A Weighting-Based Tabu Search Algorithm for the p-Next Center Problem

Qingyun Zhang, **Zhouxing Su***, **Zhipeng Lü*** and **Lingxiao Yang**
Huazhong University of Science and Technology, China
{qingyun_zhang, suzhouxing, zhipeng.lv, yanglingxiao}@hust.edu.cn

Abstract

The p-next center problem (pNCP) is an extension of the classical p-center problem. It consists of locating p centers from a set of candidate centers and allocating both a reference and a backup center to each client, to minimize the maximum cost, which is the length of the path from a client to its reference center and then to its backup center. Among them, the reference center is the closest center to a client and serves it under normal circumstances, while the backup center is the closest center to the reference center and serves the client when the reference center is out of service. In this paper, we propose a weighting-based tabu search algorithm called WTS for solving pNCP. WTS optimizes the pNCP by solving its decision subproblems with given assignment costs with an efficient swap-based neighborhood structure and a hierarchical penalty strategy for neighborhood evaluation. Extensive experimental studies on 413 benchmark instances demonstrate that WTS outperforms the state-of-the-art methods in the literature. Specifically, WTS improves 12 previous best known results and matches the optimal results for all remaining 401 ones in a much shorter time than other algorithms. More importantly, WTS reaches the lower bounds for 10 instances for the first time.

1 Introduction

The p-center problem is a classical discrete optimization problem and has been proven to be NP-hard [Kariv and Hakimi, 1979]. It consists of selecting p centers from a set of candidate centers and serving a set of clients in order to minimize the maximum distance between each client and its closest center. The p-center problem has a wide range of applications, such as the problems of determining the locations of emergency centers [Toregas *et al.*, 1971], hospitals [Hakimi, 1964], and fire stations [Drezner, 1987] to serve the communities. In the last 50 years, the p-center problem has been widely concerned in academic society. There are mainly three kinds of methodologies, including exact algorithms (Minieka [1970], Elloumi *et al.* [2004], Calik and Tansel [2013], Liu *et al.* [2020]), approximation algorithms (Hochbaum and Shmoys [1985], Martinich [1988]), and metaheuristic algorithms (Mladenović *et al.* [2003], Pullan [2008], Irawan *et al.* [2016], Yin *et al.* [2017], Zhang *et al.* [2020]), respectively.

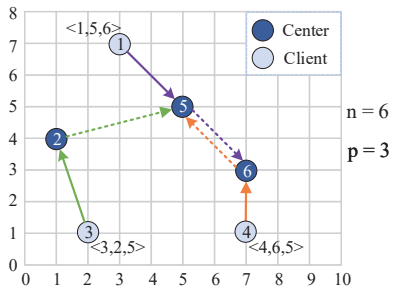


Figure 1: There are 6 nodes in the graph, $< i, j, k >$ indicates that the reference and backup of vertex i are j and k, respectively.

In real-world applications, a center may be forced to be closed due to an unpredictable incident. In that case, the clients served by the center have to turn to another center from the current center as quickly as possible. Therefore, Albareda-Sambola *et al.* [2015] introduced the p-next center problem (pNCP), which occurs in many industrial applications and has been proven to be NP-hard. It requires locating p centers from a set of candidate centers and assigning a reference center and a backup center to each client. The objective of pNCP is to minimize the maximum cost of all clients, which is calculated by the distance of a client to its closest center (reference center), plus the distance from the reference center to its closest center (backup center). Figure 1 shows an illustrative diagram of pNCP.

Albareda-Sambola *et al.* [2015] also proposed a series of different formulations for pNCP and compared their performances. López-Sánchez *et al.* [2019] proposed a hybrid metaheuristic algorithm, which incorporates Greedy Randomized Adaptive Search Procedure (GRASP) and Variable Neighborhood Search algorithm (VNS) to solve the pNCP. Londe *et al.* [2021] introduced a Biased Random-Key Genetic Algorithm (BRKGA), which combines a multi-parent (MP) strategy and a path-relinking (PR) procedure. They conducted experiments on 413 instances derived from p-median [Mladenović *et al.*, 2007] in the OR-Library [Beasley, 1990].

*Co-corresponding authors.

In this paper, we present a weighting-based tabu search (WTS) algorithm for solving the pNCP. WTS uses the weighting technique and a solution-based tabu search framework to strengthen the search capability. Instead of directly optimizing the original problem like previous metaheuristics, WTS converts the original pNCP into a series of decision subproblems and solves them with a fast tabu search procedure which adopts a neighborhood structure based on the closing-opening swap and an incremental neighborhood evaluation technique based on a hierarchical penalty strategy.

Our main contributions can be summarized as follows:

1) We propose a new solution approach which transforms the pNCP from an optimization problem to a series of decision subproblems and solves each of them in turn to obtain the solution of the original optimization problem. It improves the smoothness of the solution space, which makes it easier for the local search-based algorithms to solve the pNCP.
2) We present a weighting technique for solving the pNCP, which diversifies the search trajectory and helps the search to escape from the local optima trap.
3) WTS algorithm employs a new neighborhood structure based on a dedicated swap move which is made up of a closing operation and an opening operation. In addition, we use a new hierarchical penalty strategy for neighborhood evaluation and an incremental evaluation strategy to improve the efficiency of search.
4) Tested on 413 classical instances of the pNCP, the proposed WTS algorithm improves the best known results on 12 instances and obtains the optimal solutions on 10 instances for the first time, and matches the records on the remaining ones. In addition, the computational time of WTS is much shorter than that of the state-of-the-art algorithms in the literature.
5) We conduct additional experiments to analyze the importance of the key components of our algorithm, such as the solution-based tabu strategy, the weighting technique, and the hierarchical penalty strategy.

2 Problem Description and Transformation

Given an undirected complete graph $G = (V, E)$, where $V = \{1, 2, ..., n\}$ is the vertex set and $E = \{(i, j)|i, j \in V\}$ is the edge set. The distance between each pair of vertices i and j is d_{ij}. Each vertex $i \in V$ corresponds to a client to be served by a reference and a backup center, where the two centers are both one of the p centers, which should be selected from a set of candidate centers $C \subseteq V$ (usually $C = V$). The goal of pNCP is to choose p best centers from C such that the maximum length of all serving paths can be minimized. We define d_{ijk} as the total distance from a vertex i to a center j and then to another center k, where $d_{ijk} = d_{ij} + d_{jk}, i \in V, j, k \in C$. The solution can be defined as $(\boldsymbol{x}, \boldsymbol{y}, z)$. In detail, $\boldsymbol{x} = \{x_j|j \in C\}$ and x_j is a decision variable which equals to 1 if and only if a candidate center $j \in C$ is opened as a center. $\boldsymbol{y} = \{y_{ijk}|i \in V, j, k \in C\}$ and y_{ijk} is a binary variable, where y_{ijk} = 1 iff the reference center and backup center of client vertex $i \in V$ are $j, k \in C$, respectively. $z \in R^+$ is the upper bound of the assignment cost, i.e., the length of the longest path. Based on the above notations, the classical mixed-integer programming (MIP) model [Albareda-Sambola *et al.*, 2015] for the pNCP can be formulated as follows.

$$\min\ z, \tag{1}$$

$$\text{s.t.} \sum_{j \in C} x_j \leq p, \tag{2}$$

$$\sum_{j \in C} \sum_{\substack{k \in C, k \neq i,j, \\ d_{ik} \geq d_{ij}}} y_{ijk} = 1, \forall i \in V, \tag{3}$$

$$\sum_{\substack{k \in C, k \neq i,j, \\ d_{ik} \geq d_{ij}}} y_{ijk} \leq x_j, \forall i \in V, \forall j \in C, \tag{4}$$

$$\sum_{j \in C} y_{ijk} \leq x_k, \forall i \in V, \forall k \in C, k \neq i, j, d_{ik} \geq d_{ij}, \tag{5}$$

$$x_j + \sum_{\substack{h \in C, \\ d_{ih} > d_{ij}}} \sum_{\substack{k \in C, k \neq i,j, \\ d_{ik} \geq d_{ij}}} y_{ihk} \leq 1, \forall i \in V, \forall j \in C, \tag{6}$$

$$\sum_{j \in C} \sum_{\substack{k \in C, k \neq i,j, \\ d_{ik} \geq d_{ij}}} d_{ijk} y_{ijk} \leq z, \forall i \in V, \tag{7}$$

$$x_j \in \{0, 1\}, z \in R^+, \forall j \in C, \tag{8}$$

$$y_{ijk} \in \{0, 1\}, \forall i \in V, \forall j, k \in C, k \neq i, j, d_{ik} \geq d_{ij}. \tag{9}$$

Objective (1) aims to minimize the assignment cost. Constraint (2) limits the number of opened centers to at most p. Constraints (3) ensure that each vertex must be assigned to one reference and one backup center. Constraints (4) and (5) restrict that only the opened centers can be assigned to the vertices. Constraints (6) force that the reference center of each vertex must be its closest center. Constraints (7) ensure that the cost z is not less than the length of any path from each vertex to its reference center and then to its backup center.

We can transform pNCP to a series of decision subproblems called z-cost pNCP, where the maximum length of serving paths is fixed as z, and we need to locate p centers under the constraint that the actual cost is no more than z. Once the z-cost pNCP is solved, we can find a feasible solution for pNCP with cost equal to z. Obviously, the value of z must be one of d_{ijk}. Let $Z = \{z_1, z_2, \ldots, z_m\}$ be the set containing all distinct values of d_{ijk}, where $z_1 < z_2 < z_m$. Then, we can optimize the pNCP by decreasing z from z_m to z_1 and solving each corresponding z-cost pNCP.

3 Vertex Weighting-Based Tabu Search

3.1 General Framework

The main framework of the WTS algorithm is described in Algorithm 1. First, we employ a constructive heuristic to obtain an initial feasible solution and the upper bound of the cost. Then, WTS converts pNCP into a series of z-cost pNCP decision subproblems and solves them by a solution-based tabu search procedure combined with a weighting technique.

Algorithm 1 The main framework of the WTS algorithm

Input: A graph G, the center number p
Output: The best solution found so far X^*

1: Solution tabu list $ST \leftarrow \varnothing$
2: $iter \leftarrow 1$, vertex weights $w_i \leftarrow 1, \forall i \in V$
3: Age $a_j \leftarrow 0, \forall j \in C$
4: Current solution X, initial cost $z_q \leftarrow$ Initialize(G, p)
5: Previous solution $X' \leftarrow X$, $X^* \leftarrow X$
6: The ordered list of z-costs $Z = \{z_1, z_2, ..., z_q\}$
7: **while** Termination condition is not met **do**
8: $q \leftarrow q - 1$, update set of illegal vertices $I(X)$
9: **while** Termination condition is not met **do**
10: $(i, j) \leftarrow$ FindPair$(X', ST, iter)$ // Algorithm 4
11: $X \leftarrow X \cup \{i\} \setminus \{j\}$
12: **if** $|I(X)| = 0$ **then**
13: $X^* \leftarrow X$
14: **break**
15: **else if** $|I(X)| > |I(X')|$ **and** $|I(X)| \leq 5$ **then**
16: $w_k \leftarrow w_k + 1, \forall k \in I(X)$ // Section 3.2
17: **end if**
18: $ST \leftarrow ST \cup \{X\}, X' \leftarrow X, iter \leftarrow iter + 1$
19: **end while**
20: **end while**
21: **return** X^*

Algorithm 2 Closing a center

1: **function** CLOSINGCENTER(j)
2: **for all** $l \in X$ **do**
3: **if** $FY_l^1 = j$ **then**
4: $FY_l^1 \leftarrow FY_l^2, DY_l^1 \leftarrow DY_l^2$
5: $(FY_l^2, DY_l^2) \leftarrow$ FindNextFY(j, l)
6: **else if** $FY_l^2 = j$ **then**
7: $(FY_l^2, DY_l^2) \leftarrow$ FindNextFY(j, l)
8: **end if**
9: **end for**
10: **for all** $v \in S_j$ **do**
11: **if** $FN_v^1 = j$ **then**
12: $FN_v^1 \leftarrow FN_v^2, DN_v^1 \leftarrow DN_v^2$
13: $(FN_v^2, DN_v^2) \leftarrow$ FindNextFN(j, v)
14: **else if** $FN_v^2 = j$ **then**
15: $(FN_v^2, DN_v^2) \leftarrow$ FindNextFN(j, v)
16: **end if**
17: $t \leftarrow FN_v^1$
18: $D_v \leftarrow DN_v^1 + DY_t^1$
19: update $I(X)$ and δ_j
20: **end for**
21: **end function**

As soon as the z_q-cost pNCP is solved, we decrease z_q by setting z_q to z_{q-1} and solve the resulting subproblem again.

Specifically, we first generate an initial solution X with an initial cost z_q by a greedy construction algorithm (line 4). The constructive heuristic first picks a center randomly, and then it opens centers one by one under the principle of minimizing objective (1) until $|X| = p$. According to the initial cost z_q, we initialize the ordered cost list Z with all distinct path lengths (line 6), where $z_1 < z_2 < ... < z_q$. Then, we solve the z_q-cost subproblems in turn by decreasing q to $q - 1$. When solving a subproblem, we relax constraints (7) where the assignment cost of each vertex is no more than z_q and record every vertex that violates the constraint as an illegal vertex. For each subproblem, we first record $I(X)$ by the illegal vertices (line 8), and then optimize it iteratively by tabu search until $|I(X)| = 0$ or the time limit is met (lines 9-20). At each iteration of WTS, it evaluates the neighborhood of the current solution X and selects the best neighborhood move in non-tabu status (line 10, Algorithm 4). If the number of illegal vertices in the current solution X is 0, i.e., $|I(X)| = 0$, it means that a legal solution for the current decision subproblem is found, and the best solution X^* will be updated with X (lines 12-14). Otherwise, when the current solution is trapped into a local optimum, i.e., the best move returned by FindPair() cannot reduce the number of illegal vertices, WTS will increase the weight of each illegal vertex to encourage eliminating their constraint violations in the next iterations (lines 15–16). At the end of each iteration, the associated data structures will be updated (line 18). Finally, once the specified termination condition (e.g., time limit or maximum number of iterations) is met, WTS terminates and returns the best solution X^* (line 21).

3.2 Reformulation and Weighting Technique

The weighting technique helps the search to escape from the local optima by introducing an adaptive objective function. It has been successfully applied to many problems, such as unicost set covering problem [Gao *et al.*, 2015], classical p-center problem [Zhang *et al.*, 2020], and optimal camera placement problem [Su *et al.*, 2021]. We transform pNCP into a series of decision subproblems, which determine whether all vertices can assign a reference center and a backup center within specific z-cost. In order to effectively solve each subproblem, we relax the constraints that each vertex must be allocated within z-cost and impose a penalty for each illegal vertex on the objective function. Specifically, let w_i be the weight of vertex $i \in I(X) \subset V$, the objective function $f(X)$ can be defined as Eq. (10).

$$\min f(X) = \sum_{i \in I(X) \subset V} w_i \tag{10}$$

The initial weight of each vertex is 1. In the subsequent search, if WTS fails to reduce the number of illegal vertices, i.e., the tabu search is trapped in the local optima and the objective value is below the threshold, WTS will increase the weight w_i of each illegal vertex $i \in I(X)$ by one unit (Algorithm 1, lines 15–16). Specifically, at the early stage of the neighborhood search, tabu strategy is sufficient for jumping out of the local optima trap, so the threshold is set to 5 in WTS. In the later stage of the search, once stagnation is encountered, the weight of the corresponding illegal vertices will be increased, and the more times a vertex appears in $I(X)$, the greater its weight will be. The weighting strategy changes the landscape of the solution space, and is thus able to effectively prevent vertices from being illegal for a long time.

Algorithm 3 Opening a center

```
1: function OPEINGCENTER(i)
2:     for all l ∈ X do
3:         if DY_l^1 > d_li then
4:             FY_l^2 ← FY_l^1, DY_l^2 ← DY_l^1
5:             FY_l^1 ← i, DY_l^1 ← d_li
6:         else if DY_l^2 > d_li then
7:             FY_l^2 ← i, DY_l^2 ← d_li
8:         end if
9:     end for
10:    for all v ∈ S_i do
11:        if DN_v^1 > d_li then
12:            FN_v^2 ← FN_v^1, DN_v^2 ← DN_v^1
13:            FN_v^1 ← i, DN_v^1 ← d_vi
14:        else if DN_v^2 > d_li then
15:            FN_v^2 ← i, DN_v^2 ← d_vi
16:        end if
17:        t ← FN_v^1
18:        D_v ← DN_v^1 + DY_t^1
19:        update I(X) and δ_i
20:    end for
21: end function
```

Algorithm 4 Find the best swap pair

```
1: function FINDPAIR(X, ST, iter)
2:     Best swap move (i*, j*)← ∅, δ ← +∞
3:     for all j ∈ X do // evaluate closing center j
4:         ClosingCenter(j) // for evaluation only
5:         if (δ_j < δ) ∨ ((δ_j = δ) ∧ (a_j > a_j^*)) then
6:             δ ← δ_j, j* ← j
7:         end if
8:     end for
9:     δ ← +∞
10:    for all i ∈ C \ X do // evaluate opening center i
11:        if X ⊕ m(i, j*) ∉ ST then
12:            OpeningCenter(i) // for evaluation only
13:            if δ_i < δ then
14:                δ ← δ_i, i* ← i
15:            end if
16:        end if
17:    end for
18:    return (i*, j*)
19: end function
```

3.3 Neighborhood Structure and Evaluation

In order to solve each z-cost pNCP subproblem, WTS adopts a swap-based neighborhood structure. Specifically, a swap move consists of closing a center $j \in X$ and opening a center $i \in C \setminus X$, denoted as $m(i, j)$. Based on the incumbent solution X, performing a move produces a new neighborhood solution $X \oplus m(i, j)$. At each neighborhood evaluation, we perform the best closing operation out of $O(p)$ ones and the best opening operation out of $O(n - p)$ ones in turn. So the size of the complete neighborhood is $O(p+(n-p)) = O(n)$.

In local search procedure, the neighborhood evaluation is an extremely important ingredient. To obtain the best neighboring solution and improve the incumbent solution X, the proposed WTS algorithm uses the best-improvement policy, i.e., for a swap move based on closing center and opening center operations, it evaluates all neighborhood moves and performs the closing and opening operations that lead to the best neighboring solution. To evaluate a neighborhood move of the current solution, we need to allocate a reference and a backup center for each client to obtain its assignment cost for updating $I(X)$, which means that the time complexity of the naive neighborhood evaluation is $O(2np)$. However, it is quite time-consuming for large instances. In order to improve efficiency without dramatically sacrificing the solution quality, WTS follows a hierarchical penalty strategy for neighborhood evaluation and an incremental evaluation technique.

Notation	Description
FN_i^k	The k-th nearest center to $i \in V$
FY_j^k	The k-th nearest center to $j \in X$, excluding j
DN_i^k	The distance from $i \in V$ to FN_i^k
DY_j^k	The distance from $j \in X$ to FY_j^k

Table 1: Notation description.

On the one hand, in order to accelerate the neighborhood evaluation procedure, WTS uses a set of dedicated data structures to store the k-nearest centers to each vertex and each center, and the corresponding distances, as shown in Table 1, where k is set to 1 and 2. With this set of data structures, the status of vertices can be quickly calculated each time closing or opening a center. Specifically, the assignment cost of each vertex $i \in V$ is $D_i = DN_i^1 + DY_j^1$ ($j = FN_i^1$), and the assignment of vertex i is legal iff $D_i \leq z_q$. In addition, WTS adopts an incremental evaluation mechanism to efficiently evaluate all neighborhood moves instead of naively calculating the objective by Eq. (10). Specifically, there are only small changes in the neighboring solutions, thereby calculating their objective by naively evaluating function f is unnecessary. The objective of a neighboring solution can be calculated incrementally by the variation of closing a center i and opening a center j, i.e., $f(X \oplus m(i, j)) = f(X) + (\delta_i + \delta_j)$, where δ represents the variation of the objective value after closing a center or opening a center. Algorithms 2 and 3 update the data structure and calculate the improvement δ (line 19) after closing center j and opening center i, respectively, where S will be introduced in Section 3.4. The worst-case complexities of ClosingCenter() and OpeningCenter() are $O(np)$ and $O(n)$, respectively.

On the other hand, WTS uses a hierarchical penalty strategy to reformulate the neighborhood evaluation function. We divide the illegality of vertices into two parts, and use different penalty-factors for them. Specifically, when the distance from the illegal vertex $i \in I(X)$ to its reference center is not greater than z_q, i.e., $DN_i^1 \leq z_q$, the penalty factor is 1. Otherwise, the penalty factor is 2. Combining the hierarchical penalty strategy described in this section and Eq. (10) in Section 3.2, the new evaluation function becomes Eq. (11).

$$\min f(X) = 2 \sum_{\substack{i \in I(X) \\ DN_i^1 > z_q}} w_i + \sum_{\substack{i \in I(X) \\ DN_i^1 \leq z_q}} w_i \qquad (11)$$

In the neighborhood evaluation process, when there exist multiple best neighboring solutions according to the reformulated evaluation function Eq. (11), we employ an age-based strategy to break ties. Specifically, the age of a center denotes the number of iterations for which it remains opening. The larger the age of a center is, the more iterations the center has been opened for. Thus, we break ties by favoring the neighboring solution which closes an older center. In addition, to avoid re-closing the just-opened center, the center which is just opened will not be evaluated at the next iteration.

Algorithm 4 shows the neighborhood evaluation procedure. In the worst-case, the time complexity is $O(n^2)$.

3.4 Neighborhood Reduction

Obviously, if the assignment cost of vertex i satisfies $D_i \leq z_q$, then $d_{ij} \leq z_q$ and $d_{jk} \leq z_q$, where j and k are the reference and backup of vertex i, respectively, i.e., if the total cost of a vertex i is not more than z_q, the distance between i and its reference center, and the distance between the reference center and its backup center are not more than z_q. So we adopt a neighborhood reduction rule to accelerate the neighborhood search procedure. Specifically, we ignore the centers which cannot reduce the cost of vertex i, and save the promising centers with the set $S_i = \{j | d_{ij} \leq z_q, j \in C\}$.

3.5 Solution-Based Tabu Search Strategy

Tabu search is an optimization algorithm based on local search framework [Glover, 1989]. In recent years, the solution-based tabu search strategy has been widely used to solve many combinatorial problems, such as minimum differential dispersion problem [Wang *et al.*, 2017], 0-1 multidimensional knapsack problem [Lai *et al.*, 2018], and inventory routing problem [Su *et al.*, 2020]. WTS algorithm adopts a solution-based tabu search strategy. Different from the classic attribute-based tabu strategy, the solution-based tabu strategy records complete information of each visited solution, and these solutions will never be evaluated again. This policy avoids the drawbacks of the attribute-based tabu strategy, which may falsely accept already visited solutions or forbid some promising moves.

At each iteration, making a move will produce a new current solution X to be stored into the tabu list ST, which is a Boolean vector of length $L = 10^8$. Specifically, WTS defines a hash function h to map the solution X to an integer in the range $[0, L)$, denoted as $h(X)$, where $ST(h(X)) = 1$ means that the solution with the hash value $h(X)$ is in tabu status. As there may be hash collisions that lead to incorrect identification of the tabu status of some unvisited solutions, we use multiple hash functions h^t $(t = 1, 2, 3)$ and corresponding $ST^t(h^t(X))$ to reduce the collision rate. Similarly, the solution X is in tabu state iff $\bigwedge_{t=1}^{3} ST^t(h^t(X)) = 1$. The hash function h^t is defined as Eq. (12).

$$h^t(X) = \sum_{j \in X} (\lfloor j^{\alpha_t} \rfloor \bmod L) \tag{12}$$

4 Experiments and Analysis

In order to evaluate the performance of the WTS algorithm, we conduct extensive experiments on 413 public benchmark instances, and compare WTS with the state-of-the-art algorithms in the literature, including GRASP-VNS [López-Sánchez *et al.*, 2019], ILS [Londe *et al.*, 2021], and BRKGA [Londe *et al.*, 2021].

Algorithm	375 Closed Instances			38 Open Instances		
	Count	Hit (%)	CPU	Count	Hit (%)	CPU
WTS	375	100.00	0.11	38	100.00	0.14
BRKGA-NLS	345	79.54	-	35	77.96	-
BRKGA-FI	347	81.20	-	32	66.64	-
BRKGA-BI	359	86.57	-	29	55.43	-
GRASP-VNS	96/131	74.42	72	0/1	0	150
ILS	96	6.90	-	13	21.12	-
CPLEX	365	97.33	12461	0	0	604800

Table 2: Computational results on instances of pmed Dataset.

Instance	n	p	LB	CPLEX		WTS	
				UB	CPU	UB	CPU
pmed6_200_30	200	30	60	69	604800	**66**	0.10
pmed16_350_40	350	40	33	35	604800	**34**	3.40
pmed21_450_90	450	50	28	29	604800	**28**	5.01
pmed23_450_50	450	50	31	34	604800	**31**	2.01
pmed29_550_60	550	60	25	30	604800	**25**	4.21
pmed31_650_70	650	70	22	25	604800	**22**	0.04
pmed31_700_140	700	140	21	22	604800	**21**	0.03
pmed32_650_70	650	70	22	25	604800	**22**	5.18
pmed32_650_130	650	130	22	23	604800	**22**	0.03
pmed32_650_170	650	170	22	23	604800	**22**	0.03
pmed33_700_140	700	140	22	23	604800	**22**	0.02
pmed35_750_80	750	80	20	22	604800	**20**	0.40

Table 3: Computational results on the improved pmed instances.

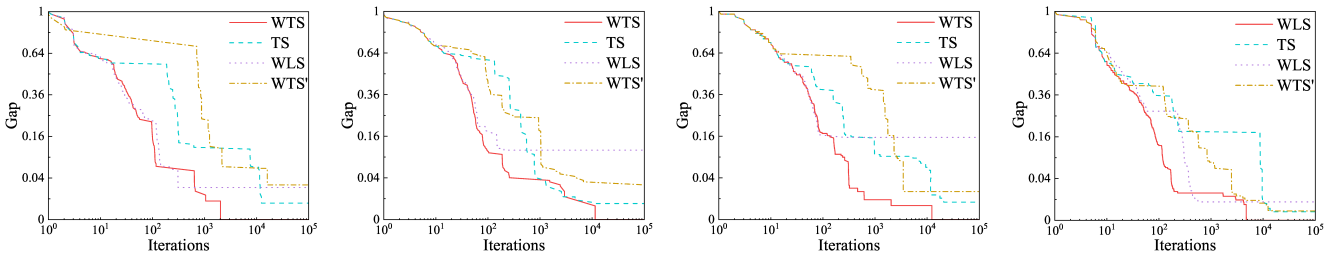
4.1 Experimental Protocol

There are 413 instances for the pNCP generated from 40 p-median instances from OR-Library [Londe *et al.*, 2021]. The number of vertices n ranges from 10 to 900, and the number of centers p distributes between 5 and 450. The shortest path between each pair of vertices is computed by Floyd algorithm [Floyd, 1962] for all instances.

All our experiments are run on Windows x64 with an Intel Core i7-10700 2.90 GHz CPU. We performed 20 independent runs on each instance under a 60-second time limit. The computational platform for the reference algorithm GRASP-VNS is Intel Core 2 Duo E8400 (3 GHz) with 4 GB RAM. The reference algorithms ILS and BRKGA are tested on an Intel Xeon E5530 CPU at 2.40 GHz and 120 GB of RAM running CentOS Linux and the time limit for each instance is 1800 seconds. The MIP models proposed by [Albareda-Sambola *et al.*, 2015] were solved with IBM ILOG CPLEX 12.10 solver and the cutoff time for each instance was 7 days.

4.2 Computational Results

In order to test the performance of our WTS algorithm, we report the detailed performance metrics and computational results in Tables 2 and 3.



(a) pmed16_350_40 (b) pmed21_450_50 (c) pmed29_550_60 (d) pmed32_650_70

Figure 2: Evolution of the objective value gaps by WTS, TS, WLS, and WTS′ on four largest instances.

"Closed Instances" and "Open Instances" represent the group of instances whose optimality has been proven and the group of instances that have not yet been proven to be optimal, respectively. Column "Count" describes the number of instances in which the algorithm found the optimal or best solutions. Column "Instance" gives the names of instances. Columns "n" and "p" report the numbers of vertices and centers, respectively. Column "Hit" shows the success rate of the algorithm for reaching the best result it has obtained under the given time limit. Column "CPU" indicates the normalized average total CPU execution time in seconds. If an algorithm did not report the result, the corresponding item will be marked with a hyphen "-". Column "LB" gives the lower bound of each instance calculated by the MIP solver CPLEX in 7 days [Londe *et al.*, 2021]. Column "UB" represents the best upper bound obtained by these algorithms, and the numbers in bold stand for the best known results.

Table 2 compares the overall results on all instances obtained by GRASP-VNS [López-Sánchez *et al.*, 2019], CPLEX [Londe *et al.*, 2021], BRKGA (BRKGA-NLS, BRKGA-FI, and BRKFA-BI) [Londe *et al.*, 2021], ILS [Londe *et al.*, 2021] and our WTS. In particular, GRASP-VNS only reports 132 instances from pmed1 to pmed8 (except pmed5). We can observe that WTS obtains the optimal results and the best known results for all the instances. Specifically, on the closed instances, our WTS solves 10 more instances to optimality comparing to the previous best known results, which are obtained for the first time, while on the open instances, it improves the previous best known results on 3 instances. Moreover, the average CPU time of WTS is less than 0.2 seconds. WTS is very stable since it obtains the reported results with 100% success rate in all 20 independent runs, while no reference algorithm can achieve this overall performance.

Table 3 shows the details of the 12 improved instances. Note that BRKGA and ILS did not report detailed results for these instances. Except for instances pmed6_200_30 and pmed16_350_40, the upper bound obtained by our WTS algorithm is equal to the lower bound obtained by CPLEX, i.e., LB = UB. This demonstrates that the lower bounds of these instances are their optimal solutions, while CPLEX cannot prove the optimality for these 10 instances within the time limit of 7 days. In summary, these statistics indicate that WTS is highly effective, efficient, and stable for solving the pNCP.

4.3 Importance of WTS Strategies

In order to evaluate the merits of the solution-based tabu search strategy, the weighting technique, and the hierarchical penalty strategy for neighborhood evaluation, we compare WTS with three alternative versions.

(1) **WLS:** Disable the solution-based tabu strategy.

(2) **TS:** Deactivate the weighting technique.

(3) **WTS′:** Disable the hierarchical penalty strategy, i.e., the objective function uses Eq. (10).

We conduct experiments on four representative instances (pmed16_350_40, pmed21_450_90, pmed29_550_60, and pmed32_650_70). Figure 2 depicts the evolution of the objective value gaps of WTS, WLS, TS, and WTS′ as the search proceeds. Each point (x, y) on the curves represents that the gap between the cost of the current solution and the best known one is y at x iterations.

From Figure 2, one can observe that WTS is the best one of the four versions. Specifically, WLS can obtain similar solution quality to WTS in a short time. However, the convergence rate of WLS decreases constantly in the later search stage. The reason for this phenomenon might be that WLS keeps searching in a loop for previously visited solutions. Furthermore, we can observe that both TS and WTS′ optimize the solution much slower than WTS. In addition, WLS, TS, and WTS′ cannot reach the best solution in 10^5 iterations, while WTS can obtain the best solution within about 10^4 iterations. These observations indicate that the solution-based tabu strategy, the weighting technique, and the hierarchical penalty strategy all play important roles, and they are essential for the effectiveness and efficiency of WTS.

5 Conclusion

In this paper, we study a variant of the p-center problem, named the p-next center problem (pNCP), and propose a weighting-based tabu search (WTS) algorithm for solving this challenging NP-hard problem. We solve the original pNCP by decomposing and solving a series of decision subproblems. Tested on 413 widely used benchmark instances and compared with several state-of-the-art algorithms in the literature, WTS shows its advantages in solving the pNCP. It improves the best known results on 12 instances, while matching the best records in the literature for all the remaining ones. Thus, we can conclude that WTS is highly competitive in terms of the effectiveness and the efficiency.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 72101094.

References

[Albareda-Sambola *et al.*, 2015] Maria Albareda-Sambola, Yolanda Hinojosa, Alfredo Marín, and Justo Puerto. When centers can fail: A close second opportunity. *Comput. Oper. Res.*, 62:145–156, 2015.

[Beasley, 1990] John E Beasley. OR-library: distributing test problems by electronic mail. *J. Oper. Res. Soc.*, 41(11):1069–1072, 1990.

[Calik and Tansel, 2013] Hatice Calik and Barbaros C Tansel. Double bound method for solving the p-center location problem. *Comput. Oper. Res.*, 40(12):2991–2999, 2013.

[Drezner, 1987] Zvi Drezner. On the rectangular p-center problem. *Nav. Res. Logist.*, 34(2):229–234, 1987.

[Elloumi *et al.*, 2004] Sourour Elloumi, Martine Labbé, and Yves Pochet. A new formulation and resolution method for the p-center problem. *INFORMS J. Comput.*, 16(1):84–94, 2004.

[Floyd, 1962] Robert W Floyd. Algorithm 97: shortest path. *Commun. ACM*, 5(6):345, 1962.

[Gao *et al.*, 2015] Chao Gao, Xin Yao, Thomas Weise, and Jinlong Li. An efficient local search heuristic with row weighting for the unicost set covering problem. *Eur. J. Oper. Res.*, 246(3):750–761, 2015.

[Glover, 1989] Fred Glover. Tabu search-part I. *ORSA J. Comput.*, 1(3):190–206, 1989.

[Hakimi, 1964] S Louis Hakimi. Optimum locations of switching centers and the absolute centers and medians of a graph. *Oper. Res.*, 12(3):450–459, 1964.

[Hochbaum and Shmoys, 1985] Dorit S Hochbaum and David B Shmoys. A best possible heuristic for the k-center problem. *Math. Oper. Res.*, 10(2):180–184, 1985.

[Irawan *et al.*, 2016] Chandra Ade Irawan, Said Salhi, and Zvi Drezner. Hybrid meta-heuristics with vns and exact methods: application to large unconditional and conditional vertex p-centre problems. *J. Heuristics*, 22(4):507–537, 2016.

[Kariv and Hakimi, 1979] Oded Kariv and S Louis Hakimi. An algorithmic approach to network location problems. I: The p-centers. *SIAM J. Appl. Math.*, 37(3):513–538, 1979.

[Lai *et al.*, 2018] Xiangjing Lai, Jin-Kao Hao, Fred Glover, and Zhipeng Lü. A two-phase tabu-evolutionary algorithm for the 0–1 multidimensional knapsack problem. *Inf. Sci.*, 436:282–301, 2018.

[Liu *et al.*, 2020] Xiaolu Liu, Yuan Fang, Jiaming Chen, Zhouxing Su, Chumin Li, and Zhipeng Lü. Effective approaches to solve p-center problem via set covering and sat. *IEEE Access*, 8:161232–161244, 2020.

[Londe *et al.*, 2021] Mariana A. Londe, Carlos Eduardo de Andrade, and Luciana S. Pessoa. An evolutionary approach for the p-next center problem. *Expert Syst. Appl.*, 175:114728, 2021.

[López-Sánchez *et al.*, 2019] Ana Dolores López-Sánchez, Jesús Sánchez-Oro, and Alfredo García Hernández-Díaz. GRASP and VNS for solving the p-next center problem. *Comput. Oper. Res.*, 104:295–303, 2019.

[Martinich, 1988] Joseph S Martinich. A vertex-closing approach to the p-center problem. *Nav. Res. Logist.*, 35(2):185–201, 1988.

[Minieka, 1970] Edward Minieka. The m-center problem. *SIAM Review*, 12(1):138–139, 1970.

[Mladenović *et al.*, 2003] Nenad Mladenović, Martine Labbé, and Pierre Hansen. Solving the p-center problem with tabu search and variable neighborhood search. *Networks*, 42(1):48–64, 2003.

[Mladenović *et al.*, 2007] Nenad Mladenović, Jack Brimberg, Pierre Hansen, and José A.Moreno-Pérez. The p-median problem: A survey of metaheuristic approaches. *Eur. J. Oper. Res.*, 179(3):927–939, 2007.

[Pullan, 2008] Wayne Pullan. A memetic genetic algorithm for the vertex p-center problem. *Evol. Comput.*, 16(3):417–436, 2008.

[Su *et al.*, 2020] Zhouxing Su, Shihao Huang, Chungen Li, and Zhipeng Lü. A two-stage matheuristic algorithm for classical inventory routing problem. In *Proc. 29th Int. Joint Conf. Artif. Intell., IJCAI 2020*, pages 3430–3436, 2020.

[Su *et al.*, 2021] Zhouxing Su, Qingyun Zhang, Zhipeng Lü, Chu-Min Li, Weibo Lin, and Fuda Ma. Weighting-based variable neighborhood search for optimal camera placement. In *Pro. 35th AAAI. Conf. Artif. Intell., AAAI 2021*, pages 12400–12408, 2021.

[Toregas *et al.*, 1971] Constantine Toregas, Ralph Swain, Charles ReVelle, and Lawrence Bergman. The location of emergency service facilities. *Oper. Res.*, 19(6):1363–1373, 1971.

[Wang *et al.*, 2017] Yang Wang, Qinghua Wu, and Fred Glover. Effective metaheuristic algorithms for the minimum differential dispersion problem. *Eur. J. Oper. Res.*, 258(3):829–843, 2017.

[Yin *et al.*, 2017] Ai-Hua Yin, Tao-Qing Zhou, Jun-Wen Ding, Qing-Jie Zhao, and Zhi-Peng Lv. Greedy randomized adaptive search procedure with path-relinking for the vertex p-center problem. *J. Comput. Sci. Technol.*, 32(6):1319–1334, 2017.

[Zhang *et al.*, 2020] Qingyun Zhang, Zhipeng Lü, Zhouxing Su, Chumin Li, Yuan Fang, and Fuda Ma. Vertex weighting-based tabu search for p-center problem. In *Proc. 29th Int. Joint Conf. Artif. Intell., IJCAI 2020*, pages 1481–1487, 2020.