means and Euclidean $k$-median by primal-dual algorithms, *SIAM Journal on Computing*, 49 (2019), FOCS17-97–FOCS17-156.

[4] D. Arthur and S. Vassiliavitskii, $K$-means++: The advantages of careful seeding, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, 2007, 1027–1035.

[5] Y. Endo and S. Miyamoto, Spherical $k$-means++ clustering, *Proceedings of International Conference on Modeling Decisions for Artificial Intelligence*, (2015), 103–114.

[6] Q. Feng, Z. Zhang, F. Shi and J. Wang, An improved approximation algorithm for the $k$-means problem with penalties, *Proceedings of International Workshop on Frontiers in Algorithmics*, (2019), 170–181.

[7] Z. Friggstad, K. Khodamoradi, M. Rezapour and M. Salavatipour, Approximation schemes for clustering with outliers, *ACM Transactions on Algorithms*, 15 (2019), 26:1–26:26.

[8] A. Georgogiannis, Robust $k$-means: A theoretical revisit, *Proceedings of 30th Conference on Neural Information Processing Systems*, (2016), 2891–2899.

[9] J. Han, M. Kamber and J. Pei, *Data Mining: Concepts and Techniques*, Elsevier, 2012.

[10] A. K. Jain, Data clustering: 50 years beyond $k$-means, *Pattern Recognition Letters*, 31 (2010), 651–666.

[11] S. Ji, D. Xu, L. Guo, M. Li and D. Zhang, The seeding algorithm for spherical $k$-means clustering with penalties, *Journal of Combinatorial Optimization*, 2 (2020), 1–18.

[12] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman and A. Y. Wu, A local search approximation algorithm for $k$-means clustering, *Computational Geometry-Theory and Applications*, 28 (2004), 89–112.

[13] M. Li, The bi-criteria seeding algorithms for two variants of $k$-means problem, *Journal of Combinatorial Optimization*.

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

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

[16] R. Ostrovsky, Y. Rabani, L. Schulman and C. Swamy, The effectiveness of Lloyd-type methods for the $k$-means problem, *Journal of the ACM*, 59 (2012), 28:1–28:22.

[17] D. Wei, A constant-factor bi-criteria approximation guarantee for $k$-means++, *Proceedings of the Thirtieth International Conference on Neural Information Processing Systems*, (2016), 604–612.
[18] X. Wu, Y. Kumar, J. Quinlan, J. Ross Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z. H. Zhou, M. Steinbach, D. J. Hand and D. Steinberg, Top 10 algorithms in data mining, Knowledge and Information Systems, 14 (2008), 1–37.

[19] D. Zhang, Y. Cheng, M. Li, Y. Wang and D. Xu, Local search approximation algorithms for the spherical k-means problem, Proceedings of International Conference on Algorithmic Applications in Management, Springer (2019), 341–351.

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

Received July 2020; 1st revision January 2021; 2nd revision February 2021.

E-mail address: wu_chenchentjut@163.com
E-mail address: weilupao@163.com
E-mail address: wangyujie@mails.bjut.edu.cn
E-mail address: xudc@bjut.edu.cn