Applied Intelligence (2023) 53:20272–20292
https://doi.org/10.1007/s10489-023-04543-2

# A new clustering algorithm based on connectivity

Jiaqiang Wan[1] · Kesheng Zhang[1] · Zhenpeng Guo[1] · Duoqian Miao[1,2]

Accepted: 24 February 2023 / Published online: 4 April 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

**Abstract**
The $k$-median problem is the theoretical foundation of partitioning-based clustering algorithms. It was first proposed in 1964 and later it was demonstrated that $k$-median problem is an NP-hard problem in a network. To be exact, the $k$-median problem under Euclidean distance is NP hard. Fortunately, the $k$-median problem under connectivity measure is proved to be a deterministic polynomial problem in this study, and the optimal solution is solved. According to the work above, a connectivity-based clustering theory and algorithm is proposed, obtaining the theoretically optimal partition within polynomial time, and an outstanding performance in real data sets.

**Keywords** Connectivity · Clustering · $k$-means · $k$-medians

## 1 Introduction

In recent years, the field of clustering has flourished, the research enthusiasm has increased, and many excellent algorithms have emerged (such as [1, 2]). The so-called clustering refers to the division of data into multiple groups, the data objects in each group are similar to each other, and the groups represent different categories, respectively, the greater the similarity within the group, the more significant the difference between the groups, meaning a better clustering algorithm [3]. These clustering algorithms have a wide range of applications and are relevant in statistics, computational geometry, optimization, image processing, and various other fields [4–7]. In this work, we focus on partition-based clustering theories and algorithms. $K$-means algorithm is the most classical algorithm for the $k$-way partition problem ($k$-median problem or $k$-center problem). It is one of the most widely used clustering algorithms, and has spawned many well-performing variational algorithms [8]. We propose a $k$-median problem based on connectivity and a corresponding clustering model. Finally, the proposed model achieves excellent performance in our experiments.

The $k$-median problem derives from the facility location problem. Hale and Moberg [9] early proposed facility location problem in the seventeenth century. It investigates where to physically locate facilities to minimize the total cost of serving all nodes. In 1964, the location problem was studied further, and the $k$-median problem was first formulated by Hakimi [10]. The objective of the $k$-median problem is to select (at most)$k$ best servers. The $k$-servers problem is also called metric uncapacitated $k$-median problem. Concerning the at-most-$k$-servers problem, it is called capacitated $k$-median problem. Unless specified otherwise, the $k$-median problem denotes the $k$-servers problem in general.

Since the traditional $k$-median problem is subjected to triangle inequality (In a triangle, the sum of arbitrary two edges is bigger than the rest), it is an NP-hard problem [11]. In fact, connectivity can be used as a similarity metric, breaking the constraint of the triangle inequality. Based on this connectivity-based similarity, theoretically optimal partitioning can be ensured. There are few studies on connectivity-based clustering theory, but there are some studies on connectivity-based similarity. In recent clustering studies, a portion of researchers has focused on connectivity-based similarity studies. Liao et al. [12] have used dimension transformation to consider the connectivity problem in the beginning, not only explicitly considering the similarity of attribute information, but also implicitly considering the local graph structure. Guo et al. [13] improved the accuracy of local center

✉ Jiaqiang Wan
wanjiaqiang@cqut.edu.cn

1 School of Artificial Intelligence, Chongqing University of Technology, Chongqing, 401135, China

2 Department of Computer Science and Technology, Tongji University, Shanghai, 200092, China

allocation by adding connectivity information to distance calculation. Hadi [14] proposed a new method to calculate the distance between a pair of clusters in a data set, which is simple and interpretable. Geng and Tang [15] regarded the similarity of elements in data as the connectivity of vertices in undirected graphs, which enables the determination of clustering centers and the assignment of class become very simple, natural and effective.

To sum up, there are three common problems for research on Connectivity-based clustering:

1. Hyperparameters.
2. Similarity: It is hard to design and calculate connectivity-based similarity.
3. Optimal solution: Most current algorithms do not consider the minimum total cost, so they cannot obtain the theoretical optimal solution.

In the traditional $k$-median problem, Euclidean distance describes the relationship between two nodes, and each node is served by the nearest server. In fact, the closest server may be incapable of pointing the most suitable cluster in a complicated distribution (e.g., two overlapping spiral clusters). Then, the traditional $k$-median model is applied hardly to grouping irregularly-shaped clusters. The connectivity between two nodes is taken as the new cost function to overcome the influence of irregular shapes. Hence, we suggest a new cost function and propose a new k-median model with polynomial complexity. Then, the new $k$-median model and the corresponding clustering algorithm is not only suited to cope with arbitrarily clusters shaped but also ensures optimal solutions.

The significance of our work can be summarized as follows:

1. Converting the traditional $k$-median problem to a $k$-median problem in MST relies on the connectivity metric.
2. The globally optimal solution of the proposed $k$-median problem is proved and obtained within polynomial time. Then, it is guaranteed that the clustering result is optimal in theory.

The rest of the paper is organized as follows. Related work about the $k$-median problem is given in Section 2. Section 3 defines a $k$-median problem based on connectivity and gives the solution. Through the $k$-median problem based on connectivity, a novel clustering algorithm is given in Section 4. The experiments and analysis are shown in Section 5. The last part is the conclusion.

## 2 Related work

The researches on facility location problem include two categories: the $k$-center problem and the $k$-medoids problem/ $k$-median problem.

The $k$-center problem is also called the absolute location problem. It would select a location, which does not necessarily locate in those client locations, to establish a server. The objective of the $k$-center problem is to find locations for at most $k$ servers to minimize the maximal distance between a client and its nearest server [16]. It was demonstrated that the $k$-center problem is NP-hard [17]. The problem can already be solved by different methods such as heuristic algorithms [18], approximate algorithms [19], exact algorithms [20]. Among these algorithms, approximate algorithms are the most efficient and reliable. At least three known approximation algorithms are known so far: the Sh algorithm [21, 22], the Gon algorithm [23, 24], and the HS algorithm [25, 26].

The $k$-medoids partition is based on a location model (the $k$-median model) with the following general formulation [27]: Given a finite number of users, whose demands for a given service are known and must be satisfied, and given a limited set of possible locations among which $k$ must be chosen for the location of service centers, select the sites in such a way as to minimize the total distance traveled by the users. In the formulation used in clustering, the sets of users and possible locations coincide, and both correspond to the set of objects to be clustered. The location of a center is interpreted as the selection of an object as a representative object (or center type, median, or medoid) of a cluster. With respect to the $k$-median problem, the objective [16] is to find locations for at most $k$ servers to minimize the sum of distances between a client and its nearest server. Hence, the $k$-medoids problem is also the (metric uncapacitated) $k$-median problem. The $k$-median problem is an NP-hard problem [11]. That is, it is NP-hard to solve exactly in general metric spaces. In order to decrease the time complexity, many approximate schemes were proposed, such as the constant-factor approximation algorithm [28–31] , the reverse greedy algorithm [32], the approach independent of data size [33], the online-median approach [34], and the incremental approach [35]. Some scholars have also solved the complex manifold problem by setting up natural core nodes. Besides, some new applications of the $k$-median problem were proposed. For example, the $k$-median problem is applied to tree graphs [36] and graded distances [16].

Based on the $k$-centers and $k$-medoids problems, the $k$-means and $k$-medoids models were formed, and then the related clustering algorithms were proposed. MacQueen [37] first proposed a $k$-means algorithm in 1967. Kaufman [27] proposed PAM algorithm (one of the first $k$-medoids algorithms). **Since the dissimilarity measures are subjected to triangle inequality (In a triangle, the sum of arbitrary two edges is more than the rest one.), the partitioning models above are NP-hard problems**. Therefore, the related clustering algorithms are approximation schemes (e.g. [1, 2, 38, 39]), not ensuring the theoretically optimal partition. Most of the researchers believe that spectral clustering (including the two-way-cut and multiway-cut algorithms e.g. [40, 41]) is a better method owing to the solid theoretical proof—spectral theory. In fact, spectral clustering does not guarantee the quality of the solution of the relaxed problem compared with the exact resolution [42]; the partitioning strategy also uses the $k$-means algorithm. Hence, the optimal partition is also incapable to been guaranteed. In addition, JiaQiang Wan focused on connectivity and studied the transitive-closure-based **dissimilarity** measure [43], which **broke the law of triangle inequality** and could cope with data with complex manifolds. So, the $k$-median problem based on connectivity is more worthy of study.

In addition to the theoretical research above, researchers also pay much attention to dealing with arbitrary shape clusters. Partitioning methods (like K-means) have difficulties in finding clusters with non-spherical shapes. On the contrary, density-based clustering is able to discover clusters with arbitrary shapes, and the number of clusters. DBSCAN [44] is known as a classical clustering method based on density. The further research on DBSCAN focused on parameter reduction and performance optimization, such as RNN-DBSCAN [45]. In 2014, another density-based clustering algorithm DPC [46] (Clustering by Fast Search and Find of Density Peaks) was proposed. Clustering research based on density peaks is gradually increasing. LDP-MST [47] employs local density peaks to construct minimum spanning tree (MST) and then cluster in MST. According to density peaks, GADPC [48] clusters the closer nodes which have stronger graph connectivity and higher density. These methods show good performance in clustering arbitrary shape data.

In fact, there is an important problem in the density-based methods: the algorithms above are derived from rules rather than mathematical models. Hence, the optimal partition is also incapable to been guaranteed. Especially, we hope that our clustering algorithm can not only deal with arbitrary shape data, but also obtain the optimal partition. So, a new algorithm is proposed in this study.

## 3 Clustering theory based on connectivity

A $k$-median problem based on connectivity was designed and finally get a solution theorem for $k$-medians. Due to the adoption of the connectivity metric, the $k$-median problem in the fully connected graph is converted to one in MST and the corresponding proofs show that the optimal solutions of both are equivalent. The following inference gives the idea for solving the $k$-median problem based on connectivity.

### 3.1 A $k$-median problem based on connectivity

The traditional $k$-median problem (metric uncapacitated $k$-median problem.) is briefly given as follows: Firstly, build a total cost function–the sum of the costs that all nodes respectively select their nearest servers (medians) to obtain service; next, minimize the total cost function through selecting the optimal $k$ medians. There is an example shown in Fig. 1. Suppose the three nodes, $o$, $p$ and $q$ are three medians. Each node selects the respective nearest server (median) to obtain service. The nodes served by the same median form a single cluster. Then, we would get 3 clusters (see Fig. 1). According to the shapes of the 3 clusters, u should have been classified into $p's$ cluster instead of $q's$ cluster. However, the traditional $k$-median model would give the opposite result. Obviously, this partition is not unreasonable, implying that the traditional $k$-median model is not applied to the irregular clusters.

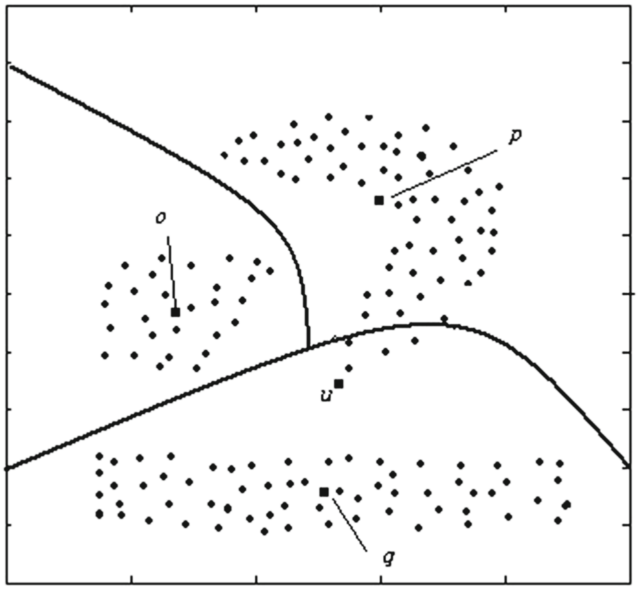

**Fig. 1** Uncapacitated k-median problem

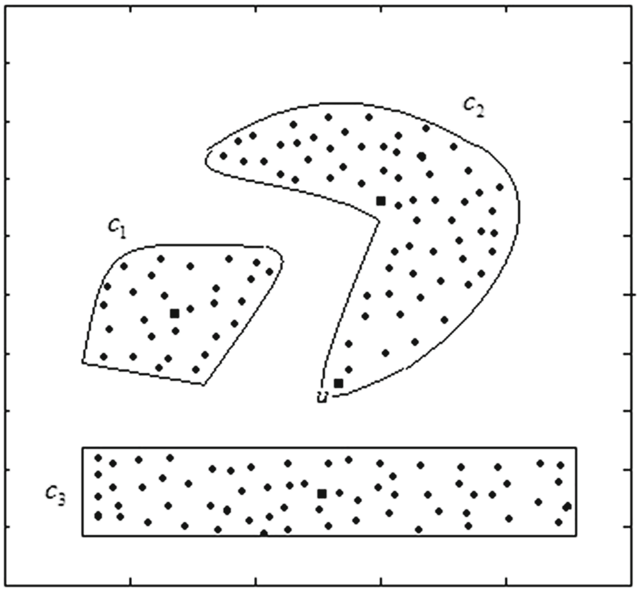

**Fig. 2** $k$-median problem based on connectivity

In order to overcome the impact of the irregular clusters, we propose a $k$-median problem based on connectivity. It tries to find three new medians to respectively serve the three regions, $c_1$, $c_2$ and $c_3$. Obviously, if Euclidean distance is taken to measure the service cost, the clustering result would keep the same as Fig. 1. Considering a single cluster is always a continuous region, each node of the cluster always has high connectivity to other intra-cluster nodes. Hence, we suggest a connectivity measure to describe the cost of each node, and then a $k$-median problem based on connectivity is formed, which is suited to irregularly-shaped clusters. Given that the high connectivity between u and the other nodes in $c_1$, $u$ is classified into $c_1$. Finally, the $k$-median problem based on connectivity reflects the optimal partition as shown in Fig. 2.

## 3.2 A new cost function

**Theorem 1** *Given a data set* $X = \{x_1, x_2, \ldots, x_n\}$, *SDM is a symmetric dissimilarity matrix of X. MST denotes a minimum spanning tree of SDM. Treepath*$(x_i, x_j)$ *denotes the MST path from* $x_i$ *to* $x_j$. $\max$ (***Treepath***$(x_i, x_i)$) ***denotes the weight of the longest edge of the path Treepath***$(x_i, x_j)$. $PC(x_i, x_j) = \{path_1, path_2, \ldots, path_L\}$ *denotes the collection which contains all possible paths from* $x_i$ *to* $x_j$ *in SDM. Then,*

$$\max(Treepath(x_i, x_j)) \equiv \min(\max(path_1), \max(path_2),$$

$$\cdots \max(path_L))$$

***(see Proof 1)***

**Notice:** In the following contents, $\max(G)$ denotes the weight of the longest edge of the graph $G$. $\min(G)$ denotes the weight of the shortest edge of the graph $G$. For example, max(Treepath $(x_{i,}x_j)$) denotes the weight of the longest edge of the $MST$ path Treepath $(x_i, x_j)$; for a $MST$ path Treepath$(x_i x_j)$ $= \left\{\overline{o_i\ o_1}, \overline{o_1\ o_2}, \quad \overline{o_2\ o_3} \quad \ldots \quad \overline{o_{l-1}\ o_l}, \overline{o_l\ x_j}\right\}$, max( Treepath$(x_i, x_j)$) $=$ max({z|z $= |e\ |, e \in$ Treepath$(x_i, x_j)$}). In addition, if $x_i = x_j$, $max(Treepath(x_i, x_j)) = 0$.

**Definition 1** (Cost function – cost$(x_i, x_j)$) Given a data set $X = \{x_1, x_2, \ldots, x_n\}$, SDM is the symmetric dissimilarity matrix of X, implying a weighted undirected graph. The cost function is defined as:

$$\text{cost}(x_i, x_j) = \max(Treepath(x_i, x_j))$$

The essence of the cost function is to reflect the weighted connectivity between two nodes in a $MST$. Meanwhile, the connectivity degree is given by the longest edge of a $MST$ path. If the $MST$ is unique, $Treepath(x_i, x_j)$ is an unambiguous path. Considering that minimum spanning trees which are built from different start nodes may be not unique, $Treepath(x_i, x_j)$ may be an ambiguous path. However, $\max(Treepath(x_i, x_j))$ is always unique according to **Theorem 1**, implying $Treepath(x_i, x_j)$ can be from any $MST$. So, cost$(x_i, x_j)$ is a definite measure.

Considering that the complexity of **Theorem 1** is too high, we compute the cost function according to **Definition 1**.

## 3.3 The equivalent problem

Through introducing the cost function above, the $k$-median problem based on connectivity is formed:

Given a data set $X = \{x_1, x_2, \ldots, x_n\}$, containing $k$ clusters $C = \{c_1, c_2, \ldots, c_k\}$, the task of the clustering problem is to discover the $k$ clusters. We select one node in each cluster, and let it serve the cluster. Suppose that the collection of service nodes is $SN = \{sn_1, sn_2, \ldots, sn_k\}$, where $sn_i$ belongs to $c_i$ and only serves $c_i$. Let cost$(sn_i, o)$ denote the cost of $sn_i$ which serves $o$ . Then, the service cost of $c_i$ is denoted by $C\text{cost}(sn_i, c_i) = \sum_{o \in c_i} \text{cost}(sn_i, o)$, and the total cost is $T\text{cost}(SN, C) = \sum_{i=1}^{k} C\text{cost}(sn_i, c_i)$. If $\text{Ccost}(sn_i, c_i) \equiv \min(\{z \mid z = \text{Ccost}(o, c_i), o \in c_i\})$ holds, then $sn_i$ is a **median** of $c_i$. If the total cost $T\text{cost}(SN, C)$ is the lowest, then $SN$ is a set of optimal

```
Input: $MST = (V, E)$
Output: $CF = \left[\mathrm{cost}\left(x_i, x_j\right)\right]_{n\times n}$

1: $[E\_weight,\ E\_idx] = sort\left(E,'descending'\right)$; //rank edges in descending order.
2: for $i = 1 : (n-1)$ do
3:     Find the $i$-th edge in $E\_idx$, $\overline{u\quad v}$, and the weight, $E\_weight(i)$.
4:     Remove $\overline{u\quad v}$ in $MST$.
5:     $T\_node1 = BFS\_subtree(u)$;
6:     $T\_node2 = BFS\_subtree(v)$;
7:     $CF(T\_node1, T\_node2) = E\_weight(i)$;//According to Definition 1, $cost(a,b) = Eweight(i)$, where $a \in T\_node1$ and $b \in T\_node2$.
8:     $CF(T\_node2, T\_node1) = E\_weight(i)$;//According to Definition 1, $cost(a,b) = Eweight(i)$, where $a \in T\_node2$ and $b \in T\_node1$.
9: end for
10: $BFS\_subtree(u)$: Conduct a breadth-first search on the sub-tree containing $u$ and return all nodes of that sub-tree.
```

**Algorithm 1** Cost function algorithm.

medians. Thus, the $k$-median problem based on connectivity is described as

$$\underset{(SN,C)}{\arg\min}\, T\mathrm{cost}(SN, C)$$

In actual practice, each node always selects the lowest-cost server to obtain service. Then, the service cost of node $p$ should be

$$\mathrm{Ncost}(p)_{SN} = \min(\{z \mid z = \mathrm{cost}(o, p), o \in SN\})$$

where $p \in X$, $\mathrm{cost}\,(sn^*, p) \equiv \mathrm{Ncost}(p)_{\mathrm{SN}}$. The total cost of all clusters is also the total cost of all nodes. Thus, the total cost of all nodes is the dual problem of the original problem. Then, we obtain a new total cost function, $\mathrm{Tcost}(SN, C) = \sum_{p\in X} \mathrm{cost}\,(sn^*, p)\,, sn^* \in SN$ where $sn^*$ denotes the corresponding server. Assuming that $C$ denotes an optimal partition and $SN$ provides the optimal service nodes, then $T\mathrm{cost}(SN, C)$ is the minimum total cost. Hence, for each $p \in X$, $\mathrm{cost}\,(\mathrm{sn}^*, p) \le N\cos t(p)_{SN}$ holds. $\mathrm{cost}\,(sn^*, p) \ge \mathrm{Ncost}(p)\mathrm{SN}$ also holds because $\mathrm{Ncost}(p)_{SN} = \min(\{z \mid z = \mathrm{cost}(o, p), o \in SN\})$. Thus, for each $p \in X$, $\mathrm{cost}\,(\mathrm{sn}^*, p) \equiv \mathrm{Ncost}(p)_{SN}$ holds. Hence, $\mathrm{Tcost}(SN, C) \equiv \mathrm{Tcost}(SN)(= \sum_{p\in X} N\mathrm{cost}(p)_{SN})$ holds for the optimal median collection $SN$ and the optimal partition $C$. Then, the dual problem of the original problem is represented by

$$\underset{SN,|SN|\equiv k}{\arg\min}\ T\mathrm{cost}(SN).$$

Generally, the optimal median collection $SN$ is not unique because a number of equivalent medians exist, such that $C\mathrm{cost}(sn_i', c_i) \equiv C\mathrm{cost}(sn_i, c_i)$ holds for $sn_i' \in c_i$, $sn_i \in c_i$, and $sn_i' \neq sn_i$. In addition, if $\exists p \ni N\mathrm{cost}(p)_{SN} \equiv \mathrm{cost}\,(Sn_i, p) \equiv \mathrm{cost}(Sn_j, p), sn_i, sn_j \in SN$, then the optimal partition $C$ is non-unique. Meanwhile, some optimal partitions may be falsely optimal. For example, a cluster intercross may occur when the total cost function presented earlier obtains the optimal solution. Therefore, we add a few constraints: Similar nodes should be classified into a same cluster, and the bound of a single cluster should be continuous. Finally, the refined problem is given as follows.

Given a data set $X = \{x_1, x_2, \ldots, x_n\}$, $SDM$ denotes the symmetric dissimilarity matrix of X; $MST$ denotes a minimum spanning tree of $SDM$; and $Treepath(x_i, x_j)$ denotes the $MST$ path from $x_i$ to $x_j$. The refined problem can be addressed as:

$$\underset{SN}{\arg\min} T\mathrm{cost}(SN)\ s.t.$$

Constraint 1 :

If $\mathrm{cost}(x_i, x_j) < N\mathrm{cost}(x_i)_{SN} \equiv N\mathrm{cost}(x_j)_{SN}$, $x_i$ and $x_j$ must obtain service from a same server.

Constraint 2 :

$\exists\{st_1, st_2 \cdots st_k\}$ and must obtain service from a same server. where $st_i (i = 1, 2\cdots k)$ is a connected sub-tree of $MST$.

$\ni \{V\,(st_1)\,, V\,(st_2)\cdots V\,(st_k)\}$ is an optimal solution ( $V(st_i)$ denotes the node collection of $st_i$ .).

It is demonstrated that the solution of the constrained problem above is also **the globally optimal solution of the original problem (see Proof 2)**. So, the constrained problem above is taken as the equivalent problem of the $k$-median problem based on connectivity.

## 3.4 Solve the optimal medians

The optimal solution for the aforementioned equivalent problem is a collection composed of $k$ medians. In the following contents, we design a heuristic method ($\mathrm{O}(k^*n^2)$) to search for the $k$ optimal medians:

1. For a data set $X = \left\{x_1, x_2, \ldots, x_n\right\}$, the first server $s_1 = \underset{s\in X}{\arg\min} \sum_{o\in X} \mathrm{cost}(s, o)$.
2. The second server $s_2 = \underset{s\in X/\{s_1\}}{\arg\min} \sum_{o\in X} \min(\mathrm{cost}(s_1, o),$ $\mathrm{cost}(s, o))$.

3. Next, $s_k = \underset{s\in X/\{s_1,s_2\cdots s_{k-1}\}}{\arg\min} \sum_{o\in X} \min(\mathrm{cost}(s_1,o), \mathrm{cost}(s_2,o)\ldots \mathrm{cost}(s_{k-1},o), \mathrm{cost}(s,o))$.
4. Finally, the optimal medians $S_k = \{s_1, s_2 \ldots s_k\} \subset X$ are gradually obtained.

**Theorem 2** *Given a data set* $X = \{x_1, x_2, \ldots, x_n\}$, $Ncost(p)_{SN} = \min(\{z \mid z = cost(o,p), o \in SN\})$ *denotes the cost of p, and* $Tcost(SN) = \sum_{p\in X} Ncost(p)_{SN}$ *denotes the total cost on SN (*$SN \subset X$ *and* $|SN| \equiv k$ *). Then,* $S_k = \{s_1, s_2 \ldots s_k\} \subset X$ *is a solution to* $\underset{SN,|SN|\equiv k}{\arg\min}\ Tcost(SN)$.

***(see Proof 3)***

**Generally, heuristic methods do not ensure an optimal solution. However, through a theoretical demonstration,** $S = \{S_1, \quad S_2, \quad \cdots, S_k\}$**is validated to be a solution to** $\underset{SN,|SN|=k}{\arg\min}\ \mathrm{Tcost}(SN)$ In addition, **Theorem 2** means the first $i$ optimal medians $S_i$ is a subset of $S_k.(i \le k)$. Therefore, a series of optimal solutions exists, such that

$S_1 \subset S_2 \cdots \subset S_i \cdots \subset S_k$.

In order to minimize the total cost function, the servers must have high connectivity to most of the nodes. Hence, unlike general medians or centers, our medians always locate in the dense regions of big sub-clusters instead of the centers of clusters. Figure 3 gives an example which supports this proposition. Suppose the first three optimal servers are $p$, $q$ and $o$ in sequence. Then, when $k = 2$, only $p$ and $q$ are taken as the optimal medians. Meanwhile, node p serves $c_1 \cup c_3$ and $q$ serves $c_2$. When $k = 3$, $p$, $q$ and $o$

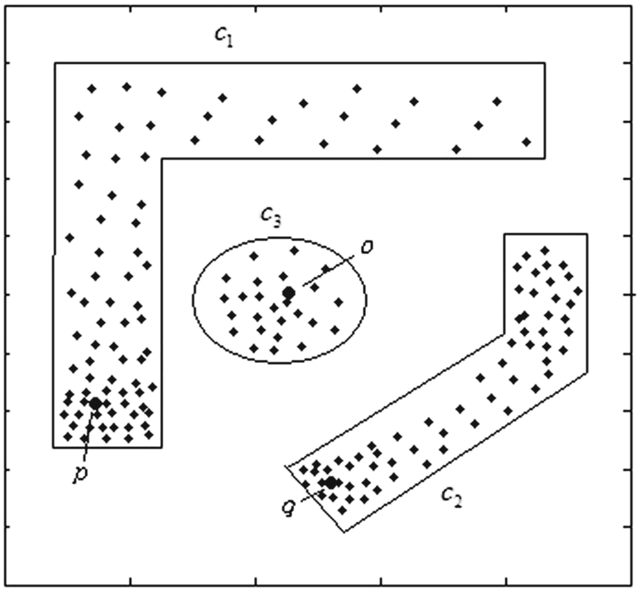

**Fig. 3** $k$-median problem based on connectivity

respectively serve $c_1$, $c_2$ and $c_3$. Obviously, $o$, $p$ and $q$ do locate in the dense regions. When $k = 2$, $o$ would not be taken as a server; on the contrary, $p$ and $q$ are recognized as servers. Hence, it is concluded that the optimal servers always locate in big sub-clusters.

## 3.5 Relevant proof

### 3.5.1 Proof 1 – Theorem 1

**Lemma 1** *If the node set of a cycle is arbitrarily divided into two disjoint nonempty subsets, there exist at least two edges whose endpoints belong to the two nonempty subsets respectively.*

*Proof of Lemma 1* We argue by contradiction. Let us consider the following two cases:

Case 1. If there has no edge connected the two nonempty subsets, then we easily obtain that the cycle is disconnected, a contradiction.

Case 2. If there has a unique edge, say $u_i u_j$, connected the two nonempty subsets, then there exists a unique path from $u_i$ to $u_j$ is in the cycle, implying that $u_i u_j$ is not the edge of the cycle, a contradiction. So, **Lemma 1** holds. □

**Lemma 2** *Let C be an arbitrary cycle of the connected graph* $G = (V, E)$*, and C's longest edge is* ***unique****. Then the unique longest edge in C doesn't locate in the minimum spanning tree MST of G.*

*Proof of Lemma 2* Denote $\{u_1, u_2, ..., u_k\}$ by the node set of $C$. Without loss of generality, assume that $\overline{u_i u_j}$ is the unique longest edge of $C$. If $\overline{u_i u_j}$ locates in the minimum spanning tree $MST$ of $G$, then we delete the edge $C$ in $MST$, yielding that $MST$ is decomposed into two connected sub-trees $MST_1$ and $MST_2$. Let $V(C_1) = V(C) \cap V(MST_1)$ and$V(C_2) = V(C) \cap V(MST_2)$. Then$V(C_1)$ and $V(C_2)$ are nonempty; one of which contains at least $u_i$, the other contains at least $u_j$, otherwise deleting $\overline{u_i u_j}$ in $MST$, $MST$ is also connected, a contradiction to the minimum spanning tree. By **Lemma 1**, there exists other edge $\overline{u_l u_m}$ connected the sub-trees $MST_1$ and $MST_2$, which is shorter than $\overline{u_i u_j}$ in $G$, implying that the whole of $MST_1$, $MST_2$, and $\overline{u_l u_m}$ possesses less total weight than $MST$, a contradiction.

So, **Lemma 2** holds. □

*Proof of Theorem 1* For convenience, we design some conditions: $MST$ is an arbitrary minimum spanning tree. There are two arbitrary nodes $x_i$ and $x_j$. Let $Treepath(x_i, x_j)$ denote the $MST$ path from $x_i$ to $x_j$. There exists an edge $\overline{a \quad b} \in Treepath\left(x_i, x_j\right)$ such that $|\overline{a\quad b}| \equiv$ max (Treepath $(x_i, x_j)$) holds. The paths, which do not contain $\overline{a\quad b}$ in $PC(x_i, x_j)$, make up of a collection $sub\_pc$, Let $Pt$ denote an arbitrary path in $sub\_pc$, implying $\overline{a\ b} \notin pt$.

$\because \overline{a\ b} \in Treepath(x_i, x_j)$ and $\overline{a\ b} \notin pt$.

$\therefore Treepath(x_i, x_j)$and $pt$ are two different paths.

$\because Treepath(x_i, x_j) \cup pt$ constructs a circle $Cir$ which contains the edge $\overline{a\ b}$.

$\therefore$**Lemma1 2** $\Rightarrow \left|\overline{a\quad b}\right| \leq$ max(Cir), and the longest edge of $Cir$ must locate in $pt$.

$\therefore \max(pt) \geq |\overline{a\quad b}| \equiv \max(Treepath(x_i, x_j))$

$\because pt$ denotes an arbitrary path in $sub\_pc$.

$\therefore \forall pt \in sub\_pc, \ni \max(pt) \geq mx(Treepath(x_i, x_j))$

$\because \forall ph \in PC/sub\_pc, |\overline{a\ b}| \in ph$

$\therefore \forall ph \in PC/sub\_pc, \max(ph) \geq |\overline{a\ b}| \equiv \max(Treepath(x_i, x_j))$

$\therefore \max(Treepath(x_i, x_j)) \leq \min(\max(path_1), \max(path_2), \cdots \max(path_L))$

$\because PC(x_i, x_j) = \{path_1, path_2, \cdots path_L\}$ denotes the collection including all possible paths between $x_i$ and $x_j$ in$SDM$.

$\therefore Treepath(x_i, x_j) \in PC(x_i, x_j)$

$\therefore \max(Treepath(x_i, x_j)) \geq \min(\max(path_1), \max(path_2), \cdots \max(path_L))$

$\therefore \max(Treepath(x_i, x_j)) \equiv \min(\max(path_1), \max(path_2), \cdots \max(path_L))$ □

### 3.5.2 Proof 2 – equivalent problem

The demonstration process contains two parts:

1. The solution of the refined problem is a globally optimal solution of the original problem under Constraint 1.
2. The solution of the refined problem is a globally optimal solution of the original problem under Constraint 2.

**Validation: The solution of the refined problem is a globally optimal solution under Constraint 1.**

Given a data set $X = \{x_1, x_2, \ldots, x_n\}$, it includes k optimal regions, $C = \{c1, c2, \ldots, ck\}$, $|c_1| + |c_2| \ldots |c_k| \equiv$ n, and $SN = \{sn_1, sn_2, \ldots, sn_k\}$ denotes the corresponding $k$ optimal medians such that $C\text{cost}(sn_i, c_i) = \sum_{o \in c_i} \text{cost}(sn_i, o)$ obtains the minimum cost of $C_i$, and $T\text{cost}(SN)$ obtains the minimum total cost. Let $N\text{cost}(p)_{SN} = \min(\{z \mid z = \text{cost}(o, p), o \in SN\})$ Suppose $\exists$ $a$ and $b$, $\ni$ $\text{cost}(a, b) < N\cos t(a)_{SN} \equiv N\text{cost}(b)_{SN}$, $N\text{cost}(a)_{SN} \equiv \text{cost}(sn_a, b)$, and $N\text{cost}(b)_{SN} \equiv \text{cost}(sn_b, b)$ where $sn_a, sn_b \in SN$, $sn_a \in c_a$, $sn_b \in c_b$, $c_a, c_b \subset C$. Then, $a$ and $b$ must be classified into a same cluster according to **Constraint 1**.

Because of $N\text{cost}(a)_{SN} \equiv N\text{cost}(b)_{SN}$, $a$ and $b$ can be together classified into either $c_a$ or $c_b$. Firstly, let $a$ and $b$ be classified into $c_a$. By **Theorem 1**, $\max(Treepath(sn_a, b)) \leq \max(Treepath(sn_a, a) \cup Treepath(a, b))$ Then, $\text{cost}(sn_a, b) \leq \max(\text{cost}(sn_a, a), \text{cost}(a, b))$ where $\text{cost}(sn_a, b) = \max(Treepath(sn_a, b))$ and $\max(\text{cost}(sn_a, a), \text{cost}(a, b)) = \max(Treepath(sn_a, a) \cup Treepath(a, b))$. Due to $\text{cost}(a, b) < N\text{cost}(a)_{SN} \equiv N\text{cost}(b)_{SN}$, $\max(\text{cost}(sn_a, a), \text{cost}(a, b)) \equiv \text{cost}(sn_a, a)$ holds. Then, $\text{cost}(sn_a, b) \leq \max(\text{cost}(sn_a, a), \text{cost}(a, b)) \equiv \text{cost}(sn_a, a) \equiv N\text{cost}(a)_{SN} \equiv N\text{cost}(b)_{SN}$, that is, $\text{cost}(sn_a, b) \leq N\text{cost}(b)_{SN}$ holds. If other nodes do not apply Constraint 1, then $\forall o \in X/b$, $\ni$ $\text{cost}(sn^*, o) \leq N\text{cost}(o)$, where $sn*$ is the given server of $o$. So, $\text{Tcost}(SN, C') = \sum_{i=1}^{k} \text{Ccost}(sn_i, c_i) \leq \sum_{p \in X} N\text{cost}(p)_{SN} = \text{Tcost}(SN)$ where $C'$ denotes a new partition after applying Constraint 1. Because $T\text{cost}(SN)$ obtains the minimum total cost on $SN$, $\text{Tcost}(SN, C') \geq \text{Tcost}(SN)$ holds. So, $\text{Tcost}(SN, C') \equiv \text{Tcost}(SN)$ holds. Similarly, if $a$ and $b$ are classified into $c_b$, $\text{Tcost}(SN, C') \equiv \text{Tcost}(SN)$ also holds. So, **the solution of the refined problem is a globally optimal solution under Constraint 1**.

**Validation: The solution of the refined problem is a globally optimal solution under Constraint 2.**

There are two arbitrary nodes $p_m, q_l$ such that

$$\begin{aligned}\text{Ncost}(p_m)_{SN} &\equiv \text{cost}(sn_p, p_m) \\ &< \min(\{z \mid z = \text{cost}(sn, p_m), \text{sn} \in SN/sn_p\})\end{aligned}$$

$$\begin{aligned}\text{Ncost}(q_l)_{SN} &\equiv \text{cost}(sn_q, q_l) \\ &\equiv \min(\{z \mid z = \text{cost}(sn, q_l), sn \in SN/sn_q\})\end{aligned}$$

Let $Treepath(x_i, x_j)$ denote the path from $x_i$ to $x_j$ in a minimum spanning tree, $MST$.

$\because$ All nodes are connected in $MST$.

$\therefore \exists Treepath(sn_p, p_m) = \{\overline{sn_p\ p_1}, \overline{p_1\ p_2} \cdots, \overline{p_{m-1}\ p_m}\}$

$\because$

$$\begin{aligned}\text{Ncost}(p_m)_{\text{SN}} &\equiv \text{cost}(sn_p, p_m) \\ &< \min(\{z \mid z = \text{cost}(sn, p_m), sn \in SN/sn_p\})\end{aligned}$$

$\therefore \forall sn_u \in SN/sn_p, \text{cost}(p_j, p_m) \leq \text{cost}(sn_p, p_m) < \text{cost}(sn_u, p_m)$, $j \leq m$

$\therefore \text{cost}(sn_u, p_m) > \text{cost}(p_j, p_m)$

$\because \text{cost}(sn_u, p_m) \leq \max(\text{cost}(sn_u, p_j), \text{cost}(p_j, p_m))$

$\therefore \text{cost}(sn_u, p_m) \leq \text{cost}(sn_u, p_j)$

$\because Treepath(sn_p, p_j) \subseteq Treepath(sn_p, p_m)$ and $N\text{cost}(p_m)_{SN} \equiv \text{cost}(sn_p, p_m)$

$\therefore \text{cost}(sn_p, p_j) \le \text{cost}(sn_p, p_m) < \text{cost}(sn_u, p_m)$

$\therefore \text{cost}(sn_p, p_j) < \text{cost}(sn_u, p_j)$

$\therefore \forall sn_u \in SN/sn_p, \ni \text{cost}(sn_p, p_j) < \text{cost}(sn_u, p_j)$

$\therefore \text{Ncost}\left(p_j\right)_{SN} \equiv \text{cost}(sn_p, p_j) < \min(\{z \mid z = \text{cost}(sn, p_j), sn \in SN/sn_p\}), j \le m$

$\therefore \forall p_j \in \{p_1, p_2 \cdots p_m\}, \ni \text{Ncost}(p_j)_{SN} \equiv \text{cost}(sn_p, p_j) < \min(\{z \mid z = \text{cost}(sn, p_j), sn \in SN/sn_p\}), j \le m$

$\because$ All nodes are connected in $MST$

$\therefore \exists Treepath(p'_m, q_l) = \{\overline{p'_m\ q_1}, \overline{q_1\ q_2} \cdots, \overline{q_{l-1}\ q_l}\}$, where N cost $(q_i)_{SN} \equiv \text{cost}(sn_q, q_i) \equiv \min(\{z \mid z = \text{cost}(sn, q_i), sn \in SN/sn_q\}), i = 1, 2 \cdots l$, and

$\text{Ncost}(p'_m)_{SN} \equiv \text{cost}(sn_q, p'_m) < \min(\{z \mid z = \text{cost}(sn, p'_m), sn \in SN/sn_q\})$.

$\therefore \exists Treepath(sn_q, q_l) = \{\overline{sn_q\ p'_1}, \overline{p'_1\ p'_2} \cdots, \overline{p'_{m-1}\ p'_m}, \overline{p'_m\ q_1}, \overline{q_1\ q_2} \cdots, \overline{q_{l-1}\ q_l}\} \forall N\text{cost}(o)_{SN} \equiv \text{cost}(sn_p, o), o \in V(Treepath(sn_q, q_l))$

$\therefore$ Classify $\{p'_1, p'_2 \cdots, p'_m, q_1, q_2 \cdots q_l\}$ into $sn_q$'s cluster. Meanwhile, it is obvious that they locate in a sub-tree of $MST$ (because there exists the path $Treepath(p'_m, q_l)$).

$\therefore$ Similarly, each node locates in the respective sub-tree $st_i$.

$\because$ For each $o \in V(st_i)$, $N\text{cost}(o) \equiv \text{cost}(sn_i, o)$ holds.

$\therefore C\text{cost}(sn_i, V(st_i)) \equiv \sum\limits_{o \in V(st_i)} N\text{cost}(o)$

$\therefore$ Similarly, for each sub-tree $st_i$, $C\text{cost}(sn_i, V(st_i)) \equiv \sum\limits_{o \in V(st_i)} N\text{cost}(o)$ holds.

$\therefore T\text{cost}(SN, \{V(st_1), V(st_2) \cdots V(st_k)\}) = \sum\limits_{i=1}^{k} C\text{cost}(sn_i, c_i) \equiv \sum\limits_{o \in X} N\text{cost}(o) = T\text{cost}(SN)$

$\therefore$ **The solution of the refined problem is a globally optimal solution under Constraint 2.**

$\therefore$ **The solution of the refined problem is globally optimal under Constraints 1 and 2.**

### 3.5.3 Proof 3 – Theorem 2

The process of validating the globally optimal medians includes 2 key steps:

1. The first server $s_1$ is an optimal median (Correspond to $sn_1$ ).
2. The first $i$ servers are $i$ optimal medians (Correspond to $\{sn_1, sn_2 \cdots sn_i\} \subseteq SN$).

**Validation:** The first server $s_1$ is an optimal median ($sn_1$). Before demonstration, the partitioning rules are given as follows:

1. Given a set of service nodes (servers), $SN = \{sn_1, sn_2, \ldots sn_k\}$, each node selects the lowest-cost service node to obtain service, that is, $\text{Ncost}(p)_{SN} = \min(\{z \mid z = \text{cost}(o, p), o \in SN\})$
2. Each cluster must be subjected to **Constraints 1** and **2**.

**Lemma 3** *There is a set of arbitrary servers, $SN = \{sn_1, sn_2, \ldots sn_k\}$ and let $Ncost(p)_{SN} = \min(\{z \mid z = cost(o, p), o \in SN\})$ denote the cost of $p$. Then, each node selects the respective lowest-cost server to obtain service, and all nodes are classified into $k$ clusters $C = \{c_1, c_2, \ldots c_k\}$ according to $SN$. If the partition is subject to **Constraints 1** and **2**, then*

$$\max(cost(sn_i, b_1), cost(sn_j, b_2)) \le cost(b_1, b_2)$$

*holds, where $sn_i, sn_j \in SN$, $sn_i, b_1 \in c_i$, $s_j, b_2 \in c_j$, $V(Treepath(b_1, b_2)) / \{b_1, b_2\}\ X/(c_i \cup c_j)$, and $Treepath(b_1, b_2)$ denotes the $MST$ path from $x_i$ to $x_j$.*

**Notice:** $V(Treepath(b_1, b_2)) / \{b_1, b_2\}\ X/(c_i \cup c_j)$ means that $b_1$ and $b_2$ denote two boundary nodes of $c_i$ and $c_j$, respectively. $V(G)$ denotes the node collection of $G$.

*Proof of Lemma 3* As for the demonstration of **Lemma 3**, Fig. 4 gives the necessary details. All edges are the edges of an arbitrary minimum spanning tree. Each circle denotes a cluster. Because each node always selects the lowest-cost server to obtain service, each server would cover a continuous region, denoted by a circle in Fig. 4. The demonstration is given as follows:

$\because sn_i, b_1 \in c_i, sn_j, b_2 \in c_j.$

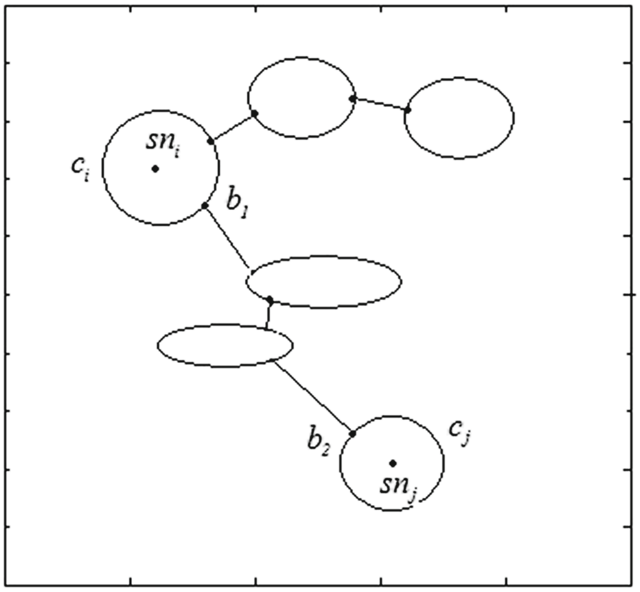

**Fig. 4** Constraints

$\therefore$ $\mathrm{cost}(sn_j, b_2) \le \mathrm{cost}(sn_i, b_2) = \max(\mathrm{cost}(sn_i, b_1), \mathrm{cost}(b_1, b_2))$, and $\mathrm{cost}(sn_i, b_1) \le \mathrm{cost}(sn_j, b_1) = \max(\mathrm{cost}(sn_j, b_2), \mathrm{cost}(b_1, b_2))$.

$\therefore$ $\max(\mathrm{cost}(sn_i, b_1), \mathrm{cost}(sn_j, b_2)) \le \max(\mathrm{cost}(sn_j, b_2), \mathrm{cost}(b_1, b_2))$, and $\max(\mathrm{cost}(sn_i, b_1), \mathrm{cost}(sn_j, b_2)) \le \max(\mathrm{cost}(sn_i, b_1), \mathrm{cost}(b_1, b_2))$.

$\max(\mathrm{cost}(sn_i, b_1), \mathrm{cost}(sn_j, b_2)) \le$
$\therefore$ $\min(\max(\mathrm{cost}(sn_i, b_1), \mathrm{cost}(b_1, b_2)),$ .
$\max(\mathrm{cost}(sn_j, b_2), \mathrm{cost}(b_1, b_2)))$
$\min(\max(\mathrm{cost}(sn_i, b_1), \mathrm{cost}(b_1, b_2)),$
$\max(\mathrm{cost}(sn_j, b_2), \mathrm{cost}(b_1, b_2)))$
$\because$ $= \min(\max(\mathrm{cost}(sn_i, b_1), \mathrm{cost}(sn_j, b_2),$ .
$\mathrm{cost}(b_1, b_2)),$
$\max(\min \mathrm{cost}(sn_i, b_1), \mathrm{cost}(sn_j, b_2)), \mathrm{cost}(b_1, b_2)))$

$\therefore$ $\max(\mathrm{cost}(sn_i, b_1), \mathrm{cost}(sn_j, b_2)) \le \max(\min(\mathrm{cost}(sn_i, b_1), \mathrm{cost}(sn_j, b_2)), \mathrm{cost}(b_1, b_2))$.

$\therefore$

$$\max(\mathrm{cost}(sn_i, b_1), \mathrm{cost}(sn_j, b_2)) \le \min(\mathrm{cost}(sn_i, b_1), \mathrm{cost}(sn_j, b_2)) \quad (1)$$

or

$$\max(\mathrm{cost}(sn_i, b_1), \mathrm{cost}(sn_j, b_2)) \le \mathrm{cost}(b_1, b_2) \quad (2)$$

$\because$ (1) $\Rightarrow$ $\mathrm{cost}\,(\mathrm{sin}_i, b_1) \equiv \mathrm{cost}\left(\mathrm{sn}_j, b_2\right)$, and $sn_i, b_1 \in c_i, sn_j, b_2 \in c_j$.

$\therefore$ $N\mathrm{cost}(b_1)_{SN} \equiv N\mathrm{cost}(b_2)_{SN}$ holds. (Due to each node selects the respective lowest-cost server to obtain service, $\mathrm{cost}(sn_i, b_1) \equiv N\mathrm{cost}(b_1)_{SN}$ and $\mathrm{cost}(sn_j, b_2) \equiv N\mathrm{cost}(b_2)_{SN}$.).

$\therefore$ **Constraint 1** $\Rightarrow$ $\max(\mathrm{cost}(\mathrm{sn}_i, b_1), \mathrm{cost}(\mathrm{sn}_j, b_2)) \le \mathrm{cost}\,(b_1, b_2)$.

$\therefore$ (1) and (2) $\Rightarrow$ $\max(\mathrm{cost}(sn_i, b_1), \mathrm{cost}(sn_j, b_2)) \le \mathrm{cost}\,(b_1, b_2)$.

$\therefore$ **Lemma 3** holds. □

**Lemma 4** *There is a set of optimal medians* $SN = \{sn_1, sn_2 \cdots sn_k\}$ *which minimizes* $f(SN) = Tcost(SN)$, $|SN| \equiv k$. *Meanwhile, according to the partitioning rules above, all nodes are classified into k optimal clusters* $C = \{c_1, c_2 \cdots c_k\}$. *The first server* $s_1$ *is one of the k optimal medians, that is,*

$$Ccost(sn_1, c_1) \equiv Ccost(s_1, c_1), sn_1, s_1 \in c_1$$

*Proof of Lemma 4* Given a data set $X = \{x_1, x_2, \ldots, x_n\}$ which contains k optimal clusters—$C = \{c_1, c_2, \ldots, c_k\}$, $SDM$ denotes the symmetric dissimilarity matrix of $X$ and $MST$ is a minimum spanning tree of $SDM$. Let $Treepath(x_i, x_j)$ denote the edge collection of the $MST$ path from $x_i$to $x_j$. $V(G)$ denotes the node collection of $G$. The optimal cluster collection $C = \{c_1, c_2 \cdots c_k\}$ is subject to $|c_1|+|c_2|\ldots|c_k| \equiv n$,

and the optimal median collection is $SN = \{sn_1, sn_2, \ldots, sn_k\}$. According to **Theorem 2**, the server collection is obtained, $S_k = \{s_1, s_2 \cdots s_k\}$. There is a path $Treepath\,(b_1, b_2)$ $(b_1 \in c_1, b_2 \in c_2)$which does not contain any $o \in (c_1 \cup c_2)/\{b_1, b_2\}$ (**see Constraint 2**). Others details are shown in Fig. 5. All edges of Fig. 5 are the edges of $MST$ and each circle denotes an optimal cluster.

Suppose the first server $s_1$ locates in $c_1$, and the optimal median of $c_1$ is $sn_1$. If $C\mathrm{cost}(sn_1, c_1) \equiv C\mathrm{cost}(s_1, c_1)$ holds, $s_1$ is an optimal median due to $T\mathrm{cost}(\{sn_1, sn_2, sn_3 \cdots sn_k\}, C) \equiv T\mathrm{cost}(\{s_1, sn_2, sn_3 \cdots sn_k\}, C)$, where $\{sn_1, sn_2, sn_3 \cdots sn_k\}$ is a set of optimal medians and $C$ is the corresponding optimal partition.

$\because$ $Treepath(s_1, x_j) = Treepath(s_1, b_1) \cup Treepath\,(b_1, b_2) \cup Treepath(b_2, x_j), x_j \in c_2$;
$Treepath(sn_1, x_j) = Treepath(sn_1, b_1) \cup Treepath(b_1, b_2) \cup Treepath(b_2, x_j), x_j \in c_2$.

$\therefore$ $\mathrm{cost}(s_1, x_j) = \max(\mathrm{cost}(s_1, b_1), \mathrm{cost}(b_1, b_2), \mathrm{cost}(b_2, x_j)), x_j \in c_2$ and $\mathrm{cost}(sn_1, x_j) = \max(\mathrm{cost}(sn_1, b_1), \mathrm{cost}(b1, b_2), \mathrm{cost}(b_2, x_j)), x_j \in c_2$

$\because$**Lemma 3** $\Rightarrow$ $\mathrm{cost}\,(b_1, b_2) \ge \mathrm{cost}\,(sn_1, b_1)$

$\therefore$ $\mathrm{cost}(sn_1, x_j) = \max(\mathrm{cost}(b_1, b_2), \mathrm{cost}(b_2, x_j)), x_j \in c_2$

$\because$ $\max(\mathrm{cost}(b_1, b_2), \mathrm{cost}(b_2, x_j)) \le \max(\mathrm{cost}(s_1, b_1), \mathrm{cost}(b_1, b_2), \mathrm{cost}(b_2, x_j)) = \mathrm{cost}(s_1, x_j)$

$\therefore$ $\mathrm{cost}(sn_1, x_j) \le \mathrm{cost}(s_1, x_j), x_j \in c_2$

$\therefore$ $\forall x_j \in c_2 \mathrm{cost}(sn_1, x_j) \le \mathrm{cost}(s_1, x_j)$

$\therefore$ $C\mathrm{cost}(sn_1, c) \le C\mathrm{cost}(s_1, c_2)$

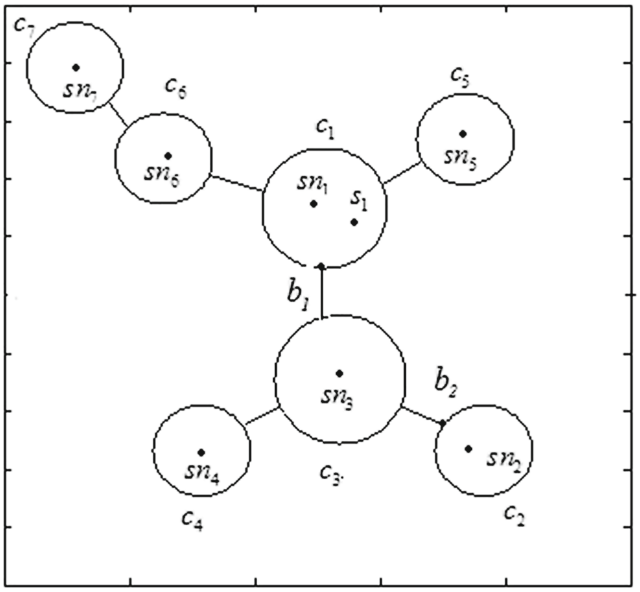

**Fig. 5** Constraints

$\therefore$ Similarly, $\forall c \in C/\{c_1\}, \ni$ Ccost $(sn_1, c) \leq$ Ccost $(s_1, c)$

$\therefore \sum_{o\in X/c_1} \text{cost}(sn_1, o) \leq \sum_{o\in X/c_1} \text{cost}(s_1, o)$

$\because$ $C\text{cost}(sn_1, c_1) \leq C\text{cost}(s_1, c_1)$ (because $sn_1$ is the medians of $c_1$)

$\therefore$

$$T\text{cost}(sn_1) = C\text{cost}(sn_1, c_1) + \sum_{o\in X/c_1} \text{cost}(sn_1, o) \leq C\text{cost}(s_1, c_1) + \sum_{o\in X/c_1} \text{cost}(s_1, o) = T\text{cost}(s_1) \quad (3)$$

$\because$ When all nodes are classified into one cluster, $S_1$ minimizes $f(SN) = T\text{cost}(SN)$, $|SN| \equiv 1$.

$\therefore$

$$T\text{cost}(sn_1) \geq T\text{cost}(s_1) \quad (4)$$

$\therefore$ (3) and (4) $\Rightarrow$ Tcost $(sn_1) \equiv$ Tcost $(s_1)$.

$\because$

$C\text{cost}(sn_1, c_1) + \sum_{o\in X/c_1} \text{cost}(sn_1, o) = T\text{cost}(sn_1) \equiv T\text{cost}(s_1) = C\text{cost}(s_1, c_1) + \sum_{o\in X/c_1} \text{cost}(s_1, o)$; $C\text{cost}(sn_1, c_1) \leq C\text{cost}(s_1, c_1)$; $\sum_{o\in X/c_1} \text{cost}(sn_1, o) \leq \sum_{o\in X/c_1} \text{cost}(s_1, o)$

$\therefore \sum_{o\in X/c1} \text{cost}(sn_1, o) \equiv \sum_{o\in X/c1} \text{cost}(s_1, o)$ and $C\text{cost}(sn_1, c_1) \equiv C\text{cost}(s_1, c_1)$

$\therefore T\text{cost}(SN', C) \equiv T\text{cost}(SN, C) \equiv T\text{cost}(SN)$ holds where $SN = \{sn_1, sn_2, sn_3 \cdots sn_k\}$ denotes the optimal median collection, $C$ is the optimal partition, and $SN' = \{s_1, sn_2, sn_3 \cdots sn_k\}$.

$\therefore s_1$ is an optimal median.

**Validation:** The first $i$ servers are optimal medians, that is, $\{s_1, s_2 \cdots s_i\} \subseteq SN = \{sn_1, sn_2 \cdots sn_k\}, i \leq k$, where $SN$ is a set of optimal medians.

The rules of partitioning $k$ regions include two items:

1. Each node selects the lowest-cost server, that is $\text{Ncost}(p)_{S_k} = \min(\{z \mid z = \text{cost}(o, p), o \in S_k\})$ denotes the cost of $p$, where $S_k = \{s_1, s_2 \cdots s_k\}$ is the first $k$ servers obtained by **Theorem 2**.
2. Each cluster is subjected to **Constraints 1** and **2**.

There is a set of optimal medians $SN = \{sn_1, sn_2 \cdots sn_k\}$ such that $T\text{cost}(SN)$ obtains the minimum total cost.$S_i = \{s_1, s_2 \cdots s_i\}$ denotes the first $i$ servers. Next, let's demonstrate that $T\text{cost}(\{s_1, s_2 \cdots s_i, sn_{i+1} \cdots sn_k\}) \equiv T\text{cost}(SN)$ holds. □

*Proof of Theorem 2* Given a data set $X = \{x_1, x_2, \ldots, x_n\}$ which contains $k$ optimal clusters $C = \{c_1, c_2, \ldots, c_k\}$ ($|c_1|+|c_2|\ldots|c_k|\equiv$ n), let $SDM$ denote the symmetric dissimilarity matrix of $X$ and $MST$ denote a minimum spanning tree of $SDM$. The optimal median collection is denoted by $SN = \{sn_1, sn_2, \ldots, sn_k\}$. Theorem 2 gives the first $i$ servers $S_i = \{s_1, s_2 \cdots s_i\}$. According to the first $i$ servers and the partition rules above, we would get a partition $SC_i = \{sc_1, sc_2 \cdots sc_i\}$ ($|sc_1| + |sc_2| \ldots |sc_i| \equiv n$). Now, let us demonstrate **the first $i$ servers are optimal (by inductive method)**, that is, $T\text{cost}(\{s_1, s_2 \cdots s_i, sn_{i+1} \cdots sn_k\}) \equiv T\text{cost}(SN)$.

**When** $i = 1$

$\because$ **Lemma 4**$\Rightarrow$The first 1 server is one of the $k$ optimal medians.

$\therefore$**The first** $i$**servers are optimal medians.**

**When** $i > 1$

Suppose **the first** $i - 1$ **servers are optimal medians**, and then we can replace $\{sn_1, sn_2 \cdots sn_{i-1}\}$ with $\{s_1, s_2 \cdots s_{i-1}\}$. Let $s_i \in c_i$ denote the $i$-th server, and then we have $S_i = \{sn_1, sn_2 \cdots sn_{i-1}, s_i\}$. As for the first $i$ servers, there is a partition $SC_i$ according to the partition rules above, $SC_i = \{sc_1, sc_2 \cdots sc_i\}$ $|sc_1|+|sc_2| \ldots |sc_i| \equiv n$. Let $sn_j \in c_j$ denote the server of $sc_j$, $j < i$, and $sn_i \in c_i$ denote an optimal median of $c_i$. In Fig. 6, all edges are the edges of $MST$. The first $i$-1 servers cover the region **A** ($SC_i/sc_i$) and $s_i$ serves the other one, **B** ( = $sc_i$ ). The details are shown in Fig. 6.

The demonstration includes 2 steps:

1. Demonstrate that each optimal cluster in region$B$ is whole. To be exact, if $B$ contains a part (e.g. $cb$) of an optimal cluster $c_d$ instead of the whole $c_d$, $C' = \{c_1, c_2 \cdots c_d/cb, \cdots c_u \cup cb, \cdots c_k\}$ is also an optimal partition (imply, $T\text{cost}(SN, C) \equiv T\text{cost}(SN, C')$

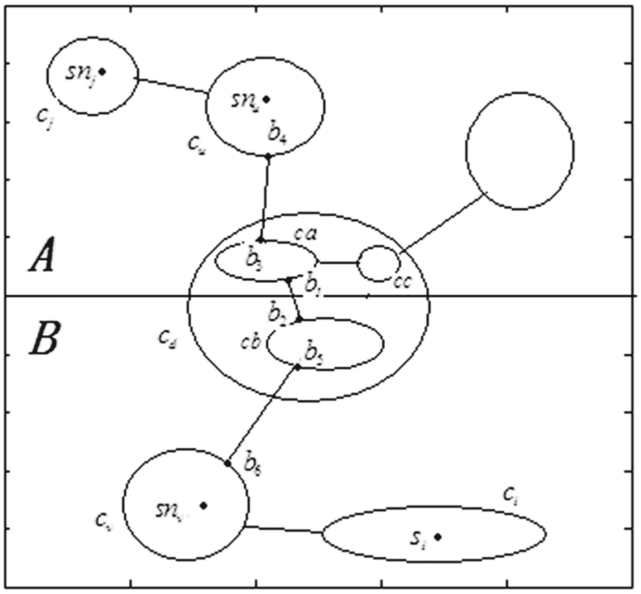

**Fig. 6** The first $i$ optimal medians

where $C = \{c_1, c_2 \cdots c_d \cdots c_u \cdots c_k\}$ denotes an optimal partition and $SN$ is the corresponding optimal median collection.).

2. Demonstrate that $s_i$ is an optimal median.

According to **Constraints 1** and **2**, if $A$ and $B$ jointly split one of the $k$ optimal clusters, the details would be shown as Fig. 6. Let $c_d = ca \cup cb \cup cc$, $b_1 \in ca$, $b_2 \in cb$, $\overline{b_1\ b_2} \in MST$, $s_i \in c_i$, $(V(Treepath(b_1, b_2))/\{b_1, b_2\}) \subseteq X/(c_i \cup c_j)$. $c_u$ is the most similar cluster of $ca$ in $sc_j$, $ca, c_j, c_u \subseteq sc_j$, and $c_v$ is the most similar cluster of $cb$, $cb, c_i, c_v \subseteq sc_i$.

Notice: It is unknown that whether $c_i$ is $c_d$. We would discuss the different cases in the following demonstration.

$(c_i \neq c_d)$

$\because$ In the partition $SC_i$, $sn_j$ denotes the server of $sc_j$ and $s_i$ denotes the server of $sc_i$; $b_1 \in ca \subseteq sc_j$, $b_2 \in cb \subseteq sc_i$, $(V(Treepath(b_1, b_2))/\{b_1, b_2\}) \subseteq X/(c_i \cup c_j)$.

$\therefore$ **Lemma 3** $\Rightarrow \max(\mathrm{cost}(sn_j, b_1), \mathrm{cost}(s_i, b_2)) \leq \mathrm{cost}(b_1, b_2)$

$\because$ **Lemma 3** $\Rightarrow \mathrm{cost}(sn_v, b_6) \leq \mathrm{cost}(b_5, b_6)$ and $\mathrm{cost}(sn_u, b_4) \leq \mathrm{cost}(b_3, b_4)$

$\therefore \mathrm{cost}(sn_v, b_2) \leq \mathrm{cost}(b_2, b_6)$ and $\mathrm{cost}(sn_u, b_1) \leq \mathrm{cost}(b_1, b_4)$

$\because Treepath(s_i, b_2) = Treepath(s_i, b_6) \cup Treepath(b_2, b_6)$, $Treepath(sn_j, b_1) = Treepath(sn_j, b_4) \cup Treepath(b_1, b_4)$

$\therefore \mathrm{cost}(b_2, b_6) \leq \mathrm{cost}(s_i, b_2)$; $\mathrm{cost}(b_1, b_4) \leq \mathrm{cost}(sn_j, b_1)$

$\therefore \mathrm{cost}(sn_v, b_2) \leq \mathrm{cost}(s_i, b_2)$; $\mathrm{cost}(sn_u, b_1) \leq \mathrm{cost}(sn_j, b_1)$

$\therefore \max(\mathrm{cost}(sn_u, b_1), \mathrm{cost}(sn_v, b_2)) \leq \max(\mathrm{cost}(sn_j, b_1), \mathrm{cost}(s_i, b_2))$

$\therefore$

$$\max(\mathrm{cost}(sn_u, b_1), \mathrm{cost}(sn_v, b_2)) \leq \mathrm{cost}(b_1, b_2) \tag{5}$$

$\because$

$Treepath(sn_u, b_1) = Treepath(sn_u, b_3) \cup Treepath(b_1, b_3)$, $Treepath(sn_v, b_2) = Treepath(sn_v, b_5) \cup Treepath(b_2, b_5)$

$\therefore \mathrm{cost}(sn_u, b_1) \geq \mathrm{cost}(sn_u, b_3)$; $\mathrm{cost}(sn_v, b_2) \geq \mathrm{cost}(sn_v, b_5)$

$\therefore$

$$\max(\mathrm{cost}(sn_u, b_3), \mathrm{cost}(sn_v, b_5)) \leq \mathrm{cost}(b_3, b_5) \tag{6}$$

$\because$ **Lemma 3** $\Rightarrow \mathrm{cost}(sn_d, b_5) \leq \mathrm{cost}(b_5, b_6)$, $\mathrm{cost}(sn_d, b_3) \leq \mathrm{cost}(b_3, b_4)$

$\therefore \max(\mathrm{cost}(sn_d, b_3), \mathrm{cost}(sn_d, b_5)) \leq \max(\mathrm{cost}(b_3, b_4), \mathrm{cost}(b_5, b_6))$

$\because Treepath(b_3, b_5) \subseteq (Treepath(sn_d, b_3) \cup Treepath(sn_d, b_5))$

$\therefore \max(\mathrm{cost}(sn_d, b_3), \mathrm{cost}(sn_d, b_5)) \geq \mathrm{cost}(b3, b_5)$

$\therefore \mathrm{cost}(b_3, b_5) \leq \max(\mathrm{cost}(b_3, b_4), \mathrm{cost}(b_5, b_6))$

$\because$

$$\begin{aligned} Treepath(sn_u, b_3) &= Treepath(sn_u, b_4) \\ &\cup Treepath(b_3, b_4) \\ Treepath(sn_v, b_5) &= Treepath(sn_v, b_6) \\ &\cup Treepath(b_5, b_6) \end{aligned}$$

$\therefore$

$$\max(\mathrm{cost}(sn_u, b_3), \mathrm{cost}(sn_v, b_5)) \\ \geq \max(\mathrm{cost}(b_3, b_4), \mathrm{cost}(b_5, b_6)) \geq \mathrm{cost}(b_3, b_5) \tag{7}$$

$\therefore$ (6) and (7)$\Rightarrow \max(\mathrm{cost}(sn_u, b_3), \mathrm{cost}(sn_v, b_5)) \equiv \mathrm{cost}(b_3, b_5)$

$\because$

$$\begin{aligned} Treepath(sn_u, b_1) &= Treepath(sn_u, b_3) \\ &\cup Treepath(b_1, b_3) \\ Treepath(sn_v, b_2) &= Treepath(sn_v, b_5) \\ &\cup Treepath(b_2, b_5) \end{aligned}$$

$\therefore$

$$(5) \Rightarrow \max(\mathrm{cost}(b_3, b_1), \mathrm{cost}(b_5, b_2)) \leq \mathrm{cost}(b_1, b_2) \tag{8}$$

$\therefore \max(\mathrm{cost}(b_3, b_1), \mathrm{cost}(b_5, b_2), \mathrm{cost}(b_1, b_2)) \leq \mathrm{cost}(b_1, b_2)$

$\because Treepath(b_3, b_5) = Treepath(b_3, b_1) \cup Treepath(b_1, b_2) \cup Treepath(b_2, b_5)$

$\therefore \mathrm{cost}(b_3, b_5) = \max(\mathrm{cost}(b_3, b_1), \mathrm{cost}(b_5, b_2), \mathrm{cost}(b_1, b_2)) \leq \mathrm{cost}(b_1, b_2)$

$\because \mathrm{cost}(b_3, b_5) \geq \mathrm{cost}(b_1, b_2)$ $Treepath(b_1, b_2) \subset Treepath(b_3, b_5)$

$\therefore \mathrm{cost}(b_3, b_5) \equiv \mathrm{cost}(b_1, b_2)$

$\because$

**Lemma 3** $\Rightarrow \max(\mathrm{cost}(sn_d, b_3), \mathrm{cost}(sn_d, b_5)) \leq \max(\mathrm{cost}(b_3, b_4), \mathrm{cost}(b_5, b_6))$;

and $Treepath(b_3, b_5) \subseteq (Treepath(sn_d, b_3) \cup Treepath(sn_d, b_5)) \Rightarrow \mathrm{cost}(b_3, b_5) \leq \max(\mathrm{cost}(sn_d, b_3), \mathrm{cost}(sn_d, b_5))$.

$\therefore \mathrm{cost}(b_3, b_5) \leq \max(\mathrm{cost}(b_3, b_4), \mathrm{cost}(b_5, b_6))$

$\therefore$

$$(7) \Rightarrow \max(\mathrm{cost}(b_3, b_4), \mathrm{cost}(b_5, b_6)) \equiv \mathrm{cost}(b_1, b_2) \tag{9}$$

(**Notice:** There exist two cases: 1. $Treepath(sn_d, b_3)$ passes by $\overline{b_1\ b_2}$; 2. $Treepath(sn_d, b_5)$ passes by $\overline{b_1\ b_2}$. It is obvious that the two paths cannot pass by $\overline{b_1\ b_2}$ at the same time. So, we respectively talk about the two cases in the following contents.)

$\because$ Case 1: $Treepath(sn_d, b_3)$ passes by $\overline{b_1\ b_2}$

$\therefore \mathrm{cost}(sn_d, b_3) \geq \mathrm{cost}(b_1, b_2)$

$\because$ **Lemma 3** $\Rightarrow \mathrm{cost}(b_4, b_3) \geq \mathrm{cost}(sn_d, b_3)$

$\therefore \mathrm{cost}(b_4, b_3) \geq \mathrm{cost}(b_1, b_2)$

$\because$ (9) $\Rightarrow \mathrm{cost}(b_4, b_3) \leq \mathrm{cost}(b_1, b_2)$

$\therefore$

$$\mathrm{cost}(b_4, b_3) \equiv \mathrm{cost}(b_1, b_2) \tag{10}$$

$\because$ Given a node collection $V(Treepath(b_3, b_1)) = \{b_3, m_1, m_2 \cdots m_t, b_1\}$, a node $o \in ca$ is subject to $V(Treepath(b_1, b_3)) \cap (V(Treepath(o, m_j))/m_j) = \emptyset$, $j \le t$, we have

$$\begin{aligned} Treepath(sn_d, o) &= Treepath(sn_d, b_1) \\ &\cup Treepath(b_1, m_j) \\ &\cup Treepath(m_j, o)) \\ Treepath(sn_u, o) &= Treepath(sn_u, b_3) \\ &\cup Treepath(b_3, m_j) \\ &\cup Treepath(m_j, o)). \end{aligned}$$

$\therefore$

$$\mathrm{cost}(sn_d, o) = \max(\mathrm{cost}(sn_d, b_1), \mathrm{cost}(b_1, m_j), \mathrm{cost}(m_j, o))$$
$$\mathrm{cost}(sn_u, o) = \max(\mathrm{cost}(sn_u, b_3), \mathrm{cost}(b_3, m_j), \mathrm{cost}(m_j, o))$$

$\because$ Case 1 $\Rightarrow Treepath(b_1, b_2) \subset Treepath(sn_d, b_1)$

$\therefore \max(\mathrm{cost}(sn_d, b_1), \mathrm{cost}(b_1, m_j)) \ge \mathrm{cost}(b_1, b_2)$

$\because$ **Lemma 3** $\Rightarrow \mathrm{cost}(b_4, b_3) \ge \mathrm{cost}(sn_d, b_3) \ge \mathrm{cost}(sn_d, m_j)$

$\therefore$ (10) $\Rightarrow \mathrm{cost}(b_1, b_2) \ge \mathrm{cost}(sn_d, m_j) = \max(\mathrm{cost}(sn_d, b_1), \mathrm{cost}(b_1, m_j))$

$\therefore \max(\mathrm{cost}(sn_d, b_1), \mathrm{cost}(b_1, m_j)) \equiv \mathrm{cost}(b_1, b_2)$

$\therefore \mathrm{cost}(sn_d, o) = \max(\mathrm{cost}(b_1, b_2), \mathrm{cost}(m_j, o))$

$\because$ **Lemma 3** $\Rightarrow \mathrm{cost}(sn_u, b_4) \le \mathrm{cost}(b_3, b_4)$

$\therefore$ (10) $\Rightarrow \mathrm{cost}(sn_u, b_4) \le \mathrm{cost}(b_1, b_2)$

$\because$ (10) $\Rightarrow \mathrm{cost}(b_1, b_2) \ge \mathrm{cost}(b_1, b_3) \ge \mathrm{cost}(b_3, m_j)$

$\therefore \max(\mathrm{cost}(sn_u, b_4), \mathrm{cost}(b_3, b_4), \mathrm{cost}(b_3, m_j)) \equiv \mathrm{cost}(b_1, b_2)$

$\because Treepath(sn_u, b_3) = Treepath(sn_u, b_4) \cup Treepath(b_3, b_4)$

$\therefore \max(\mathrm{cost}(sn_u, b_3), \mathrm{cost}(b_3, m_j)) \equiv \mathrm{cost}(b_1, b_2)$

$\therefore \mathrm{cost}(sn_u, o) = \max(\mathrm{cost}(b_1, b_2), \mathrm{cost}(m_j, o))$

$\therefore \mathrm{cost}(sn_u, o) = \max(\mathrm{cost}(b_1, b_2), \mathrm{cost}(m_j, o))$

$\therefore \mathrm{cost}(sn_d, o) = \max(\mathrm{cost}(b_1, b_2), \mathrm{cost}(m_j, o)) \equiv \mathrm{cost}(sn_u, o)$

$\therefore \forall o \in$ ca, $\ni$ cost $(sn_u, o) \equiv$ cost $(sn_d, o)$

$\therefore$

$T\mathrm{cost}(SN, C') \equiv T\mathrm{cost}(SN)$ holds, where $C' = \{c_1, c_2, c_d/ca, c_{d+1}, c_{d+2}, \cdots c_u \cup ca, c_{u+1}, c_{u+2}, c_k\}$.

$\therefore C' = \{c_1, c_2, c_d/ca, c_{d+1}, c_{d+2}, \cdots c_u \cup ca, c_{u+1}, c_{u+2}, c_k\}$ is an optimal partition.

$\therefore$ The clusters which $B$ contains are whole optimal clusters on Case 1.

$\because$ Case 2: $Treepath(sn_d, b_5)$ passes by $\overline{b_1\ b_2}$.

$\therefore$ Similarly, $C' = \{c_1, c_2, c_d/cb, c_{d+1}, c_{d+2}, \cdots c_v \cup cb, c_{u+1}, c_{u+2}, c_k\}$ is an optimal partition.

$\therefore$ Each optimal cluster in $B$ is whole on Case 2.

$\therefore$ Each optimal cluster which B contains is whole when either $Treepath(sn_d, b_3)$ or $Treepath(sn_d, b_5)$ passes by $\overline{b_1\ b_2}$.

$\therefore$ Each optimal cluster in $B$ is whole when $c_d \ne c_i$.

$c_i = c_d$

When $c_d (= c_v) = c_i$ happens, the related demonstration is given as follows. Similarly, there exist two cases: 1$Treepath(sn_d, b_3)$. passes by $\overline{b_1\ b_2}$; 2. $Treepath(sn_d, b_5)$ passes by $\overline{b_1\ b_2}$. It is obvious that the two paths cannot pass by $\overline{b_1\ b_2}$ at the same time. So, we respectively talk about the two cases in the following contents.

$\because$ Case 1: $Treepath(sn_d, b_3)$ passes by $\overline{b_1\ b_2}$.

$\therefore \mathrm{cost}(s_i, b_3) = \max(\mathrm{cost}(s_i, b_2), \mathrm{cost}(b_1, b_2), \mathrm{cost}(b_1, b_3))$; $\mathrm{cost}(b_1, b_2) \le \mathrm{cost}(s_i, b_3)$

$\because$ **Lemma 3** $\Rightarrow \mathrm{cost}(s_i, b_3) \le \mathrm{cost}(b_3, b_4)$; $\mathrm{cost}(sn_j, b_1) \le \mathrm{cost}(b_1, b_2)$

$\therefore \mathrm{cost}(b_1, b_2) \le \mathrm{cost}(s_i, b_3) \le \mathrm{cost}(b_3, b_4)$; $\mathrm{cost}(b_3, b_4) \le \mathrm{cost}(sn_j, b_1) \le \mathrm{cost}(b_1, b_2)$

$\therefore \mathrm{cost}(b_3, b_4) \equiv \mathrm{cost}(b_1, b_2)$; $\mathrm{cost}(b_3, b_1) \le \mathrm{cost}(b_1, b_2)$

$\therefore$ Similarly, $\forall o \in$ ca, $\ni$ cost $(sn_u, o) \equiv$ cost $(sn_d, o) \equiv N\mathrm{cost}(o)_{SN}$.

$\therefore$

$T\mathrm{cost}(SN, C') \equiv T\mathrm{cost}(SN)$ holds, where $C' = \{c_1, c_2, c_d/ca, c_{d+1}, c_{d+2}, \cdots c_u \cup ca, c_{u+1}, c_{u+2}, c_k\}$.

$\therefore$ Each optimal cluster in $B$ is whole on Case 1.

$\because SC_i$ is obtained according to the first $i$ servers. Then, the partition is subject to $N\mathrm{cost}(o)_{S_i} \equiv \mathrm{cost}(s*, o)$ where $o \in X$ is an arbitrary node and $s* \in S_i$ denotes the given server of $o$.

$\therefore T\mathrm{cost}(S_i) \equiv (\sum_{c_j \in SC_i/sc_i} C\mathrm{cost}(sn_j, c_j)) + C\mathrm{cost}(s_i, sc_i)$ where $S_i = \{sn_1, sn_2 \cdots sn_{i-1}, s_i\}$

$\therefore T\mathrm{cost}(S_i) \equiv T\mathrm{cost}(S_i, SC_i)$

$\because N\mathrm{cost}(o)_{S_i} \equiv \mathrm{cost}(s^*, o)$

$\therefore$

$C\mathrm{cost}(s_i, sc_i) = \sum_{o \in sc_i} N\mathrm{cost}(o)_{S_i} (= C\mathrm{cost}(S_i, sc_i) = \sum_{o \in sc_i} \mathrm{cost}(S_i, o))$ (where in let $\mathrm{cost}(S_i, o) = N\mathrm{cost}(o)_{S_i}$ .).

$\therefore T\mathrm{cost}(S_i, SC_i) \equiv T\mathrm{cost}(\{sn_1, sn_2 \cdots sn_{i-1}, S_i\}, SC_i)$

$\because$ Each $o \in ca$ ( $\in sc_j$ ) is subjected to $\mathrm{cost}(S_i, o) = N\mathrm{cost}(o)_{S_i} \equiv \mathrm{cost}(s*, o)$ where $s* \in S_i$ denotes the given server of $o$.

$\therefore T\mathrm{cost}(\{sn_1, sn_2 \cdots sn_{i-1}, S_i\}, SC_i) \equiv T\mathrm{cost}(\{sn_1, sn_2 \cdots sn_{i-1}, S_i\}, SC_i')$ where $SC_i' = \{sc_1, sc_2 \cdots sc_j/ca \cdots sc_i \cup ca\}$

$\therefore$ Similarly,

$T\mathrm{cost}(\{sn_1, sn_2 \cdots sn_{i-1}, S_i\}, SC_i) \equiv T\mathrm{cost}(\{sn_1, sn_2 \cdots sn_{i-1}, S_i\}, SC_i')$ where $SC_i' = \{sc_1, sc_2 \cdots sc_j/ca, \cdots c_d\}$. ($c_d \equiv c_i = sc_i \cup ca \cup \cdots, sc_i = cb$).

$\therefore$ $T\text{cost}(\{sn_1, sn_2 \cdots sn_{i-1}, S_i\}, SC_i') \equiv T\text{cost}(S_i, SC_i)$

$\because$ $T\text{cost}(S_i) \le T\text{cost}((S_i \cup sn_d)/s_i\})$ where $S_i = \{sn_1, sn_2 \cdots sn_{i-1}, s_i\}$, and $T\text{cost}(\{sn_1, sn_2 \cdots sn_{i-1}, sn_d\}) \le T\text{cost}(\{sn_1, sn_2 \cdots sn_{i-1}, sn_d\}, SC_i')$ (due to $SC_i'/c_d$ is subject to $N\text{cost}(o)_{S_i} \equiv \text{cost}(s*, o)$ and $sn_d$ serves $c_d$.)

$\therefore$ $T\text{cost}(S_i) \le T\text{cost}(\{sn_1, sn_2 \cdots sn_{i-1}, sn_d\}, SC_i')$

$\therefore$ $T\text{cost}(\{sn_1, sn_2 \cdots sn_{i-1}, S_i\}, SC_i') \le T\text{cost}(\{sn_1, sn_2 \cdots sn_{i-1}, sn_d\}, SC_i')$

$\therefore$

$$C\text{cost}(S_i, c_d) \le C\text{cost}(sn_d, c_d) \tag{11}$$

$\because$ $SN$ denotes the optimal medians.

$\therefore$

$$T\text{cost}(SN) \le T\text{cost}(SN') \tag{12}$$

where $SN' = \{sn_1, sn_2 \cdots sn_{i-1}, s_i, sn_{i+1} \cdots sn_k\}$.

$\because$ $T\text{cost}(SN')$ is subject to $N\text{cost}(o)_{SN'} \equiv \text{cost}(s*, o)$, where $s*$ denotes the given server, and $s_i \in c_d$.

$\therefore$ For each $o \in X/c_d$, $N\text{cost}(o)_{SN'} \equiv \text{cost}(s*, o) \equiv N\text{cost}(o)_{SN}$ and $s* \ne s_i$.

(According to **Lemma 3**, $o$ is always closer to its server than other cluster's node.)

$\therefore$ $\sum_{o \in X/c_d} N\text{cost}(o)_{SN'} \equiv \sum_{o \in X/c_d} \text{cost}(s*, o) \equiv \sum_{o \in X/c_d} N\text{cost}(o)_{SN}$ where $s* \ne s_i$ is the given server of $o$.

$\therefore$ (12)$\Rightarrow$ $\sum_{o \in c_d} N\text{cost}(o)_{SN} \le \sum_{o \in c_d} N\text{cost}(o)_{SN'}$.

$\because$ $S_i \subseteq SN'$.

$\therefore$ $\sum_{o \in c_d} N\text{cost}(o)_{SN'} \le \sum_{o \in c_d} N\text{cost}(o)_{S_i}$.

$\therefore$ $C\text{cost}(sn_d, c_d) = \sum_{o \in c_d} N\text{cost}(o)_{SN} \le \sum_{o \in c_d} N\text{cost}(o)_{S_i} = C\text{cost}(S_i, c_d)$.

$\therefore$ (11) $\Rightarrow$ $C\text{cost}(sn_d, c_d) \equiv C\text{cost}(S_i, c_d)$.

$\therefore$

$$T\text{cost}(SN, C) \equiv T\text{cost}(\{sn_1, sn_2 \cdots sn_{d-1}, S_i, sn_{d+1} \cdots sn_k\}, C) \tag{13}$$

where $SN = \{sn_1, sn_2 \cdots sn_k\}$ is an optimal median collection and $C = \{c_1, c_2 \cdots c_k\}$ is the corresponding partition.

$\because$ $T\text{cost}(\{sn_1, sn_2 \cdots sn_{d-1}, S_i, sn_{d+1} \cdots sn_k\}, C) \ge T\text{cost}(SN', C)$ ($C\text{cost}(S_i, c_d) \ge C\text{cost}(SN', c_d)$ due to $S_i \subset SN'$ and $SN' = \{sn_1, sn_2 \cdots sn_{i-1}, s_i, sn_{i+1} \cdots sn_k\}$ ); and $T\text{cost}(SN', C) \ge T\text{cost}(SN')$.)

$\therefore$ $T\text{cost}(\{sn_1, sn_2 \cdots sn_{d-1}, S_i, sn_{d+1} \cdots sn_k\}, C) \ge T\text{cost}(SN')$

$\therefore$ (13)$\Rightarrow$ $T\text{cost}(SN, C) \ge T\text{cost}(SN')$

$\because$ $SN$ denotes optimal medians, and $C$ is the corresponding partition.

$\therefore$ $T\text{cost}(SN, C) \equiv T\text{cost}(SN) \le T\text{cost}(SN')$

$\therefore$ $T\text{cost}(SN) \equiv T\text{cost}(SN, C) \equiv T\text{cost}(SN')$

$\therefore$ $SN' = \{sn_1, sn_2 \cdots sn_{d-1}, s_i, sn_{d+1} \cdots sn_k\}$ is an optimal median collection.

$\therefore$ $s_i$ is an optimal median on Case 2.

$\therefore$ Each optimal cluster in $B$ is whole or $s_i$ is an optimal median when $c_d = c_i$ .

$\because$ It is demonstrated that each optimal cluster in $B$ is whole when $c_d \ne c_i$ .

$\therefore$ Each optimal cluster in $B$ is whole or $s_i$is an optimal median on any case.

$\because$ (When) $B$ contains one or several whole optimal clusters. (Suppose $B$ has $w$ optimal clusters and the $w$ optimal medians are $\{sn_{B1}, sn_{B2} \cdots sn_{Bw}\} \subset SN, w < k$ )

$\therefore$ **Lemma 4** $\Rightarrow$ $s_i$ is one of the w optimal medians when divide B into w optimal regions.

$\therefore$ $s_i \in \{sn_{B1}, sn_{B2} \cdots sn_{Bw}\}$ when $B$ contains one or several whole optimal clusters.

$\therefore$ $s_i$ is one of the k optimal medians on any case.

$\because$ The first $i-1$ servers are optimal medians (the supposition of inductive method), and $s_i$ is also an optimal median.

$\therefore$ The first $i$ servers are optimal medians, that is, $S_k = \{s_1, s_2 ... s_k\}$ minimizes $f(SN) = T\text{cost}(SN)$, $|SN| \equiv k$ .

$\therefore$ **Theorem 2** holds. □

## 4 Clustering algorithm

### 4.1 Connectivity-based optimal partitioning model

Given a data set $X = \{x_1, x_2, \ldots, x_n\}$, $k$ optimal medians $S_k = \{s_1, s_2 ... s_k\}$ can be solved according to **2**. According to the $k$ optimal medians $S_k = \{s_1, s_2 ... s_k\}$, partition $X = \{x_1, x_2, \ldots, x_n\}$ into $k$ clusters $C = \{c_1, c_2 \cdots c_k\}$. The whole process above is an optimal partitioning model (**OPM**) that minimizes the total cost of $C = \{c_1, c_2 \cdots c_k\}$.

The algorithm is as follows:

**Step 1**(Building the tree of $X$): Build the minimum spanning tree(**MST**) according to the Prim algorithm, the Krusakl algorithm or others.

**Step 2**(Computing the cost function of $X$): According to **Algorithm 1** and **MST**, the cost function $CF = \left[\text{cost}\left(x_i, x_j\right)\right]_{n \times n}$ is obtained;

**Step 3**(Solving $K$ optimal medians): According to **Theorem 2** and $CF = \left[\text{cost}\left(x_i, x_j\right)\right]_{n \times n}$, the optimal medians $S_k = \{s_1, s_2 ... s_k\}$ are obtained;

**Step 4**(Connectivity-based clustering): Each node $p$ selects the median with the smallest cost(connectivity) for clustering ($N\text{cost}(p)_{S_k} = \min(\{z \mid z = \text{cost}(o, p), o \in S_k\})$). Partitioning $X = \{x_1, x_2, \ldots, x_n\}$ into $k$ clusters $C = \{c_1, c_2 \cdots c_k\}$ according to $S_k = \{s_1, s_2 ... s_k\}$, and such that the total cost is minimized. Each node always selects the lowest-cost server to obtain service; if a given node has a unique lowest-cost server, then the node and its server are classified into a same cluster; and If a given node has at least two lowest-cost servers, then the best one is determined by the $MST$ built from the given node.

The detailed algorithm is described as follows:

According to the number of the lowest-cost servers of each node, all nodes are divided into the following types.

(1) $\text{Ncost}(x) \equiv \text{cost}(sn_i, x) < \min(\{z \mid z = \text{cost}(o, x), o \in SN/sn_i\})$, that is, $x$ only has one optimal server;

(2) $\text{Ncost}(x) \equiv \text{cost}(sn_i, x) \equiv \min(\{z \mid z = \text{cost}(o, x), o \in SN/sn_i\})$, that is, $x$ has at least two optimal servers.

The first 'for' loop groups (1) into the corresponding clusters; and the second 'for' loop deals with (2). In the second 'for' loop, the $s\_tree$ is built by the Prim algorithm, and the $s\_tree$ stops if a newly merged node has been assigned a cluster label 'i' (Line 9). Next, all nodes of the $s\_tree$ are classified into the $i$-th cluster (Line 10). Finally, we get a partition, $C = \{c_1, c_2 \cdots c_k\}$, where $c_i$ is served by $s_i$.

**Input**:$S_k$, $CF = \left[\text{cost}\left(x_i, x_j\right)\right]_{n\times n}$, $MST$
**Output**: $clusterlb$

```
1:  $clusterlb = zeros(1, n)$; //**initialize cluster labels with 0**.
2:  **for** each $x \in X$ **do**
3:      **if** $\exists s_i \ni \text{cost}(s_i, x) < \min(\{z \mid z = \text{cost}(o, x), o \in S_k/s_i\})$ **then**
4:          $clusterlb(x) = i$; //assign label $'i'$
5:      **end if**
6:  **end for**
7:  **for** each $x_i \in X$ **do**
8:      **if** $clusterlb(x) \equiv 0$ **then**
9:          Start from node $x_i$ to build a minimum spanning tree $s\_tree$ for the graph $MST$ (Prim algorithm). If a newly merged node, $x_j$ is subjected to $clusterlb\left(x_j\right) \neq 0$, then $s\_tree$ stops merging other nodes.
10:         $clusterlb(V(s\_tree))=clusterlb\left(x_j\right)$ //The cluster label of $x_j$ is assigned to all nodes of the $s\_tree$.
11:     **end if**
12: **end for**
```

**Algorithm 2** OPM algorithm.

**Example:** Give a data set $X$ including 2 ($c_1$, $c_2$), clusters an arbitrary minimum spanning tree ($MST$) of $X$ is shown in Fig. 7(1). Suppose the uniquely optimal server of $u$ is $s_1$ and $o$ has two lowest-cost servers (medians), $s_1$ and $s_2$. Given that the uniquely optimal server of $u$ is $s_1$, $u$ is classified into cluster $c_1$ in the first 'for' loop, that is, $classlb(u) = 1$. Meanwhile, the classlb of $o$, $p$, and $q$ nodes are 0 in the first 'for' loop (see Fig. 7(2)). With regard to $o$, the second 'for' loop build a minimum spanning tree $s\_tree$ according to the Prim algorithm and then assign a suitable cluster label. The $s\_tree$ starts from $o$ and stop merging other nodes if the newly-merged node has been assigned a cluster label. Since $u$ has been assigned the cluster label '1', the $s\_tree$ must stop after merging $u$. Then, a sub-tree including $o$, $q$, $p$ and $u$ is obtained. Finally, $o$, $q$ and $p$ are assigned the cluster label $classlb(u)$, that is, $o$, $q$, $p$ are classified into $u's$ cluster in the second 'for' loop (see Fig. 7(3)).

The current partitioning algorithms derive from two clustering models: the $k$-means and $k$-medoids models. The representative algorithms are $k$-means algorithm and Partitioning Around Mediods (**PAM**). In fact, these algorithms are not the complete mappings of their models. Considering those models are NP-hard problems, the corresponding clustering algorithms cannot guarantee that their clustering results minimize the total cost functions. In other words, the current algorithms are incapable to ensure the optimal clustering result in theory. On the contrary, our algorithm obtains the optimal partition. On the one hand, the partition minimizes the total cost function; on the other hand, it is subjected to Constraints 1 and 2 (**see Proof 4**). Hence, our algorithm is a perfect mapping of the equivalent $k$-median problem and guarantees that the final partition is theoretically optimal.

The first 'for' loop means $\text{O}(k^*n)$ time in the OPM algorithm. With respect to the second 'for', the running time is less than $O\left(n^2\right)$. The time of building a heap is $O(n)$. If $m$ $(< n)$ nodes is not grouped in the first 'for', the total time of building heaps is at most $\text{O}(n^*m)$ in the Line 9. Since the input $MST$ is a minimum spanning tree, the number of each node's edges is $O(1)$. Then, the total time of modifying heaps is $\text{O}(m * \lg(n))$. In general, only a few of the $m$ nodes need to build a heap, implying the total time of building heaps in the best case is $O(n)$. Hence, the second 'for' costs $\text{O}(n + m^* \lg(n))$ time (The time complexity in the worst case is $\text{O}(n^*m)$. Thus, the running time of the proposed algorithm is $\text{O}(k^*n + m^* \lg(n))$.

In Table 1, the time complexity of the four steps corresponding to the entire algorithm process is shown. The step3 has the highest time complexity and determines the overall time $\text{O}\left(k^*n^2\right)$.

### 4.2 Validate the minimum total cost

Generally, heuristic methods are incapable to find the optimal solution. Hence, readers may doubt that the $k$ optimal servers do not necessarily minimize the total cost. Therefore, in addition to the theoretical demonstration, we conduct actual experiments to verify the minimum total

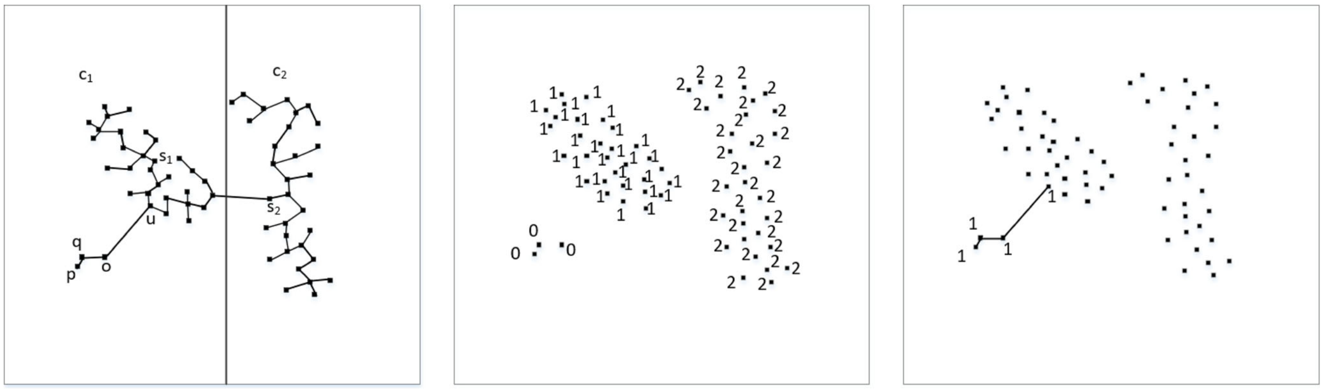

**Fig. 7** Example

cost. $SN' = \underset{SN,|SN|\equiv k}{\arg\min}\ T\text{cost}(SN)$ can be obtained by using the exhaustion method. Then, the minimum total cost $T\text{cost}(S_k, C)$ can be verified by computing $T\text{cost}K = T\text{cost}(SN')$ (**Algorithm 3**) .

The minimum total cost can be computed by the exhaustion algorithm and then our algorithm can be verified. The validation process is a NPC problem and the running time is O $\left(k^* C_n^k\right)$. Hence, the validation experiments cannot be carried out when the scale of a given data set and $k$ are very large.

**Input**: $CF = \left[\text{cost}\left(x_i, x_j\right)\right]_{n\times n}$, $X = \{x_1, x_2, \ldots, x_n\}$
**Output**: $TcostK$

```
1: $TcostK = inf$
2: for each combination $SN' = \{sn_1, sn_2 \ldots sn_k\} \subset X$ do
3:     $tmin = \sum_{o\in X} \min(\text{cost}(sn_1, o), \text{cost}(sn_2, o) \ldots \text{cos}\,t\,(sn_k, o))$
4:     $TcostK = \min(tmin, TcostK)$
5: end for
```

**Algorithm 3** Exhaustion algorithm.

## 4.3 Proof 4 – Algorithm

**Lemma 5** *There are 3 nodes o, p, q such that* $\max(Treepath(o, p)) < \max(Treepath(o, q))$. *Then, in the process of building a minimum spanning tree from o, p would be earlier merged into the tree than q.*

*Proof of Lemma 5* According to Lemma 3, $\max(Treepath(a, b))$ keeps the same in different minimum spanning trees. Hence, $\max(Treepath(a, b))$ always denotes the weight of the longest edge of $Treepath(a, b)$ .

According to the Prim algorithm, we can build a minimum spanning tree $MST$ from $o$. Let $G = Treepath(o, q) \cup Treepath\,(o, p)$ where $Treepath\,(o, q)\,MST$ and $Treepath\,(o, p)\,MST$. In $G$, we build a new minimum spanning tree from $o$ according to the Prim algorithm. Because $\max(Treepath(o, p)) < \max(Treepath(o, q))$ holds, the longest edge of $Treepath(o, p)$ is shorter than $Treepath(o, p)'s$ longest edge. The Prim algorithm always use a short edge to determine the next merged node, which is not longer than $Treepath(o, p)'s$ longest edge. Hence, o and q would be connected before selecting 's longest edge. Similarly, in the process of constructing $MST$, $o$ and $q$ has been connected before selecting $Treepath(o, p)'s$ longest edge. So, $p$ would be earlier merged into $MST$ than $q$. □

*Proof of Algorithm 2* **Validation:** On the one hand, the final result must be subject to Constraints 1 and 2; on the other hand, the final result must minimize the total cost function.

In the proposed algorithm, the first type of nodes (the first 'for' loop) is the special case of the second type of nodes in

**Table 1** Time complexity

| | Best | Worst | Average |
|---|---|---|---|
| Step1 | O $(E + n\log n)$ | O $(n^2)$ | O $(n^2)$ |
| Step2 | O $(n^2)$ | O $(n^2)$ | O $(n^2)$ |
| Step3 | O $(k^*n^2)$ | O $(k^*n^2)$ | O $(k^*n^2)$ |
| Step4 | O $(k^*n)$ | O $(k^*n + m^*n)$ | O $(k^*n + m^*\lg(n))$ |

essence. Therefore, we only validate the second 'for' loop is subject to Constraints 1 and 2.

Given two nodes $x_i$, $x_j \in c_u$, an optimal median, $sn_u \in c_u$, suppose $\mathrm{cost}(x_i, x_j) < N\mathrm{cost}(x_i)_{SN} \equiv N\mathrm{cost}(x_j)_{SN}$. Then, according to the proposed algorithm, we would built a minimum spanning tree from $x_i$ (or $x_j$) Due to $\mathrm{cost}(x_i, x_j) < N\mathrm{cost}(x_i)_{SN} \equiv \mathrm{cost}(sn_u, x_i)$, $\max(Treepath(x_i, x_j)) < \max(Treepath(sn_u, x_i))$ holds. Then, according to **Lemma 5**, $x_j$ would be earlier merged into the minimum spanning tree (containing $x_j$ ) than $sn_u$. In the proposed algorithm, the minimum spanning tree stops when it meets an node served by $sn_u$, and then the tree is classified into $sn_u's$ cluster. Hence, $x_i$ and $x_j$ are classified into a same cluster. So, the algorithm is subject to Constraint 1.

Given a node $p$, it costs the same and lowest cost in several different medians. Let $SNC = \{snc_1, snc_2 \cdots snc_l\}$ denote those medians, and then

$$\begin{aligned} &N\mathrm{cost}(p) = \max(Treepath(snc_1, p)) \\ &\equiv \max(Treepath(snc_2, p)) \cdots \\ &\equiv \max(Treepath(snc_l, p)) \\ &< \max(Treepath(o, p)), o \in SN/SNC \end{aligned}$$

holds. According to **Lemma 5**, if a minimum spanning tree $sub\_tree$, built from $p$, stops merging other nodes till a median is merged, $p$ would firstly reach a median of $SNC$ instead of any one of $SN/SNC$. Then, if we build a new minimum spanning tree $s\_tree$ from any node $p$ of $sub\_tree$, the new tree always firstly reach the same median. If $|SNC| \equiv 1$ (implying $s\_tree = p$), $p$ has a uniquely optimal medians. Then, $p$ uniquely belongs to that optimal median. So, the $s\_tree$ uniquely belongs to the median's cluster and locates in the tree $sub\_tree$. If $|SNC| > 1$, the minimum spanning tree $s\_tree$, built from $p$, stops till it meets a node $q$ which has a unique optimal median. It is obvious that the $s\_tree$ is a subset of $sub\_tree$. According to the proposed algorithm, the tree $s\_tree$ is classified into $q's$ cluster. So, each cluster is a sub-tree, that is, the algorithm is subject to Constraint 2.

According to **Lemma 5**, the $s\_tree$ always firstly reach its optimal median. Meanwhile, each node selects the firstly-meeting median $sn$ to obtain service. Hence, $\mathrm{cost}(sn, p) = \min(\{z \mid z = \mathrm{cost}(o, p), o \in SN\}) \equiv \mathrm{Ncost}(p)_{SN}$ holds. Then, $T\mathrm{cost}(SN, C') \equiv T\mathrm{cost}(SN)$ where $C'$ is obtained by the proposed algorithm. In **Theorem 2**, $S_k$ minimizes $f(SN) = T\mathrm{cost}(SN)$. So, the proposed algorithm obtains the minimum total cost $T\mathrm{cost}(S_k, C')$. □

## 5 Experiments and analysis

The proposed method is able to obtain the optimal partition in theory. Now, we would observe the performance in real applications. In this section, the **distance matrix** is taken as the **SDM**. NMI and ARI are taken as the evaluation criterion of clustering performance. For comparing to the current partitioning models (the $k$-means and the $k$-medoids models), $k$-means, $k$-medoids and normalized spectral clustering algorithms are used in Sections 5.1 and 5.2 has compared K-MEANS [37], RNN-DBSCAN [45], LDP-MST [47], GADPC [48] and OPM.

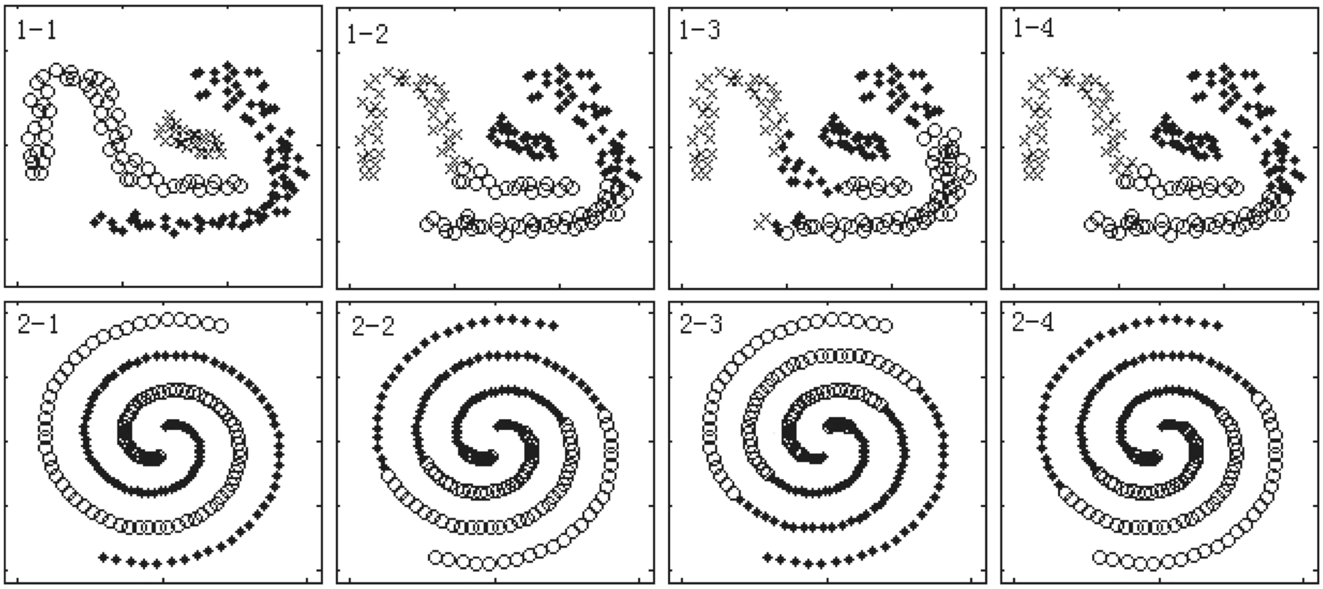

**Fig. 8** Group irregular clusters. OPM algorithm: 1-1,2-1; $k$-means algorithm: 1-2,2-2; $k$-medoids algorithm: 1-3,2-3; normalized spectral clustering algorithm: 1-4,2-4

## 5.1 Experiments for rationality analysis

Because OPM is a clustering algorithm based on minimum spanning tree, it can hadle complex manifold data. Then, we give a set of synthetic data sets as shown in Fig. 8 and then observe the different clustering results. Obviously, OPM displays outstanding performance in grouping irregular clusters (see Fig. 8 **(1-1,2-1)**). However, other algorithms ($k$-means, $k$-medoids and normalized spectral clustering algorithms) give wrong results. In general, a single cluster covers a continuous region, that is, each node of a single cluster has high connectivity to others intra-cluster nodes. Hence, some far nodes may belong to a same cluster owing to their high connectivity. Since the $k$-means and the $k$-medoids models only consider the absolute distance and neglect the connectivity, they cannot correctly partition the far and connected nodes. As for OPM algorithm, the proposed cost function reflects the connectivity. Then, if two nodes are connected via some intermediate nodes, they are classified into a same cluster. Hence, our algorithm is able to detect arbitrarily-shaped clusters.

From the above results, we can see other partitioning algorithms ($k$-means, $k$-medoids, normalized spectral clustering) cannot deal with manifold data sets. On the contrary, OPM can get satisfactory clustering results for the data sets which contain complex-manifold clusters. That is, OPM is a better partitioning algorithm for manifold data sets.

In addition, based on the clustering theory above, the minimum total cost of $k$-median problem is obtained. Thus, the clustering results can ensure that minimum total cost and theoretical optimal partition.

## 5.2 Experiments for performance evaluations

Next, we do experiments on real data sets and compare the clustering performance according to NMI and ARI. For the sake of evaluating the clustering effect of OPM, it is compared with K-MEANS [37], RNN-DBSCAN [45], LDP-MST [47] and GADPC [48]. The data sets come from UCI website and joensuu.fi (https://cs.joensuu.fi/sipu/datasets/). In the two tables (Tables 2 and 3), some of the results are from the related references. Meanwhile, some experiments are supplemented in the whole data and we try to get high scores by adjusting the parameters. In addition, all experiments need to specify the number of clusters $k$.

The following analysis of the OPM algorithm is based on the three common problems summarized previously. The results of the analysis of the algorithm are given in Table 4. The only parameter in the OPM is the number of clusters $k$, which needs to be given in advance. The optimal solution model ensures that the OPM can obtain the theoretically optimal solution. Then, the new $k$-median model and the corresponding OPM algorithm is suited to cope with arbitrarily shaped clusters. Finally, OPM shows

**Table 2** NMI evaluation

| | K-MEANS | RNN-DBSCAN | LDP-MST | GADPC | OPM |
|---|---|---|---|---|---|
| Synthetic data sets | | | | | |
| Spiral | 3.7751e-04 | **1** | **1** | **1** | **1** |
| R15 | 0.9281 | 0.9876 | 0.9913 | **0.9942** | 0.9893 |
| Pathbased | 0.5493 | 0.6345 | 0.52015 | — | **0.7311** |
| Jain | 0.3571 | 0.9570 | **1** | **1** | **1** |
| Flame | 0.3941 | 0.9002 | 0.9269 | **1** | **1** |
| D31 | 0.9257 | 0.8936 | **0.9647** | — | 0.9573 |
| Compound | 0.6640 | 0.8408 | 0.8418 | **0.9122** | 0.9071 |
| Real-world data sets | | | | | |
| Wine | 0.4241 | 0.3789 | **0.4327** | — | 0.3982 |
| Seeds | 0.6949 | 0.3133 | 0.5418 | **0.6982** | 0.6605 |
| Iris | 0.7582 | 0.6667 | 0.8058 | 0.8057 | **0.8705** |
| Segment | 0.4903 | 0.6521 | 0.6083 | — | **0.6952** |
| Control | 0.6846 | 0.7837 | 0.8 | — | **0.8092** |
| Sonar | 0.0058 | 0.0318 | 3.9620e-05 | — | **0.0615** |
| Abalone | **0.1582** | 0.0790 | 0.0823 | — | 0.1553 |
| Banknote | 0.0303 | **0.6289** | 0.1286 | — | 0.6111 |
| Yeast1 | 0.0797 | **0.1291** | 0.0466 | — | 0.1097 |

Bold means the highest score

**Table 3** ARI evaluation

| | K-MEANS | RNN-DBSCAN | LDP-MST | GADPC | OPM |
|---|---|---|---|---|---|
| Synthetic data sets | | | | | |
| Spiral | −0.0060 | **1** | **1** | **1** | **1** |
| R15 | 0.8169 | 0.9823 | 0.9891 | **0.9928** | 0.9857 |
| Pathbased | 0.4642 | 0.5786 | 0.4291 | — | **0.6133** |
| Jain | 0.3004 | 0.9804 | **1** | **1** | **1** |
| Flame | 0.4312 | 0.9550 | 0.9666 | **1** | **1** |
| D31 | 0.8267 | 0.7875 | **0.9484** | — | 0.9353 |
| Compound | 0.1965 | **0.8582** | 0.8269 | 0.8513 | 0.8577 |
| Real-world data sets | | | | | |
| Wine | 0.3518 | 0.3603 | **0.3627** | — | 0.2534 |
| Seeds | **0.7166** | 0.1989 | 0.4010 | 0.6982 | 0.6886 |
| Iris | 0.7302 | 0.6448 | 0.7592 | 0.8057 | **0.8858** |
| Segment | 0.2832 | 0.5213 | 0.3874 | — | **0.5855** |
| Control | 0.5065 | 0.6065 | 0.6143 | — | **0.6184** |
| Sonar | **0.0011** | −0.0045 | −0.0040 | — | −0.0045 |
| Abalone | **0.1501** | 0.0212 | 0.0095 | — | 0.1137 |
| Banknote | 0.0485 | 0.5534 | −0.0021 | — | **0.6262** |
| Yeast1 | 0.0189 | 0.0039 | 0.0124 | — | **0.0928** |

Bold means the highest score

a good clustering effect on manifold data sets according to Tables 2 and 3.

The performance of RNN-DBSCAN is dependent on the choice of $k$. Therefore, its performance is stable with respect to $k$ as evidences by the heuristic for choosing $k$. LDP-MST tunes for several times to get the best clustering results, which enables its superiority on clustering data sets with complex structured clusters and large number of noise nodes. GADPC provides detection measures for outliers. Then, GADPC has good adaptability in parameter sensitivity. In addition, the algorithms above may lead to unstable results due to hyperparameters and different initialization, but in general they are stable. OPM shows stable performance due to no hyperparameters. In addition, the Sonar data set and the Abalone data set represent aggregated structures whose data are not separated well. Thus, all algorithms exhibit non-ideal results in this type of data set. In summary, the performance of OPM is relatively ideal and equally stable on suitable clustered data.

In Tables 2 and 3, most of the clustering scores of OPM are significantly higher than others. However, in some data sets, OPM shows an ordinary performance. In fact, some of the data sets above have fuzzy boundaries, and their clusters even overlap. For convenience, we transform the seeds data set into a 2-D data set (see Fig. 9(1)). The three clusters of the seeds data set overlap with one another and the bounds of the clusters are ambiguous. Obviously, this distribution is not subjected to Constraints 1 and 2. Thus, the OPM exhibits an ordinary performance in partitioning the seeds data set. By contrast, the clusters of the iris data set have clear boundaries, i.e., the iris data set is significantly subjected to Constraints 1 and 2. Hence, the OPM obtains high scores. If we separate the 3 clusters like Fig. 9(2), the distribution completely satisfies Constraints 1 and 2. Then, OPM algorithm can get a 100% correct partition. In

**Table 4** Performance evaluations

| | No hyperparameters | Minimizing total cost | Good results |
|---|---|---|---|
| K-MEANS | ✓ | ✓ | |
| RNN-DBSCAN | | | |
| LDP-MST | ✓ | | ✓ |
| GADPC | | | ✓ |
| OPM | ✓ | ✓ | ✓ |

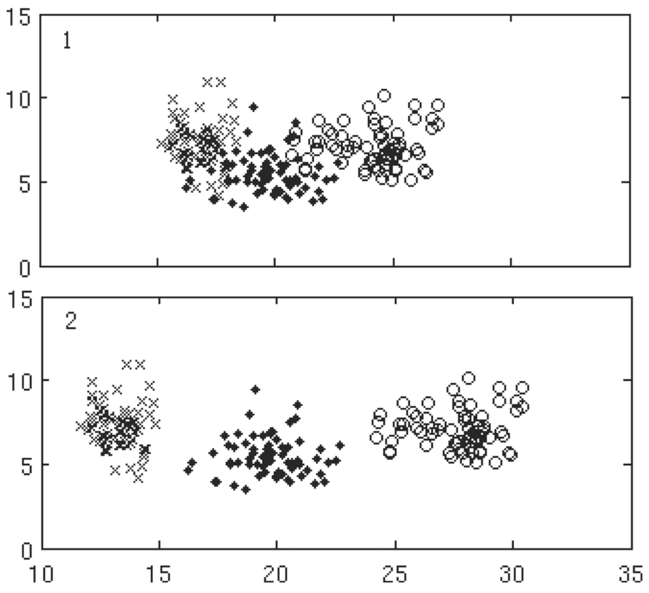

**Fig. 9** The seeds data sets

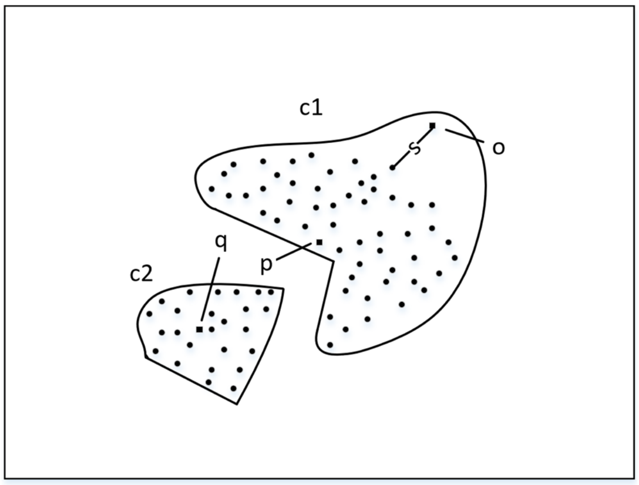

**Fig. 10** Partition with outliers

the following contents, Constraints 1 and 2 are transformed into Lemma 3. In short, the constraints mean that the intra-cluster connectivity must be higher than the connectivity between clusters. Then, if a data set satisfies this constraint to a great extent, OPM can get the realistically optimal partition.

## 5.3 Weakness analysis

The goal of clustering is to obtain the optimal partition of reality. If the total cost of the real optimal partition is minimal, then this partition is also the theoretical optimal partition. The new $k$-median model proposed in this paper can solve the real optimal partition correctly. If the total cost of the real optimal partition is not the minimum, the model will not get the real optimal partition, which is contrary to the clustering goal. When there are outliers, the theoretical optimal partition and the real optimal partition may not be the same partition, and the model fails.

In our model, the smallest total cost is not necessarily the optimal partition when outliers occur. In the case of outliers, the total generation value under the ideal partition is not necessarily the minimum. The following is a partition diagram with outliers and proves that the total cost under the ideal partition is not the minimum.

As shown in the Fig. 10, there are two clusters $c1$ and $c2$ obtained by the ideal partition, and the corresponding medians are $p$ and $q$ (**Theorem 2** and **Algorithm 2**), where there is an outlier $o$, and $s$ represents the distance between the outlier $o$ and the nearest node in the $c1$ cluster.

$\because$ According to **Theorem 2** and **Algorithm 2**, the optimal partition $C$={$c1, c2$} is obtained, and the optimal medians are $p$ and $q$.

$\therefore$ This gives the minimum total cost $Total\text{cost}(C) = \text{cost}(c1) + \text{cost}(c2)$, where $\text{cost}(c1) = \text{cost}(p, o) + \sum_{x\varepsilon(c1/o)} \text{cost}(p, x)$, and $\text{cost}(c2) = \sum_{x\varepsilon c2} \text{cost}(q, x)$.

$\because$ S represents the distance of the outlier $o$ from the nearest node in the $c1$ cluster.

$\therefore \text{cost}(p, o) = s$.

$\therefore Total\text{cost}(C) = s + \sum_{x\varepsilon(c1/o)} \text{cost}(p, x) + \sum_{x\varepsilon c2} \text{cost}(q, x)$.

$\because$ When $o$ is an outlier and $s$ is large enough, the optimal partition $C' = \{o, (c1/o) + c2\}$ and the optimal medians $o$ and $p$ are obtained according to **Theorem 2** and **Algorithm 2**.

$\therefore$ At this time, $Total\text{cost}(C') = \sum_{x\varepsilon\{(c1/o)+c2\}} \text{cost}(p, x) + \text{cost}(o, o)$, where $\text{cost}(o, o) = 0$.

$\because$ When $o$ is an outlier and $s$ is large enough.

$\therefore s + \sum_{x\varepsilon(c1/o)} \text{cost}(p, x) + \sum_{x\varepsilon c2} \text{cost}(q, x) > \sum_{x\varepsilon\{(c1/o)+c2\}} \text{cost}(p, x)$.

$\therefore Total\text{cost}(C) > Total\text{cost}(C')$.

$\therefore$ The ideal partition and the minimum total cost cannot be guaranteed when there are outliers.

The noise influences the clustering results, it is recommended to remove these noise nodes. Of course, not all noise nodes have an impact on the algorithm. When a sub-cluster is large enough or not particularly small, outliers that are not very far from this sub-cluster do not affect the selection of the medians of this sub-cluster. The above influence is very small, which means that the outlier error is ignored in this sub-cluster.

## 6 Conclusion

A $k$-median problem based on connectivity and the corresponding clustering algorithm are proposed in this study. Since the $k$-median problem based on connectivity takes the connectivity to measure the service cost, the far and connected nodes would be classified into the same cluster. Hence, the corresponding clustering algorithm is suited to partition arbitrarily-shaped clusters. Based on experiments using synthetic and actual data sets, the outstanding performance of the proposed algorithm is verified. Meanwhile, the applied scope is also analyzed, and an easy-to-understand constraint is provided as a guide for applying the algorithm: The intra-cluster connectivity must be higher than the connectivity between clusters.

On the one hand, the $k$-median problem based on connectivity is solvable in polynomial time; on the other hand, the corresponding clustering algorithm is a perfect mapping of the $k$-median problem based on connectivity. Hence, the clustering result is an optimal partition of the $k$-median problem based on connectivity in theory. To the best of our knowledge, the current partitioning algorithms are not strictly subjected to their partitioning models. The $k$-means and $k$-medoids algorithms cannot guarantee that their results would converge to the globally optimal solutions. In this study, we successfully obtain the globally optimal solution within polynomial time. Hence, the new $k$-median model provides the perfect theoretical support for the research on partitioning problems.

**Acknowledgements** This research was funded by National Science Foundation of China, No.61976158.

**Data Availability** Publicly available data sets were analyzed in this study. This data can be found here: http://archive.ics.uci.edu/ml/datasets.php.

## Declarations

**Competing interests** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

## References

1. Zhu X, Gan J, Lu G, Li J, Zhang S (2020) Spectral clustering via half-quadratic optimization. World Wide Web 23(3):1969–1988
2. Kang Z, Zhao X, Peng C, Zhu H, Zhou JT, Peng X, Chen W, Xu Z (2020) Partition level multiview subspace clustering. Neural Netw 122:279–288
3. Belhaouari SB, Ahmed S, Mansour S (2014) Optimized k-means algorithm. Math Probl Eng 2014
4. Ahmed M (2018) Collective anomaly detection techniques for network traffic analysis. Ann Data Sci 5(4):497–512
5. Ahmed M (2017) An unsupervised approach of knowledge discovery from big data in social network. EAI Endorsed Trans Scalable Inf Syst 4(14):3
6. Ahmed M (2018) Collective anomaly detection techniques for network traffic analysis. Ann Data Sci 5(4):497–512
7. Tondini S, Castellan C, Medina MA, Pavesi L (2019) Automatic initialization methods for photonic components on a silicon-based optical switch. Appl Sci 9(9):1843
8. Zhang X, He Y, Jin Y, Qin H, Azhar M, Huang JZ (2020) A robust k-means clustering algorithm based on observation point mechanism. Complexity 2020
9. Hale TS, Moberg CR (2003) Location science research: a review. Ann Oper Res 123(1):21–35
10. Hakimi SL (1964) Optimum locations of switching centers and the absolute centers and medians of a graph. Oper Res 12(3):450–459
11. Kariv O, Hakimi SL (1979) An algorithmic approach to network location problems. i: the p-centers. SIAM J Appl Math 37(3):513–538
12. Liao H, Hu J, Li T, Du S, Peng B (2022) Deep linear graph attention model for attributed graph clustering. Knowl-Based Syst 21:246
13. Guo W, Wang W, Zhao S, Niu Y, Zhang Z, Liu X (2022) Density peak clustering with connectivity estimation. Knowl-Based Syst 243:108501
14. Hadi AS (2022) A new distance between multivariate clusters of varying locations, elliptical shapes, and directions. Pattern Recognition: The Journal of the Pattern Recognition Society 129
15. Geng X, Tang H (2020) Clustering by connection center evolution. Pattern Recogn 98:107063
16. Lin G-H, Xue G (1998) K-center and k-median problems in graded distances. Theor Comput Sci 207(1):181–192
17. Hartmanis J (1982) Computers and intractability: a guide to the theory of np-completeness (Michael R. Garey and David S. Johnson). Siam Review 24(1):90
18. Rana R, Garg D (2009) Heuristic approaches for k-center problem. In: 2009 IEEE international advance computing conference, IEEE, pp 332–335
19. Friedler SA, Mount DM (2010) Approximation algorithm for the kinetic robust k-center problem. Comput Geom 43(6–7):572–586
20. Contardo C, Iori M, Kramer R (2019) A scalable exact algorithm for the vertex p-center problem. Comput Oper Res 103:211–220
21. Plesník J (1987) A heuristic for the p-center problems in graphs. Discret Appl Math 17(3):263–268
22. Shmoys DB (1995) Computing near-optimal solutions to combinatorial optimization problems. Comb Optim 20:355–397
23. Dyer ME, B AMFA (1985) A simple heuristic for the p-centre problem. Oper Res Lett 3(6):285–288
24. Gonzalez TF (1985) Clustering to minimize the maximum intercluster distance. Theor Comput Sci 38:293–306
25. Hochbaum DS, Shmoys DB (1985) A best possible heuristic for the k-center problem. Math Oper Res 10(2):180–184
26. Charikar M, Li S (2012) A dependent lp-rounding approach for the k-median problem. In: International colloquium on automata, languages, and programming, Springer, pp 194–205
27. KAUFMAN L (1990) Finding groups in data. An Introduction to Cluster Analysis 230–234
28. Charikar M, Guha S, Tardos É, Shmoys DB (2002) A constant-factor approximation algorithm for the k-median problem. J Comput Syst Sci 65(1):129–149
29. Jain K, Vazirani VV (2001) Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. J ACM (JACM) 48(2):274–296
30. Charikar M, Li S (2012) A dependent lp-rounding approach for the k-median problem. In: International colloquium on automata, languages, and programming, Springer, pp 194–205
31. Li S, Svensson O (2016) Approximating k-median via pseudo-approximation. SIAM J Comput 45(2):530–547
32. Chrobak M, Kenyon C, Young N (2006) The reverse greedy algorithm for the metric k-median problem. Inf Process Lett 97(2):68–72

33. Meyerson A, O'callaghan L, Plotkin S (2004) A k-median algorithm with running time independent of data size. Mach Learn 56(1):61–87
34. Mettu RR, Plaxton CG (2003) The online median problem. SIAM J Comput 32(3):816–832
35. Fotakis D (2006) Incremental algorithms for facility location and k-median. Theor Comput Sci 361(2-3):275–313
36. Vigneron A, Gao L, Golin MJ, Italiano GF, Li B (2000) An algorithm for finding a k-median in a directed tree. Inf Process Lett 74(1-2):81–88
37. Macqueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proc. Fifth berkeley symposium on math. Stat. and prob
38. Ostrovsky R, Rabani Y (2002) Polynomial-time approximation schemes for geometric min-sum median clustering. J ACM (JACM) 49(2):139–156
39. Kumar A, Sabharwal Y, Sen S (2010) Linear-time approximation schemes for clustering problems in any dimensions. J ACM (JACM) 57(2):1–32
40. Wu Z, Leahy R (1993) An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. IEEE Trans Pattern Anal Mach Intell 15(11):1101–1113
41. Zhu X, Gan J, Lu G, Li J, Zhang S (2020) Spectral clustering via half-quadratic optimization. World Wide Web 23(3):1969–1988
42. Von Luxburg U (2007) A tutorial on spectral clustering. Stat Comput 17(4):395–416
43. Wan J, Zhu Q, Lei D, Lu J (2015) Outlier detection based on transitive closure. Intell Data Anal 19(1):145–160
44. Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press
45. Bryant A, Cios K (2018) Rnn-dbscan: a density-based clustering algorithm using reverse nearest neighbor density estimates. IEEE Trans Knowl Data Eng 30(6):1109–1121
46. Rodriguez A, Laio A (2014) Clustering by fast search and find of density peaks. Science 344(6191):1492
47. Cheng D, Zhu Q, Huang J, Wu Q, Yang L (2021) Clustering with local density peaks-based minimum spanning tree. IEEE Trans Knowl Data Eng 33(2):374–387
48. Xu T, Jiang J (2022) A graph adaptive density peaks clustering algorithm for automatic centroid selection and effective aggregation. Expert Syst Appl 195:116539

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.