A Reduced-Order Backtracking Algorithm for the P-Center Location Problem Postprint

Authors: Shang Chunjian, Ning Aibing, Peng Dajiang, Zhang Huizhen

Date: 2020-09-28T00:00:00+00:00

Abstract

In the field of operations research, numerous solution models exist for the emergency service facility location problem. This paper selects the P-center model for investigation. First, the mathematical properties of this problem are studied and proofs are provided. These mathematical properties can be utilized to reduce the problem' s complexity and thereby decrease its scale. Subsequently, based on these properties, a backtracking algorithm based on upper and lower bounds is designed to solve the problem. Finally, through an example analysis, the principle of the algorithm is further elaborated, and it is demonstrated that the algorithm can obtain the optimal solution within a relatively short time frame.

Full Text

Preamble

Volume 38, Issue 3
Application Research of Computers
ChinaXiv Partner Journal
Vol. 38 No. 3 Accepted Paper

A Reduction Backtracking Algorithm for the P-Center Location Problem

Shang Chunjian, Ning Aibing, Peng Dajiang, Zhang Huizhen
(Business School, University of Shanghai for Science & Technology, Shanghai 200093, China)

Abstract: The emergency service facility location problem in operations research has many solution models. This paper selects the P-center model for study. First, we investigate the mathematical properties of this problem and provide proofs. These properties can reduce the problem' s order and thereby

shrink its scale. Based on this, we design a backtracking algorithm that employs upper and lower bounds. Finally, through an example analysis, we further elaborate on the algorithm' s principles and demonstrate that it can obtain the optimal solution in a relatively short time.

Keywords: facility location problem; p-center model; reduction algorithm; upper bound; lower bound; backtracking algorithm

Chinese Library Classification: TP
doi: 10.19734/j.issn.1001-3695.2020.04.0057

Backtracking algorithm with reduction for p-center location problem

Shang Chunjian, Ning Aibing, Peng Dajiang, Zhang Huizhen
(Business School, University of Shanghai for Science & Technology, Shanghai 200093, China)

Abstract: There are various solution models for the emergency service facility location problem and this paper studies the P-center model. This paper proposed a backtracking algorithm with upper and lower bound, and the algorithm added a reduction algorithm by studying the mathematical properties. The algorithm can decrease the scale and the degree of complexity of the problem, so as to accelerate the execution speed. At the end of the paper, this paper illustrated an instance to elaborate this algorithm further.

Key words: facility location problem; p-center model; reduction algorithm; upper bound; lower bound; backtracking algorithm

0 Introduction

Various emergency incidents in daily life have caused tremendous impacts on people's safety and livelihood. When disasters occur, emergency service facilities are the most basic and important infrastructure for providing rescue support and ensuring public safety, playing a crucial role.

Most emergency facility location problems are NP-hard problems with multiple models to address them. For the fairness requirement in emergency facility location, this paper selects the P-center model [1] for study. This model is a special type of discrete location model that involves selecting at most p facilities from n candidate facilities, where each emergency point e_i is served by the nearest selected facility. The objective is to minimize the maximum service distance between all emergency points and their serving facilities—that is, to minimize the maximum distance between all emergency points and their service facilities. In this model, facilities have no capacity constraints.

Many scholars have dedicated research to this location model. Some combine graph theory knowledge to solve the problem model. For example, Hockbaum [2] proposed a 2-opt algorithm with time complexity $O(E \log E)$ for the graph vertex multi-center problem, while Dyer [3] proposed a 2-opt heuristic algorithm with time complexity $O(np)$. Other scholars have applied fuzzy set theory to

solve the P-center problem. For instance, Wen [4] considered fuzzy demands when constructing facility location models to achieve complete coverage in emergency facility location. Some scholars have used heuristic search methods. For example, Domschke et al. [5] employed a binary search approach to solve the r-covering problem with center value equal to p. Zhu et al. [6] combined the analytic hierarchy process with a greedy drop heuristic algorithm to solve the p-center model for aviation rescue services. Davidović et al. [7] used a bee colony algorithm to solve the P-center problem. Xu et al. [8] combined an improved K-Means algorithm with the center-of-gravity method to solve the P-median problem. Huang et al. [9] proposed a method for solving the weighted distance continuous K-center location problem. Other scholars have studied exact algorithms for the P-center problem. For example, Sandoval et al. [10] proposed an exact algorithm based on integer programming models, while Contardo et al. [11] introduced a scalable relaxation-based iterative algorithm design for an exact algorithm to solve the P-center problem.

The advantages of heuristic and approximation algorithms for this problem are their ability to quickly obtain feasible solutions, but their drawback is the tendency to fall into local optima [12, 13]. The advantage of exact algorithms is their ability to obtain optimal solutions, but their disadvantage is that they are not suitable for large-scale problems and have high time complexity [14]. To address these shortcomings, this paper designs an exact algorithm. First, we study the mathematical properties of the P-center model and provide corresponding mathematical proofs. These properties can batch-determine certain facilities to be opened or not opened, thereby reducing the problem scale. Finally, we design a backtracking algorithm based on upper and lower bounds that can obtain the optimal solution while utilizing mathematical properties to accelerate the solution process.

1 Problem Description

This paper uses a bipartite graph $G = (E, F)$ to represent the problem. $E = \{e_i \mid i = 1, 2, \cdots, m\}$ denotes the set of all emergency points, where e_i represents the i-th emergency point and m represents the total number of emergency points. $F = \{f_j \mid j = 1, 2, \cdots, n\}$ denotes the set of facilities, where f_j represents the j-th facility and n represents the number of facilities.

We introduce an additional decision variable Q to represent the maximum distance between all emergency points and their corresponding serving facilities. Given that at most p facilities can be selected from the set of facilities, with each facility' s location relationship to emergency points known and $p \leq n$, $d_{i,j}$ denotes the distance from emergency point e_i to facility f_j. The decision variables are defined as:

$$y_{i,j} = \begin{cases} 1, & \text{if emergency point } e_i \text{ is served by facility } f_j \\ 0, & \text{otherwise} \end{cases}$$

$$x_j = \begin{cases} 1, & \text{if facility } f_j \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

The specific P-center location problem model is as follows:

FF0: Facilities in $F5$ that are assumed not to be opened during case branching in the backtracking algorithm are placed in set $FF0$, which is a global variable.

FF1: Facilities in $F5$ that are assumed to be opened during case branching in the backtracking algorithm are placed in set $FF1$, which is a global variable.

FF5 = F5 / FF1 / FF0: Facilities in $F5$ that are temporarily unprocessed during case branching in the backtracking algorithm are placed in set $FF5$, which is a global variable.

best_q: The best objective function value currently known by the backtracking algorithm, which is a global variable.

Fbest: The set of opened facilities corresponding to the best objective function value currently known by the algorithm, where $|Fbest| \leq p$, which is a global variable.

u_q: The upper bound of the objective function value in a certain state, i.e., the upper bound when facilities in $J1$ and $JJ1$ are opened, which is a local variable.

The objective function Q indicates the need to minimize the maximum distance between each demand point and its service facility.

Constraint (1) ensures that exactly one $y_{i,j}$ is non-zero, meaning exactly one facility can serve an emergency point, guaranteeing that any emergency point e_i can find a serving facility and satisfying the complete coverage requirement.

Constraint (2) ensures that when $x_j = 0$, $y_{i,j} = 0$ must hold, meaning only selected facilities can serve emergency points.

Constraint (3) ensures that at most p facilities can be opened.

Constraint (4) ensures that Q must be greater than the distance between emergency point i and its serving facility. Since this constraint applies to each emergency point, Q must be greater than the distance between any emergency point and its serving facility.

Constraints (5) and (6) specify that variables take values of 0 or 1.

This problem is NP-Hard; unless $P = NP$, no polynomial-time exact algorithm exists [15].

2.1 Mathematical Notation

For ease of description, this paper adopts the following mathematical notation: e_i, f_j represent the i-th emergency point and j-th facility respectively; m, n represent the number of emergency points and facilities respectively; $E = \{e_i \mid i = 1, 2, \cdots, m\}$ denotes the set of all emergency points; $F = \{f_j \mid j = 1, 2, \cdots, n\}$ denotes the set of facilities; p represents the maximum number of facilities to be selected ($p \leq n$); $d_{i,j}$ represents the distance from emergency point e_i to facility f_j; mini_d1(i) represents the minimum distance from emergency point e_i to all facilities; mini_n1(i) represents the facility among all facilities that has the minimum distance to emergency point e_i; mini_d2(i) represents the second-smallest distance from emergency point e_i to all facilities; mini_n2(i) represents the facility among all facilities that has the second-smallest distance to emergency point e_i; maxi_d1(i) represents the maximum distance from emergency point e_i to all facilities; minj_d1(j) represents the minimum distance from facility f_j to all emergency points; maxj_d1(j) represents the maximum distance from facility f_j to all emergency points; min_d1(i, X) represents the minimum distance from emergency point e_i to all facilities in set X; min_n1(i, X) represents the facility in set X that has the minimum distance to emergency point e_i; min_d2(i, X) represents the second-smallest distance from emergency point e_i to all facilities in set X; min_n2(i, X) represents the facility in set X that has the second-smallest distance to emergency point e_i; $F1$ is the set of facilities that must be opened—if any facility in $F1$ is not opened, the optimal solution cannot be obtained—which is a global variable; $F0$ is the set of facilities that must not be opened—if any facility in $F0$ is opened, the optimal solution cannot be obtained—which is a global variable; cur_n is the current cumulative number of opened facilities, which is a local variable; cur_i is the current search level of the backtracking algorithm, which is a local variable; opt_flag indicates whether the algorithm has obtained the optimal solution (opt_flag = 1) or cannot determine whether the current solution is optimal (opt_flag = 0), which is a global variable.

2.2 Mathematical Properties

Property 1 If there exists a facility f_j such that when f_j is opened and the upper bound u_q is less than the current lower bound best_q, then facility f_j must not be opened (where the upper and lower bound calculation algorithms are described in the following two sections).

Proof: When the upper bound u_q is less than the current lower bound best_q, opening facility f_j cannot lead to an optimal solution, so facility f_j should be closed.

Property 2 If there exists a facility f_j such that when f_j is not opened and the upper bound u_q is less than the current lower bound best_q, then facility f_j must be opened.

Proof: When the upper bound u_q is less than the current lower bound best_q,

closing facility f_j cannot lead to an optimal solution, so facility f_j should be opened.

Property 3 When $p = 1$, the facility with the smallest maxj_d1(j) should be opened.

Proof: When $p = 1$, only one facility needs to be selected. Choosing the facility with the smallest maxj_d1(j) minimizes the objective function.

Property 4 If there exist facilities f_j and f_h such that for any emergency point $e_i \in E$, $d_{i,j} \leq d_{i,h}$, then facility f_h must not be opened (i.e., facility f_h is placed in set $F0$).

Proof: Since all emergency points connected to facility f_h can be connected to facility f_j with the same or better objective function value, facility f_h should not be opened.

Property 5 If there exists a facility f_j whose minj_d1(j) is greater than or equal to the maxj_d1(h) of any other facility f_h, then facility f_j must not be opened (i.e., facility f_j is placed in set $F0$).

Proof: Since all emergency points connected to facility f_h can be connected to facility f_j with the same or better objective function value, facility f_j should not be opened.

Property 6 If there exist two emergency points e_i and e_k such that for any facility $f_j \in F$, $d_{i,j} \geq d_{k,j}$, then emergency point e_k can be deleted first, and connecting emergency point e_i to any opened facility at the end of the algorithm does not affect the objective function value.

Proof: Since for any facility $f_j \in F$, $d_{i,j} \geq d_{k,j}$, connecting emergency point e_i to any opened facility does not affect the objective function value.

Property 7 If there exist two emergency points e_i and e_k such that mini_d1(i) $\geq$ maxi_d1(k), then emergency point e_k can be deleted first, and connecting emergency point e_k to any opened facility at the end of the algorithm does not affect the objective function value.

Proof: Since emergency point e_i must be served, the objective function must be greater than or equal to mini_d1(i). The maximum distance from emergency point e_k to all facilities is less than or equal to mini_d1(i), meaning the maximum distance from e_k to all facilities is certainly less than or equal to the objective function value. Therefore, connecting emergency point e_k to any opened facility does not affect the objective function value.

Property 8 If there exists a facility f_j whose maxj_d1(j) is less than or equal to the minj_d1(h) of any other facility f_h, then facility f_j must be opened (placed in set $F1$), and the optimal objective function value is maxj_d1(j).

Proof: In this case, connecting all emergency points to facility f_j yields the optimal solution.

Property 9 If the number of currently unserved emergency points is less than or equal to $p-|F1|-|FF1|$, then for each unserved emergency point e_i, connecting e_i to facility min_n1$(i, F-F0-FF0)$ is optimal.

Proof: This operation results in a total number of opened facilities less than or equal to p, and all currently unserved emergency points are connected to their nearest facilities. This is the optimal way to handle the current unserved emergency points.

Property 10 If each emergency point i is connected to its nearest facility and the total number of opened facilities is less than or equal to p, then the optimal solution is obtained.

Proof: This is obviously true.

Property 11 Sort emergency points by their mini_d1(i) values in descending order, then sequentially connect emergency point e_i to facility mini_n1(i) until either p facilities are opened or all emergency points are served. If there remain unserved emergency points, connect them to the nearest opened facility among the p opened facilities. If all these connection distances are less than or equal to max(mini_d1(i)), then the optimal solution is obtained and opt_flag $= 1$.

Proof: In this case, the objective function value is the lower bound max(mini_d1(i)), so the property holds.

2.3 Lower Bound Sub-algorithm

The lower bound sub-algorithm designed in this paper proceeds as follows:

a) **Initialization:** For any $e_i \in E$ and $f_j \in F$, let each emergency point be served by its nearest facility among all facilities.

b) **Judge the current number of opened facilities.** If the current number of opened facilities is less than or equal to p, the optimal solution can be obtained with objective function value $Q =$ max(mini_d1(i)) $(e_i \in E)$. Output the objective function value and terminate the algorithm. If the number of opened facilities exceeds p, let max(mini_d1(i)) $= b_0$ and proceed to the next step.

c) Let T be the set of all currently opened facilities, with $|T| =$ count. For each facility f_t in T, place all emergency points e_{it} served by f_t into the corresponding set R_t.

d) For each facility f_t in T, perform the following operation: If any emergency point e_i in set R_t satisfies min_d2$(i, T) \leq b_0$, then add facility f_t to set M (meaning if any emergency point served by f_t has a second-nearest distance to facility set T smaller than the current b_0, add f_t to M). If some emergency point e_i in set R_t satisfies min_d2$(i, T) > b_0$, then add facility f_t to set N (meaning if some emergency point served by f_t has a

second-nearest distance to facility set T larger than the current b_0, add f_t to N).

e) If M is not empty, for each facility f_{tm} in M, perform the following operation: Connect all emergency points e_{itm} in set R_{tm} to their corresponding min_n2(itm, T), delete facility f_{tm} from set T, update count, and update the shortest and second-shortest distances from each emergency point to facilities in set T. If count $\leq p$, the optimal solution is obtained with opt_flag $= 1$ and the entire algorithm terminates. If count $> p$, proceed to step f). If M is empty, proceed to step f).

f) **Update the initial lower bound.** For each set R_t of facility f_t in set N, compute max(mini_d2(i)). Let b_t be the minimum among these maximum values. If $b_t > b_0$, update the lower bound to b_t; if $b_t \leq b_0$, keep the lower bound as $b_0 =$ max(mini_d1(i)) and terminate the algorithm.

2.4 Upper Bound Sub-algorithm

In fact, the objective function value corresponding to any feasible solution can serve as an upper bound for the problem. This paper first utilizes the mathematical properties described above to reduce the problem' s order, then executes the following upper bound sub-algorithm to obtain the upper bound:

a) For all $e_i \in E$, sort them in descending order of mini_d2(i). After sorting, sequentially connect emergency point e_i to its corresponding facility mini_n1(i), opening facilities one by one until either p facilities are opened or all emergency points are served. Place opened facilities into set U and unopened facilities into set V.

b) If all emergency points are served by facilities in set U, the optimal solution is obtained with opt_flag $= 1$ and the entire algorithm terminates; otherwise, proceed to the next step.

c) Connect all unserved emergency points e_{iu} to their corresponding facilities min_n1(iu, U) in set U.

d) For all $e_i \in E$, use max(min_d1$(i, U)) = u_0$ as the upper bound, and let f_h be the facility corresponding to max(min_d1(iu, U)).

e) If there exists a facility f_j in set V such that replacing the opened facility f_h with f_j (i.e., executing $U = U \cup \{f_j\} - \{f_h\}$, $V = V \cup \{f_h\} - \{f_j\}$) results in max(min_d1$(i, U)) < u_0$ for all $e_i \in E$, then decide to replace f_h with f_j and jump to step d) for execution. If no such facility f_j exists, the sub-algorithm terminates.

3 Reduction Backtracking Algorithm

This algorithm consists of two parts: a reduction algorithm and a backtracking algorithm. The reduction algorithm reduces the problem' s order by combining

mathematical properties to shrink the problem scale. The backtracking algorithm uses depth-first search to explore the solution space. The backtracking algorithm processes each facility in set $FF5$ in two cases: a) if facility f_j is opened, $FF1 = FF1 \cup \{f_j\}$, $FF5 = FF5/\{f_j\}$, and search the corresponding left subtree; b) if facility f_j is not opened, $FF0 = FF0 \cup \{f_j\}$, $FF5 = FF5/\{f_j\}$, and search the corresponding right subtree.

3.1 Reduction Sub-algorithm

The reduction sub-algorithm proceeds as follows:

a) Initialize opt_flag = 0, cur_i = 1, cur_n = 0, best_q = $+\infty$, $u_q = 0$.

b) Combine the mathematical properties from previous sections: if a facility f_j must be opened, $F1 = F1 \cup \{f_j\}$, $F5 = F5/\{f_j\}$; if a facility f_j must not be opened, $F0 = F0 \cup \{f_j\}$, $F5 = F5/\{f_j\}$. If set $F5$ changes after these operations, repeat step a).

c) Calculate the upper bound for the problem using the upper bound sub-algorithm from previous sections, assign the current opening configuration to set *Fbest*, and assign the corresponding objective function value to best_q.

d) Call the backtracking sub-algorithm described in the next section.

3.2 Backtracking Sub-algorithm

This backtracking algorithm uses depth-first search to explore the problem' s solution space, traversing all solution spaces from the root node. When searching any node, it checks whether the theoretical upper bound best_q is greater than the lower bound b_q. If satisfied, it continues the depth-first search downward; otherwise, it performs pruning, skipping that node and its subtrees, and backtracks upward level by level.

Before executing the algorithm, call the reduction sub-algorithm described previously. If opt_flag = 1, the optimal solution has been obtained and the algorithm terminates; otherwise, execute the backtracking sub-algorithm.

For ease of description, renumber the facilities in $F5$, initialize $FF5 = F5$, $FF1 = \{\}$, $FF0 = \{\}$, then call the recursive backtracking subroutine backtrack(1).

The backtracking subroutine backtrack(cur_i) is described as follows:

a) If the current search level cur_i $> |F5|$ or $|FF5| = 0$, the search has reached a leaf node. If $|F1| + |FF1| \leq p$, a feasible solution is obtained. If the corresponding objective function value $Q <$ best_q, update the optimal solution: best_q = Q, $Fbest = F1 \cup FF1$. Terminate the subroutine.

b) Check whether the opened facilities $F1 \cup FF1$ have covered all emergency points and whether $|F1 \cup FF1| \leq p$. If satisfied, a leaf node is reached

and a feasible solution is obtained. If the corresponding objective function value $Q <$ best_q, update the optimal solution: opt_flag = 1, best_q = Q, $Fbest = F1 \cup FF1$. Terminate the subroutine.

c) Calculate the lower bound b_q assuming facility $f_{\text{cur\_i}}$ is opened. If $|FF1| < p - |F1|$ and $|FF5| \geq p - |F1| - |FF1|$, and $b_q <$ best_q, then open the current facility: execute $FF1 = FF1 \cup \{f_{\text{cur\_i}}\}$, $FF5 = FF5/\{f_{\text{cur\_i}}\}$, and call backtrack(cur_i+1) to search the left subtree. If these conditions are not satisfied, jump to step f) (pruning the left subtree).

d) If opt_flag = 1, the algorithm has obtained the optimal solution; terminate the subroutine. Otherwise, continue executing the subroutine.

e) Before jumping to the upper level of the search tree, restore $FF1$ and $FF5$ to their initial states in step c): $FF1 = FF1/\{f_{\text{cur\_i}}\}$, $FF5 = FF5 + \{f_{\text{cur\_i}}\}$.

f) Calculate the lower bound b_q assuming facility $f_{\text{cur\_i}}$ is not opened. If $|FF1| < p - |F1|$ and $|FF5| \geq p - |F1| - |FF1|$, and $b_q <$ best_q, then do not open the current facility: execute $FF0 = FF0 \cup \{f_{\text{cur\_i}}\}$, $FF5 = FF5/\{f_{\text{cur\_i}}\}$, and call backtrack(cur_i + 1) to search the right subtree. If these conditions are not satisfied, terminate the subroutine (pruning the right subtree).

g) Before returning to the upper level of the search tree, restore $FF0$ and $FF5$ to their initial states in step f): $FF0 = FF0/\{f_{\text{cur\_i}}\}$, $FF5 = FF5 + \{f_{\text{cur\_i}}\}$.

After the algorithm terminates, the current optimal objective function value best_q and the corresponding opened facilities $Fbest$ constitute an optimal solution to the entire problem.

3.3 Algorithm Time Complexity Analysis

The problem scale studied in this paper is the number of facilities n. The algorithm generates at most 2^n leaf nodes in the search space tree. After the reduction sub-algorithm is called, the problem scale is reduced to $|F5|$. Let $k = |F5|$, so the worst-case time complexity of the algorithm is $O(2^k)$.

Many scholars have proposed different algorithms for emergency location problems, which are mainly divided into exact algorithms, approximation algorithms, and heuristic algorithms. Approximation and heuristic algorithms have the advantage of fast solution speed but generally cannot obtain optimal solutions. Compared with general exact algorithms, the exact algorithm proposed in this paper reduces the problem scale and solution space by studying its mathematical properties, performs reasonable pruning of the search tree before calling the backtracking algorithm, and only searches a subset of the solution tree, thereby accelerating the solution speed.

4.1 Example Analysis

To more clearly illustrate the algorithm in this paper, we present an example to explain the entire execution process of the algorithm.

Example 1 As shown in [Figure 1: see original paper], there are 6 facilities j and 5 emergency points i, from which at most $p = 3$ facilities are to be selected to minimize the maximum demand distance. gives the distances between each emergency point and facility.

The analysis process for Example 1 can be described as follows:

a) According to Property 4, facility $j5$ is strictly dominated by facility $j4$, so delete facility $j5$ from the original bipartite graph and delete edges $\{i1j5, i2j5, i3j5, i4j5, i5j5, i6j5\}$.

b) According to Property 4 or Property 5, facility $j6$ can be deleted, so delete facility $j6$ from the original bipartite graph and delete edges $\{i1j6, i2j6, i3j6, i4j6, i5j6, i6j6\}$.

c) According to Property 6, emergency point $i2$ can be disregarded relative to $i1$, so delete emergency point $i2$ and delete edges $\{i2j1, i2j2, i2j3, i2j4, i2j5, i2j6\}$ from the original bipartite graph.

d) According to Property 7, emergency point $i4$ can be disregarded relative to $i3$ (since facility $j6$ has already been deleted), so delete emergency point $i4$ and delete edges $\{i4j1, i4j2, i4j3, i4j4, i4j5, i4j6\}$ from the original bipartite graph.

e) After reduction processing, update the problem scale as shown in [Figure 2: see original paper].

f) At this point, connect the remaining emergency points to the nearest facilities in the remaining facility set. The current number of opened facilities is calculated as $4 > p$, and the lower bound $b = 5$ is computed.

g) Execute the upper bound sub-algorithm to calculate the upper bound $u = 6$.

h) Execute the backtracking algorithm, whose execution process is shown as the binary tree in [Figure 3: see original paper].

The reduced solution space binary tree is shown in the figure. Through the above algorithm, the optimal solution set for this problem is $Fbest = \{j2, j3, j4\}$ with optimal objective function value best_q = 5, and the algorithm terminates.

4.2 Experimental Results Comparison and Analysis

The experiments in this paper were conducted on a CPU Core i7-7500 @2.70GHz with a 64-bit Windows 10 operating system. Using benchmark data provided in the Benchmark Library, we selected instances from The P-median Problem with $m = n = 128$ and $p = 16$, where service distances range from (0, 4).

We implemented the exact algorithm (BA) and ant colony algorithm (ACO) in Python to solve the same data and compared the results of the two algorithms. The experimental results record the solution outcomes and program running times, as shown in .

From Example 1, it is clear that using mathematical properties can reduce problem scale and accelerate algorithm execution. From the experimental results, the conclusions are consistent with theoretical analysis: the ant colony algorithm is a heuristic algorithm with faster computation speed than exact algorithms, while the algorithm designed in this paper is an exact algorithm that can obtain optimal solutions every time. Therefore, in the above experiments, the results obtained by our algorithm demonstrate better stability than the ant colony algorithm, which is a heuristic algorithm.

Domestic and international research on algorithms for location problems generally falls into heuristic algorithms, approximation algorithms, and exact algorithms. The reduction backtracking algorithm proposed in this paper is an exact algorithm that utilizes mathematical properties, upper bounds, and lower bounds to accelerate algorithm execution. Compared with heuristic and approximation algorithms, the exact algorithm designed in this paper guarantees obtaining correct and stable global optimal solutions every time, whereas heuristic algorithms tend to fall into local optima and approximation algorithms can only approach the optimal solution within a certain similarity ratio and generally cannot obtain the optimal solution. Research on exact algorithms for this problem is relatively limited. Compared with general exact algorithms that analyze all nodes and have high time complexity, this paper utilizes the mathematical properties of the problem combined with the designed upper and lower bound sub-algorithms to batch-determine certain facilities to be opened or not opened, reducing the problem scale and requiring analysis of only partial nodes. This alleviates the difficulty of exact algorithms in handling large-scale problems to some extent, making the algorithm proposed in this paper more efficient than general exact algorithms.

5 Conclusion

This paper studies the P-center model in emergency facility location problems. Addressing the disadvantages that heuristic algorithms cannot obtain optimal solutions and exact algorithms have high time complexity for large-scale problems, we design a reduction backtracking exact algorithm that can quickly reduce problem scale while obtaining exact solutions. The algorithm aims to ensure global optimal solutions while minimizing solution time as much as possible.

The paper investigates the mathematical properties of the P-center location problem and provides corresponding proofs. These properties, combined with the designed upper and lower bound sub-algorithms, can batch-determine certain facilities to be opened or not opened during the search process, enabling extensive pruning during backtracking. This reduces the problem-solving scale

and improves algorithm execution speed. Additionally, these mathematical properties can be applied to other algorithms. Through theoretical analysis and Example 1, it can be seen that when solving the P-center location problem using the algorithm in this paper, not only can exact solutions be obtained, but the mathematical properties and upper/lower bound sub-algorithms can also reduce problem scale and accelerate solution speed.

References

[1] Hakimi S L. Optimum locations of switching centers and the absolute centers and medians of a graph [J]. Operations Research, 1964, 12 (3): 1202-1207.

[2] Hochbaum D S, Shmoys D B. A best possible heuristic for the k-center problem [J]. Mathematics of Operations Research, 1985, 10 (2): 180-184.

[3] Dyer M E, Frieze A M. A simple heuristic for the p-center problem [J]. Operations Research Letters, 1985, 3 (6): 285-288.

[4] Wen M, Kang R. Some optimal models for facility location-allocation problem with random fuzzy demands [J]. Applied Soft Computing, 2011, 11 (1): 1202-1207.

[5] Bozkaya B, Tansel B. A spanning tree approach to the absolute p-center problem [J]. Location Science, 1998, 6 (1): 83-107.

[6] Zhu Yan, Shao Quan, Jia Meng. Research on the layout of general aviation rescue point [J]. Henan Science, 2015 (2): 265-270.

[7] Davidović T, Ramljak D, Elmi M, et al. Bee colony optimization for the p-center problem [J]. Computers & Operations Research, 2011, 38 (10): 1367-1376.

[8] Xu Yanchen, Dai Tao. Solving multiple distribution center location allocation problem using K-Means algorithm and center of gravity method [J]. Logistics Technology, 2019, 38 (06): 69-73.

[9] Huang Shuqiang, Jiang Xiumei, Fan Rensheng. Method for weighted distance continuous K-center facility location problem [J]. Journal of Chinese Computer Systems, 2020, 41 (02): 310-315.

[10] Sandoval MG, Diaz JA, Rios-Mercado RZ. An improved exact algorithm for a territory design problem with p-center-based dispersion minimization [J]. Expert Systems with Applications, 2020, 146 (1): 113-124.

[11] Contardo C, Iori M, Kramer R. A scalable exact algorithm for the vertex p-center problem [J]. Computers & Operations Research, 2019, 103 (MAR): 211-220.

[12] Zhou Jianjie. The problem of road location on graph and the problem of connecting p-center and p-median [D]. Shanghai: Shanghai University, 2016.

[13] Perez-Pelo S, Sanchez-Oro J, Lopez-Sanchez AD. A multi-objective parallel iterated greedy for solving the p-center and p-dispersion problem [J]. Electronics, 2019, 8 (12): 1140.

[14] Oezsoy F A, Pinar M C. An exact algorithm for the capacitated vertex p-center problem [J]. Computers & Operations Research, 2006, 33 (5): 1420-1436.

[15] Michael R Garey, David S Johnson. Computers and intractability: A guide to the theory of NP-completeness [J]. Bulletin (New Series) of the American Mathematical Society, 1980, 3 (2): 898-904.

Note: Figure translations are in progress. See original paper for figures.