Computers & Operations Research 160 (2023) 106373

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

A vertex weighting-based double-tabu search algorithm for the classical p-center problem

Qingyun Zhang [a], Zhipeng Lü [a], Zhouxing Su [a,*], Chumin Li [b]

[a] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
[b] MIS, University of Picardie Jules Verne, Amiens 80090, France

ARTICLE INFO

Keywords:
p-Center problem
Location problem
Tabu search
Solution-based tabu strategy
Vertex weighting

ABSTRACT

The p-center problem, which is NP-hard, aims to select p centers from a set of candidates to serve all clients while minimizing the maximum distance between each client and its assigned center. To solve this challenging optimization problem, we transform the p-center problem into a series of decision subproblems, and propose a vertex weighting-based double-tabu search (VWDT) algorithm. It integrates a vertex weighting strategy and a double-tabu search which combines both solution-based and attribute-based tabu strategies to help the search to escape from the local optima trap. Computational experiments on totally 510 public instances in the literature show that VWDT is highly competitive comparing to the state-of-the-art algorithms. Specifically, VWDT improves the previous best known results for 84 large instances and matches the best results for all the remaining ones. Apart from the improvements in solution quality, VWDT is much faster than other state-of-the-art algorithms in the literature, especially on some large instances. Furthermore, we perform additional experiments to analyze the impact of the key components to VWDT, such as the vertex weighting and the double-tabu search strategy.

1. Introduction

The p-center problem is a challenging optimization problem which has been proven to be NP-hard (Kariv and Hakimi, 1979). It requires locating p centers from a set of candidate centers to provide services for a set of clients, where each client seeks one of its closest centers for service. There is a serving arc between each client and its service provider. The objective of the p-center problem is to minimize the longest serving arc between a client and its service provider, where we refer to the length of the longest serving arc as the covering radius.

The p-center problem widely exists in real-world applications. For example, it can formulate important problems in telecommunication such as the wireless sensor network optimization (Liao and Ting, 2018; Zhang et al., 2021). In the context of city planning, the p-center problem can formulate the problems of determining the locations of emergency centers (Toregas et al., 1971), hospitals (Hakimi, 1964), banks (Xia et al., 2010), and charging stations (Rigas et al., 2018) to serve the communities. Similarly, the p-center problem can also model production plants location and warehouses distribution in the supply-chain management (Amiri, 2006).

Tabu search is an optimization algorithm originally introduced by Glover (1989). It incorporates the tabu strategy into the local search framework in order to escape from the local optima. Traditional attribute-based tabu strategy usually records limited features of recently visited solutions, and its performance may be sensitive to a parameter called tabu tenure. Recently, solution-based tabu search methods have attracted much attention and have been 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 maximum min-sum dispersion problem (Wang et al., 2021). The solution-based tabu strategy tracks the search trajectories in a more systematic way by recording the visited complete solution vectors. The weighting technique is another powerful technique for jumping out of local optima trap which is very flexible and adaptive. It is similar to the guided local search (Voudouris and Tsang, 2003) and has been successfully applied to many problems, such as set covering problem (Gao et al., 2015), minimum vertex cover problem (Cai et al., 2011, 2013), and satisfiability problem (Luo et al., 2012).

In this paper, we present a metaheuristic algorithm, called vertex weighting-based double-tabu search (VWDT) algorithm for solving the p-center problem. VWDT incorporates a vertex weighting technique and a tabu search which integrates both solution-based and attribute-based

* Corresponding author.
E-mail addresses: qingyun_zhang@hust.edu.cn (Q. Zhang), zhipeng.lv@hust.edu.cn (Z. Lü), suzhouxing@hust.edu.cn (Z. Su), chu-min.li@u-picardie.fr (C. Li).

https://doi.org/10.1016/j.cor.2023.106373
Received 20 October 2022; Received in revised form 3 August 2023; Accepted 5 August 2023
Available online 9 August 2023

tabu strategies. Rather than directly tackling the original optimization problem as the previous metaheuristics, VWDT works on the decision version to solve the p-center problem. Specifically, we transform the p-center problem into a series of decision subproblems. That is to say, for each possible covering radius r, we check if all vertices can be covered by a center within the covering radius r. It is equivalent to solving a series of maximal covering location subproblems (Church and ReVelle, 1974). In fact, by utilizing the lower and upper bounds of the original problem, the number of subproblems will be relatively small and we can solve them in parallel due to their independence from each other. Tested on three sets of totally 174 classical benchmark instances and one set of 336 massive instances of the p-center problem, VWDT improves the best known results on 84 instances and matches the records on the remaining 420 ones. Moreover, the computational time for reaching these results is much shorter than that of the state-of-the-art reference algorithms in the literature.

A preliminary version of this algorithm named VWTS was proposed in Zhang et al. (2020), which was extended and improved here in the following aspects:

1. We adopt a reduction method to simplify instances and further improve the search efficiency of the algorithm.
2. In addition to the original vertex weighting technique, by combining solution-based and attribute-based tabu strategies, we use a new hybrid double-tabu search framework to replace the original simple tabu strategy to achieve a better balance between intensification and diversification and prevent possible cycles during the search.
3. We use an age strategy to break ties when there exist multiple choices of equal neighboring solutions, which prefers to select the sets that have not been moved into or out of the candidate solution for a long time.
4. Our new improved algorithm is tested on larger instances used in the literature with more than 200,000 vertices, while the largest instance in Zhang et al. (2020) contains only 3038 vertices. Our algorithm is able to improve the previous best known results for 84 instances on these large scale instances.
5. New best known results are obtained on 4 difficult instances from pcb3038 and the computational efficiency has also been significantly improved.
6. We conduct additional experimental analysis to show the significance of the key components of our algorithm.

The paper is outlined as follows. Section 2 reviews the previous research work related to the p-center problem. Section 3 introduces the definition and reformulations for the p-center problem. In Section 4, the proposed vertex weighting-based double-tabu search algorithm is described. In Section 5, we present the computational experiments to compare VWDT with the state-of-the-art algorithms in the literature. Section 6 analyzes and compares some essential ingredients of the algorithm to show how they affect the performance of the algorithm. Section 7 concludes the paper and presents future research directions.

2. Related works

As one of the most classical location problems introduced by Hakimi (1964), the p-center problem has drawn much attention from academic society in the last 50 years and various intelligent optimization algorithms have been proposed. There are mainly three kinds of methodologies for the p-center problem, which are exact, approximation, and metaheuristic algorithms.

Regarding the exact algorithms, a popular framework for solving the p-center problem is to solve a series of minimum set covering problems (Daskin, 2013). Minieka (1970) was among the first researchers who use this approach to tackle the p-center problem. Specifically, Minieka (1970) presented an iterative algorithm that solves a set covering problem with the given radius at each iteration that verifies whether all clients can be covered within the radius using no more than p centers. Daskin (2000) exploited the relationship between the p-center problem and the set covering problem, and solved the latter one using Lagrangian relaxation. Ilhan et al. (2002) proposed a two-phase algorithm. The first phase finds the best possible lower bound by iteratively judging the feasibility of LP formulation in the given radius. The second phase improves the obtained lower bound by solving a series of integer programming (IP) models until the IP formulation becomes feasible, i.e., a feasible covering radius where all clients are covered is found. Elloumi et al. (2004) presented a new mixed-integer programming (MIP) model for the p-center problem and a polynomial algorithm for computing a tighter lower bound. They also proposed an extension called fault-tolerant p-center problem. Calik and Tansel (2013) introduced a new integer programming formulation and an exact algorithm based on decomposition technique for solving the p-center problem, where a relaxation of the proposed formulation gives a tighter lower bound than that in Elloumi et al. (2004). Al-Khedhairi and Salhi (2005) introduced some modifications to the algorithms of Ilhan et al. (2002) and Daskin (2000) to reduce the number of ILP iterations during the search and the number of subproblems to be solved. Contardo et al. (2019) proposed a row generation approach that iteratively solves small covering subproblems of the p-center problem. Later, Liu et al. (2020) proposed a method for solving the p-center problem via set covering and SAT, which is the first SAT-based method for solving the p-center problem. Based on the previous best solution obtained by PBS (Pullan, 2008) and GRASP/PR (Yin et al., 2017), it improved the best known results on 3 large-scale instances and proved the optimality of the obtained solutions. Gaar and Sinnl (2022) presented a new integer programming formulation that can be obtained by a projection from the classical formulation of p-center problem, and solved it by branch-and-cut method. The algorithms proposed in Contardo et al. (2019), Liu et al. (2020), and Gaar and Sinnl (2022) are the best exact algorithms for solving the p-center problem.

Besides the exact methods, a variety of approximation algorithms have been proposed for solving the p-center problem in the last few decades. Hochbaum and Shmoys (1985) presented a 2-approximation algorithm for the p-center problem with triangle inequality. Martinich (1988) used a vertex-closing approach to solve the p-center problem and proposed several theorems for verifying optimal solutions during the search. Initially, all centers are opened, and then $n - p$ centers are selected to be closed according to the optimization objective of the p-center problem. Garcia-Diaz et al. (2019) conducted an analytical study and experimental evaluation of the most representative approximation algorithms which are some 2-approximation (Sh Shmoys, 1995 and Gon Gonzalez, 1985) and 3-approximation (CDS Garcia-Diaz et al., 2017) algorithms for the p-center problem.

Metaheuristic is an effective methodology for solving combinatorial optimization problems. In the last few decades, many metaheuristic algorithms for solving the p-center problem have been proposed. Mladenović et al. (2003) presented a variable neighborhood search (VNS), a multistart local search, and a chain substitution tabu search metaheuristic for the p-center problem. They conducted experiments on 40 p-median (Mladenović et al., 2007) instances from the OR-Library and 98 instances with up to 3038 vertices from TSPLIB. Caruso et al. (2003) proposed Dominant, a metaheuristic algorithm, to solve a series of set covering problems by predefined covering radius. Robič and Mihelič (2005) introduced a polynomial time heuristic algorithm for the minimum dominating set problem, which uses the so-called scoring technique, to solve the vertex-restricted p-center problem. Pullan (2008) presented a memetic algorithm, called PBS, which is a population-based metaheuristic algorithm based on a local search procedure. By using phenotype crossover and directed mutation operators, a population of elite initial solutions are generated and a local search algorithm improves the solutions to local optima. Salhi and Al-Khedhairi (2010) designed a three-level metaheuristic that combines VNS with perturbation schemes, and integrated it into exact algorithms such as Daskin (2000) to solve the p-center problem. Irawan et al. (2016) introduced a three-stages method with two meta-heuristics, which combines VNS, exact method, and aggregation techniques for solving large p-center problems. Yin et al. (2017) presented a GRASP/PR algorithm for the p-center problem, which combines path-relinking with greedy randomized adaptive search procedure (GRASP). It improves the previous best known results for 10 large-scale instances. Ferone et al. (2017) presented a new smart local search based on the critical vertex concept, called GRASP+plateau-surfer, and embedded it into a GRASP framework. Zhang et al. (2020) introduced a vertex weighting-based tabu search algorithm, which is the best metaheuristic algorithm for solving the p-center problem.

3. Problem description

The p-center problem is defined on an undirected complete graph $G = (V, E)$, where $V = \{1, 2, \dots, n\}$ is the vertex set and E is the edge set. The distance between each pair of vertices i and j is $d_{ij} \in \mathbb{R}^+$. Each vertex $i \in V$ corresponds to a client that requires services from one of the p centers, which should be selected from a set of candidate centers $C \subseteq V$ (usually $C = V$). The solution can be defined as $(\boldsymbol{x}, \boldsymbol{y}, r)$. In detail, $\boldsymbol{x} = \{x_j | j \in C\}$ and x_j is a decision variable which equals to 1 iff a candidate center $j \in C$ is opened as a center. $\boldsymbol{y} = \{y_{ij} | i \in V, j \in C\}$ and y_{ij} is a binary variable, where $y_{ij} = 1$ iff vertex $i \in V$ is served by center $j \in C$. $r \in \mathbb{R}^+$ is the length of the longest serving arc, i.e., the covering radius. Based on the above notations, we can formulate the classical MIP model for the p-center problem (Daskin, 2013) as follows.

$$(PC) \qquad \min \; r, \tag{1}$$

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

$$\sum_{j \in C} y_{ij} = 1, \forall i \in V, \tag{3}$$

$$y_{ij} \le x_j, \forall i \in V, \forall j \in C, \tag{4}$$

$$\sum_{j \in C} d_{ij} y_{ij} \le r, \forall i \in V, \tag{5}$$

$$x_j, y_{ij} \in \{0, 1\}, r \in \mathbb{R}^+, \forall i \in V, \forall j \in C. \tag{6}$$

In model (PC), objective (1) aims to minimize the covering radius. Constraint (2) enforces that there are p opened centers. Constraints (3) ensure that each vertex must be served by exactly one center. Constraints (4) restrict that only the opened centers can serve the vertices. Constraints (5) make sure that the covering radius r is not shorter than any serving arc. In the optimal solution, there will be at least one equality among constraints (5), otherwise r can be reduced further. This fact implies that the optimal covering radius for a specific p-center instance must be the same as the length of a certain edge d_{ij}. Thus, for an ordered list of distinct edge length $R = \{d_{ij} | i \in V, j \in C\} = \{r_1, r_2, \dots, r_k\}$ where $r_1 < r_2 < \dots < r_k$, we seek for the smallest rank q such that model (PC) is still feasible after adding a constraint $r \le r_q$, but it becomes infeasible if we add constraint $r \le r_{q-1}$. Then, such r_q is the optimal covering radius.

Based on the relationship between the maximal covering location problem and the p-center problem, if we fix the edge length rank q of an instance, the p-center problem is equivalent to the maximal covering location problem (Church and ReVelle, 1974). Specifically, for each candidate center $j \in C$, let $S_j^q = \{i \in V | d_{ji} \le r_q\}$ denotes the set of vertices that can be served by candidate center j within radius r_q. We can obtain a maximal covering location problem instance $S^q = \{S_j^q | j \in C\}$. Then, if we can select p sets out of S^q whose union includes all vertices, we can derive that r_q is an upper bound of the covering radius in the original p-center problem, and vice versa. Therefore, if the optimal edge length rank q is known, we can reformulate model (PC) to the maximal covering location problem defined by Eqs. (7)–(10) (Daskin, 2000; Ilhan et al., 2002), where the meaning of each decision variable x_j in model (MCL_q) is identical to the corresponding one in model (PC), and u_i is a binary variable where $u_i = 1$ if vertex i is not covered by any center.

$$(MCL_q) \qquad \min \sum_{i \in V} u_i, \tag{7}$$

$$\text{s.t.} \sum_{j \in C, d_{ij} \le r_q} x_j \ge 1 - u_i, \forall i \in V, \tag{8}$$

$$\sum_{j \in C} x_j = p, \tag{9}$$

$$x_j, u_i \in \{0, 1\}, \forall i \in V, \forall j \in C. \tag{10}$$

Here, we adopt the equivalent objective of maximizing the number of covered vertices, i.e., minimizing the number of uncovered vertices as shown in objective function (7). Constraints (8) guarantee that each uncovered vertex will be counted into the objective. Constraint (9) enforces that there is exactly p opened centers. In order to find the optimal covering radius, we need to traverse the distinct edge length list R to examine each possible covering radius. For each r_q, we judge if there is a feasible solution for the p-center problem, i.e., there are p sets that cover all vertices within the radius r_q. We repeatedly choose next q until r_q is invalid. Then, r_{q+1} is the best solution for the original problem.

4. Vertex weighting-based double-tabu search

4.1. General framework

Our VWDT algorithm starts from an upper bound r_q obtained by heuristics (e.g., PBS Pullan, 2008) for model (PC) under limited time, and then solves models (MCL_{q-1}), (MCL_{q-2}), …, (MCL_1) in turn until it fails to find a feasible solution within the given time limit. Therefore, we will focus on the subroutine named VWDT$_q$ for solving model (MCL_q) with a given radius.

Algorithm 1 gives the main framework of VWDT$_q$. After initializing vectors and variables, as shown in lines 1–2, the proposed VWDT$_q$ starts with an initial solution generated by a constructive heuristic algorithm (line 3, Section 4.2), and the set of opened centers are recorded in X, X' and X^*. Then, VWDT$_q$ iteratively searches for a better solution with less uncovered vertices by a vertex weighting-based double-tabu search procedure (lines 5–16). At each iteration, VWDT$_q$ first evaluates the swap-based neighborhood of the current solution and records the best neighborhood move (i, j) in non-tabu status (line 6). Then, it makes the best move which swaps in vertex i and swaps out vertex j from the current center set (line 7). If the current solution X is better than the best solution X^* found so far, the best solution is updated with X (lines 8–9), where $U(X)$ represents the set of vertices that are uncovered in the current solution X. Otherwise, when the current solution falls into a local optimum, i.e., the best move returned by function FindPair() cannot cover more vertices (line 10), VWDT$_q$ will increase the weight of each uncovered vertex (lines 11–12), where δ will be introduced in Section 4.4. At the end of each iteration, the tabu lists and other data structures will be updated (lines 14–15). Finally, once the time limit or other user-specified termination condition is met, VWDT$_q$ terminates and returns the best solution X^* (line 17).

4.2. Initialization

The initialization consists of a reduction procedure and the construction of the initial solution. First, some centers must be included in the optimal solution. If a vertex is only covered by itself, then we must pick the vertex to cover it. Apparently, such kind of vertices can be fixed constantly as centers, and then we only need to choose $p - l$ centers, where l is the number of fixed centers.

Next, VWDT$_q$ employs a constructive heuristic for selecting p centers to form an initial solution X. Let S_j denote the set of vertices that

Algorithm 1 The main framework of the VWDT$_q$ algorithm

```
Input: A graph $G$, the center number $p$, the radius $r_q$
Output: The best solution found so far $X^*$
 1: Solution and attribute tabu lists $SL \leftarrow \emptyset$, $AL_j \leftarrow 0, \forall j \in C$
 2: $iter \leftarrow 1$, vertex weights $w_i \leftarrow 1, \forall i \in V$, vertex age $a_j \leftarrow 0, \forall j \in C$
 3: Current solution $X \leftarrow$ Initialize$(G, p, r_q)$        ▷ Section 4.2
 4: Previous solution $X' \leftarrow X$, $X^* \leftarrow X$
 5: while termination condition is not met do
 6:     $(i, j) \leftarrow$ FindPair$(X', SL, AL, iter)$        ▷ Section 4.4
 7:     MakeMove$(X, i, j)$
 8:     if $|U(X)| < |U(X^*)|$ then
 9:         $X^* \leftarrow X$        ▷ $U(X)$ is uncovered vertex set of $X$
10:     else if $|U(X)| > |U(X')|$ then
11:         $w_k \leftarrow w_k + 1, \forall k \in U(X)$        ▷ Section 4.3
12:         $\delta_l \leftarrow \delta_l + 1, \forall l \in S_k, k \in U(X)$
13:     end if
14:     $SL \leftarrow SL \cup \{X\}$, $AL_j \leftarrow iter + 2$
15:     $a_j \leftarrow iter, X' \leftarrow X$, $iter \leftarrow iter + 1$
16: end while
17: return $X^*$
```

candidate center j can serve within the current radius, we have $U(X) = V \setminus \cup_{j \in X} S_j$. Conversely, we denote the set of candidate centers serving vertex i by C_i, i.e., $C_i = \{j \in C | i \in S_j\}$. The constructive heuristic iteratively opens (i.e., inserts centers into the current solution X) the current best candidate center $j^* = \arg\max_{j \in C \setminus X} |S_j \cap U(X)|$ which covers most uncovered vertices until p centers are opened. Ties are broken randomly if there are multiple current best candidate centers covering the same number of uncovered vertices. Observing that choosing the best center at each iteration takes $O(n)$ time and there are p centers in total. Thus, the time complexity of the construction procedure is $O(np)$.

4.3. Weighting technique

The quality of the initial solution is usually poor, so we employ a weighting-based tabu search to iteratively improve the solution obtained by the constructive heuristic. The main idea obtained by the vertex weighting technique is to help the search to escape from the local optima by altering the objective function. As presented in Eq. (7), we adopt a straightforward objective function which minimizes the number of uncovered vertices. When vertex weight w_i is introduced, we replace the objective function $f(X)$ with Eq. (11).

$$\min f(X) = \sum_{i \in V} w_i u_i. \tag{11}$$

As a result, VWDT$_q$ actually tackles model (SC_q^w) which is composed of Eqs. (8)–(11). Note that weight w_i is not a predefined value, instead, it varies as the search proceeds. If a vertex is not covered for a long time during the search process, it implies that this vertex is critical for achieving the full coverage and we should treat it with higher priority. Specifically, when the tabu search encounters the local optimal solution X, VWDT$_q$ increases the weight w_i of each uncovered vertex $i \in U(X)$ by one (Algorithm 1, lines 11–13). Nevertheless, the optimal value of objective function (11) is always zero regardless of the configuration of the weights. The updated weights reshape the landscape of the solution space so that X is no longer the local optimum. When encountering stagnation, the more frequently a vertex appears in $U(X)$, the greater its weight will be. Thus, it guides the search to escape from the local optima and continues to explore other solutions.

Comparing to the tabu strategy, the vertex weighting technique diversifies the search in an adaptive manner. On the one hand, it focuses on the consequences instead of the causes, which prevents uncovered vertices from appearing rather than forbids opening or closing centers like the tabu strategies. On the other hand, it modifies the solution space in a smooth way instead of a black-or-white one, i.e., the neighborhood moves become better or worse rather than valid or invalid.

Note that at each iteration of local search, when there exist multiple best neighboring solutions according to the weighted objective function Eq. (11), we employ an age-based strategy to break ties. Specifically, the age of a candidate center denotes the most recent iteration at which it was a center in the local search procedure. If a candidate center has never been opened as a center, its age is zero. The smaller the age of a candidate center is, the longer time the candidate center has not been opened for. Thus, we break ties by favoring the neighboring solution with smaller age value. The age of vertex j is denoted as a_j. This strategy is able to diversify the search when the weighted objective function cannot differentiate equal solutions.

4.4. Neighborhood structure and evaluation

To improve the generated initial solution for model (MCL_q^w), VWDT$_q$ adopts a swap-based neighborhood inspired by the classical one used by most metaheuristics for the p-center problem (Pullan, 2008). However, the neighborhood evaluation is quite different in the reformulated model, as well as the dedicated acceleration strategies. In detail, a swap move Swap(i, j) produces a neighboring solution $X \oplus \text{Swap}(i, j) = X \cup \{i\} \setminus \{j\}$, by adding a candidate center $i \in C \setminus X$ to the center set (open), and removing center $j \in X$ from the center set (close).

As we know, the neighborhood evaluation is the most time-consuming routine in local search-based metaheuristic algorithms. Under the best improvement policy, the local search evaluates all feasible moves at each iteration, and performs the best neighborhood moves providing the greatest improvement to the objective value. For the neighborhood evaluation of swap move at each iteration, the time complexity is $O(p(n-p))$ so that it could be time-consuming on large instances. In order to overcome the performance issue, VWDT$_q$ employs an incremental evaluation technique and a neighborhood sampling strategy.

On the one hand, to accelerate the neighborhood evaluation process, the VWDT$_q$ algorithm adopts an incremental evaluation mechanism to efficiently evaluate all neighborhood moves by storing and maintaining $\delta_j (j \in C)$ as Eq. (12), instead of naively calculating the objective function Eq. (11).

$$\delta_j = \sum_{i \in (S_j \cap U(X \setminus \{j\}))} w_i. \tag{12}$$

where δ_j records the consequence of flipping the open state of candidate center j. Specifically, δ_j for center $j \in X$ is the sum of the weights of the vertices which are only covered by center j. For each candidate center $j \notin X$, δ_j is the sum of the weights of all uncovered vertices in S_j. Then, we can incrementally evaluate the objective value for opening or closing a center in $O(1)$ time complexity. However, we should update the affected δ values every time after opening or closing a center. Specifically, we can calculate the objective value of each neighboring solution by $f(X \cup \{i\}) = f(X) - \delta_i$, and update the related δ values, and thus $f(X \oplus \text{Swap}(i, j)) = f(X \cup \{i\}) + \delta_j$. As we know, at each iteration, a typical tabu search algorithm evaluates many moves but only performs a single one, and a single neighborhood move only brings minor changes to the current solution. Therefore, we can benefit from maintaining and querying the cache rather than calculating the objective function from scratch for each move.

On the other hand, the objective value cannot be improved by covering already covered vertices, while it can only be improved when the uncovered vertices are covered. Therefore, VWDT$_q$ will evaluate a swap move Swap(i, j) only if candidate center j covers some uncovered vertices in $U(X)$. Because each vertex needs to be eventually covered, VWDT$_q$ explores the neighborhood in a more compact way to further reduce the size of the neighborhood. Specifically, it picks a random

Table 1
Test environment of the reference algorithms.

Algorithm	Environment	Score
ELP (Elloumi et al., 2004)	400 MHz Pentium II CPU and 384 MB RAM	272
VNS (Mladenović et al., 2003)	Sun Sparc Station 10	–
PBS (Pullan, 2008)	AMD Opteron 252 2.6 GHz CPU	760
DBR2 (Calik and Tansel, 2013)	–	–
TSA, GMA (Irawan et al., 2016)	Intel Core i5-650 3.2 GHz CPU, 4 GB of RAM	1375
EM(Z^*) (Irawan et al., 2016)	Intel Core 2 Duo 2.6 GHz CPU, 8 GB of RAM	1100
GRASP/PR (Yin et al., 2017)	Intel Xeon E5-2609v2 2.5 GHz CPU and 32 GB RAM	1356
GRASP+PS (Ferone et al., 2017)	Intel Xeon E5-4610v2 2.30 GHz CPU	1474
CIK (Contardo et al., 2019)	Intel Xeon E5462 2.8 GHz CPU AND 16GB RAM	1138
Gon+, CDSh, CDSh+ Garcia-Diaz et al. (2019)	Intel Core i5 2.3 GHz CPU and 24 GB RAM	1568
SAT-sol (Liu et al., 2020)	Intel Xeon E5-2609v2 2.5 GHz CPU and 32 GB RAM	1356
VWTS (Zhang et al., 2020)	Intel Xeon E5-2609v2 2.5 GHz CPU and 32 GB RAM	1356
fCLH (Gaar and Sinnl, 2022)	Intel Xeon E5-2670v2 2.5 GHz and 32GB RAM	1642

Algorithm 2 Find the best swap pair

```
1: function FindPair($X, SL, AL, iter$)
2:     Best swap move $(i^*, j^*) \leftarrow \emptyset$
3:     Best objective value in the neighborhood $obj \leftarrow +\infty$
4:     $k \leftarrow$ a random vertex in $U(X)$
5:     $\delta'_j \leftarrow \delta_j, \forall j \in C$
6:     for all $i \in C_k$ do
7:         TryToOpenCenter($i$)                              ▷ Algorithm 3
8:         for all $j \in X$ do                   ▷ Evaluate closing center $j$
9:             if NotTabu($Swap(i,j), X, SL, AL, iter$) then
10:                if $f(X \oplus \text{Swap}(i,j)) < obj$ then
11:                    $obj \leftarrow f(X \oplus \text{Swap}(i,j))$
12:                    $(i^*, j^*) \leftarrow (i,j)$
13:                else if $(f(X \oplus \text{Swap}(i,j)) = obj) \wedge (a_j < a_{j^*})$ then
14:                    $(i^*, j^*) \leftarrow (i,j)$
15:                end if
16:            end if
17:        end for
18:        $\delta_j \leftarrow \delta'_j, \forall j \in C$
19:    end for
20:    return $(i^*, j^*)$
21: end function
```

Algorithm 3 Open a center virtually

```
1: function TryToOpenCenter($i$)
2:     for all $v \in S_i$ do
3:         if $|X \cap C_v| = 1$ then
4:             $\delta_l \leftarrow \delta_l - w_v$, for $l \in X \cap C_v$
5:         end if                        ▷ $l$ was the only center covering $v$
6:     end for
7: end function
```

vertex $k \in U(X)$, and only evaluates moves Swap(i,j) which satisfy $i \in C_k$ and $j \in X$. This sampling strategy not only reduces the time consumption for neighborhood evaluation, but also improves the diversification of the search as a side effect and achieves a better balance between exploitation and exploration.

The neighborhood evaluation procedure is illustrated in Algorithm 2. Starting from a random vertex $k \in U(X)$ (line 4), VWDT$_q$ tries to open each candidate center $i \in C_k$ which covers vertex k, i.e., updating the corresponding δ_j, for all $j \in X$, as if candidate center i is already opened (lines 6–7), so that the objective value $f(X \oplus \text{Swap}(i,j))$ of the resulting neighboring solution can be quickly calculated. Then, we traverse each non-tabu move and record the best one and the corresponding swap move (lines 9–16). When all trial moves regarding opening candidate center i are evaluated, the δ values are restored to the value before the neighborhood evaluation begins (line 18).

Furthermore, sub-routine TryToOpenCenter() is presented in Algorithm 3. According to the objective function (12), if there is exactly one center $l \in X$ which covers vertex v before opening center i, the δ value for closing center l will decrease by w_v (lines 3–5). The reason lies in the fact that closing center l will not make vertex v uncovered anymore once center i is opened, so the penalty is reduced. We are only interested in δ_j ($\forall j \in X$) for closing a center, since lines 8–17 in Algorithm 2 only consist of closing another center. In the worst case, the time complexity of Algorithm 2 and Algorithm 3 are $O(n^2)$ and $O(n)$, respectively. Besides, when we eventually make the best move (Algorithm 1, line 7), the affected δ values should be updated in a similar way.

4.5. Hybrid double-tabu strategy

Tabu search is a powerful local search-based metaheuristic algorithm for solving a variety of combinatorial optimization problems (Glover, 1989). It is usually based on a recency-based tabu list to prohibit revisiting recently explored solutions.

Ideally, all evaluated solutions should never be visited again. However, if we explore the solution space in such a systematic way, the proposed VWDT$_q$ will be no longer a heuristic algorithm, and it will have to face the drawbacks and limitations of the exact algorithms. In contrast, the traditional attribute-based tabu search approaches usually record limited features of the recently visited solutions, and it has the ability to help the search to jump out of local optimal trap. It is simple and efficient but may misjudge the tabu status of the solution. As a consequence, promising neighborhood moves may be falsely forbidden, so synthetic parameters such as the tabu tenure must be carefully tuned. We try to combine their advantages and balance the completeness and the convergence rate when designing the VWDT$_q$ algorithm. As a result, it integrates both attribute-based and solution-based tabu strategies (Wang et al., 2017) which can lead to different search trajectories and help the search to jump out of the local optimum trap. For the solution-based tabu strategy, we do not prohibit all evaluated solutions, but only forbid the solutions that are actually visited.

Let us recall that each term x_j in the solution vector of model (SC_q^w) denotes whether vertex j is chosen to be a center. The attribute-based tabu strategy prevents closing the newly opened center immediately. In detail, the parameter of the tabu tenure tt in the attribute-based tabu strategy is fixed to two iterations. Particularly, if we set $x_j = 1$ at current iteration $iter$, it is forbidden to reset $x_j = 0$ at next tt iterations. In other words, there will be an entry $\{j\}$ in the attribute-based tabu list AL until iteration $iter + tt$.

The idea behind the solution-based tabu strategy is similar to the row generation approach in mathematical programming. When a solution X is recorded as in tabu status, it can be regarded as adding a lazy constraint $\sum_{j\in X} x_j \le p-1$ to model (MCL_q^w). If a swap move is eventually performed, a new current solution X is obtained, the tabu status of X will be saved into the solution-based tabu list SL and it will be prohibited to be visited forever. Technically, we implement the solution-based tabu list in an efficient but inaccurate way. VWDT$_q$ employs a hash function h to map a solution vector X to an integer value $h(X) \in [0, L)$ $(L = 10^8)$ and the solution-based tabu list SL is a binary vector, where $SL(h(X)) = 1$ iff solution X has been visited. In case of hash collision which leads to incorrect identification of the tabu status of some unvisited solutions, we use multiple hash functions h^t $(t = 1, 2, 3)$ to reduce the collision rate, i.e., a solution X is in tabu status iff $\bigwedge_{t=1}^{3} SL^t(h^t(X)) = 1$. We define the hash function h^t as Eq. (13), where γ_t is a constant parameter, and the values of γ_1, γ_2, γ_3 are 1.3, 1.9, 2.3, respectively.

$$h^t(X) = (\sum_{j\in X} \lfloor j^{\gamma_t} \rfloor) \bmod L \tag{13}$$

In VWDT, the solution-based and attribute-based tabu strategies are simultaneously used. When a swap move satisfies either one condition of the two tabu strategies, the VWDT algorithm will not accept it as the candidate move.

5. Experimental results and comparisons

To demonstrate the effectiveness of VWDT, we conduct extensive computational experiments on three well-known sets of 174 standard benchmark instances and a set of 336 massive instances, and compare VWDT with the state-of-the-art algorithms in the literature.

5.1. Experimental protocol

Our VWDT algorithm is programmed in C++ and compiled with Visual Studio 2017. All computational experiments are carried out in a single thread on a Windows Server 2012 × 64 with Intel Xeon E5-2609v2 2.50 GHz CPU and 32 GB RAM. We reimplement PBS algorithm (Pullan, 2008) under a 5 ms (20 s) time limit to obtain a good initial upper bound of the covering radius r_{q0} for each instance with less (more) than 70,000 vertices. For VWDT, it includes the total time for computing r_{q0} and sequentially solving all subproblems from (MCL_{q0}) to (MCL_{q^*}), where q^* is the edge length rank of the best covering radius r_{q^*}. Unless otherwise specified, the result of each instance is obtained by running VWDT in 20 independent runs, and the time limit for each decision subproblem corresponds to each covering radius is 900 s. There are four sets of instances in our experiment.

(1) **ORLIB**: The first set consists of 40 p-median problem instances (pmed) from the OR-Library[1] (Beasley, 1990). They are based on random undirected graphs where vertex number n and center number p satisfy $n \in [100, 900]$ and $p \in [5, 90]$ whose scale are relatively small.
(2) **TSPLIB**: The second set contains 98 TSP instances from the TSPLIB[2] (Reinelt, 1991). These instances are based on planar graphs and they are usually derived from real-world applications. There are 44 small instances (sTSP) whose vertex number ranges from 226 to 657, and the remaining 54 ones (u1060, rl1323, u1817, and pcb3038) are categorized as the large instances. The center number p distributes between 5 and 500.
(3) **World TSP**: The third set is composed of 36 TSP instances from the World TSP[3] (Reinelt, 1991). These instances are based on the maps of six countries, denoted as Oman (mu1979), Canada (ca4663), Tanzania (tz6117), Sweden (sw24978), Burma (bm33708), and China (ch71009), respectively. The number of vertices n ranges from 1979 to 71,009 which are explicit in their names, and the number of centers p varies from 5 to 100.

[1] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/pmedinfo.html
[2] https://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/
[3] http://www.math.uwaterloo.ca/tsp/world/countries.html

Table 2
Parameter settings.

Parameter	Value	Range	Description
tt	2	[0, 10]	The attribute-based tabu tenure
γ_1	1.3	[1, 2]	A constant parameter in Section 4.5
γ_2	1.7	[1.5, 2.5]	A constant parameter in Section 4.5
γ_3	2.3	[2, 4]	A constant parameter in Section 4.5

(4) **Massive TSP**: The last set is based on the massive graphs in TSPLIB[4] (Reinelt, 1991). The complete set was firstly used in Contardo et al. (2019), and then studied by Gaar and Sinnl (2022). The set of massive TSP instances was previously used only for testing exact algorithms. The numbers of vertices n are between 1621 and 238,025 which are explicit in their names, and the number of centers p varies from 2 to 30. According to the number of centers, the set is further divided into 9 datasets, which are P2, P3, P5, P10, P15, P20, P25, and P30, respectively.

For all instances, the distance matrix $\boldsymbol{d}$ is not given directly, so we calculate all-pair shortest paths with the Floyd algorithm (Floyd, 1962) for the ORLIB instances, and calculate the Euclidean distances for the TSPLIB, World TSP, and massive TSP instances to obtain the distance d_{ij} between each pair of vertices i and j. For the TSPLIB and World TSP, the Euclidean distances are rounded to the nearest hundredths. For the massive TSP, we round the distances to their nearest integers as in the very recent literature for fair comparison.

5.2. Reference algorithms

We compare our proposed VWDT algorithm with five exact algorithms (ELP, DBR2, CIK, SAT-sol, and fCLH), three approximation algorithms (Gon+, CDSh, and CDSh+), and six metaheuristic algorithms (VNS, PBS, GRASP/PR, GRASP+PS, TSA, and GMA) regarding the solution quality and computational time to reach the best solution. Note that for three approximation algorithms, we only report the best results among them. Meanwhile, we also compare VWDT algorithm with our preliminary VWTS algorithm (Zhang et al., 2020). Table 1 shows the experimental environments of the reference algorithms. Column Score is the single-thread rating of the processor used in each literature obtained from https://www.passmark.com. In order to compare CPU times as fairly as possible, the CPU times of all algorithms in this section are normalized by multiplying Score/1356 (the denominator is the single-thread rating of our CPU), except for VNS (Mladenović et al., 2003) and DBR2 (Calik and Tansel, 2013). For VNS and DBR2, the CPU times are not normalized due to the unclear test environment. However, because of the relatively low quality of the solutions obtained by VNS and DBR2, it does not affect the overall conclusion.

5.3. Parameter setting

The setting of parameters of VWDT is shown in Table 2, which can be considered as the default setting of the algorithm and throughout all the experiments presented in this study. The parameters of our algorithm are tested and tuned by the IRACE package (López-Ibáñez et al., 2016) within the range in column Range. The IRACE package is an automatic algorithm configuration tool that implements configuration procedures based on iterated racing. The final values of all parameters are recorded in Value. In addition, parameters γ_1, γ_2, and γ_3 are introduced to construct different hash functions. Note that the value of γ_t is insensitive to the performance of the algorithm. We only need to make sure that they are not too large to avoid overflow.

[4] https://www.math.uwaterloo.ca/tsp/data/index.html

5.4. Computational results

This section presents comprehensive investigations on the performance of VWDT.

5.4.1. Three well-known sets

Table 3 shows the overall results on three well-known sets obtained by VWDT. Columns Set and Dataset give the names of sets of benchmark instances and datasets, respectively. Column Count presents the number of instances in the dataset. Column CPU reports the average total CPU execution time in seconds on each dataset of our VWDT algorithm. Column CPU-q^* gives the run time in seconds for VWDT to solve the subproblem corresponding to the best known covering radius r_{q^*}, which shows that the performance of VWDT can be significantly enhanced when parallel computing is available. We can observe that VWDT is able to obtain the optimal solution with 100% success rate in 20 independent runs except for the three largest datasets sw24978, bm33708, and ch71009. In general, VWDT is highly stable when the number of vertices is less than 10,000 for it reaches such results in all 20 independent runs.

Tables 4 to 9 reports the detailed results. Since most reference algorithms and the proposed VWDT algorithm can obtain the optimal solutions on all instances in the pmed and sTSP datasets within one second, we omit the detailed results on these two datasets.[5] These tables follow the same convention described below. Column Ins. gives the names of instances. Columns n and p denote the numbers of vertices and centers, respectively. Column BKS represents the objective values of the best known solutions of all algorithms. For each instance whose optimal objective value can be proven, i.e., the lower bound of model (MCL_{q^*-1}) is greater than zero, it is marked with "*". Specifically, the MIP models are solved by Gurobi 8.1 under a 100-hour time limit. Columns f_{best} and f_{avg} give the best and the average objective values obtained by the corresponding algorithms, respectively. Apparently, smaller f_{best} is better according to model (PC). Column Hit shows the success rate for reaching the best known result f_{best} under the given time limit. Column CPU reports the normalized average total CPU execution time in seconds of these algorithms (see Section 5.2). If the total CPU execution time of the corresponding algorithms is not normalized due to the unclear test environment, it is represented by CPU^0. For VWDT algorithm, the CPU time includes the total time for calculating r_{q^0} using PBS and sequentially solving all subproblems from (MCL_{q^0}) to (MCL_{q^*}), where q^* is the edge length rank of the best covering radius r_{q^*}. Column CPU-q^* gives the run time in seconds for VWDT to solve the subproblem corresponds to the best known covering radius r_{q^*}, which shows the high potential of VWDT when parallel computing is available. #better, #equal, and #worse are the numbers of instances for which VWDT achieves better, equal, and worse results comparing to the corresponding algorithms, respectively. Note that the exact algorithms do not stop until they find the optimal solutions and prove their optimality, so we may omit f_{best} values for them on small instances. We record the computational time which is less than 0.01 s as "$< \epsilon$". If an algorithm did not report its result on a certain instance, the corresponding item will be marked with a hyphen "-". Row Avg. shows the average computational time taken by each algorithm over all instances of the corresponding dataset. The row p-value is given to verify the statistical significance of the comparison between VWDT and the reference algorithms, which came from the non-parametric Wilcoxon test applied to the best values of VWDT and the reference algorithms. A p-value less than 0.05 indicates a statistically significant difference. In addition, the improved best known results are indicated in bold, while the matched best ones are indicated in italic.

Tables 4 to 6 compare the experimental results of our VWDT with those obtained by ELP (Elloumi et al., 2004), DBR2 (Calik and Tansel, 2013), VNS (Mladenović et al., 2003), PBS (Pullan, 2008), GRASP/PR (Yin et al., 2017), GRASP+PS (Ferone et al., 2017), Gon+, CDSh, CDSh+ (Garcia-Diaz et al., 2019), SAT-sol (Liu et al., 2020) and VWTS (Zhang et al., 2020) on the datasets u1060, rl1323 and u1817 from TSPLIB.

Table 3
Overall results of VWDT on three Well-Known sets including 12 datasets.

Set	Dataset	Count	CPU-q^*	CPU	Hit
ORLIB	pmed	40	$< \epsilon$	$< \epsilon$	100.00%
TSPLIB	sTSP	44	$< \epsilon$	$< \epsilon$	100.00%
	u1060	15	0.02	0.36	100.00%
	rl1323	10	0.11	0.79	100.00%
	u1817	15	0.23	0.92	100.00%
	pcb3038	14	22.30	118.17	100.00%
World TSP	mu1979	8	0.12	0.78	100.00%
	ca4663	8	0.59	4.62	100.00%
	tz6117	8	6.67	43.17	100.00%
	sw24978	4	302.55	1056.66	55.00%
	bm33708	4	519.80	3110.63	46.25%
	ch71009	4	480.02	5754.24	20.00%

As shown in Tables 4, 5, and 6, the proposed VWDT improves the previous best known results obtained by GRASP/PR and PBS on rl1323 with $p = 100$. Apart from the improved solution quality, VWDT outperforms all other algorithms on instances u1060, rl1323, and u1817 in terms of the computational time. In detail, VWDT obtains the optimal solutions in no more than 5 s on all the 40 instances, while PBS, GRASP/PR, and SAT-sol may spend over 1000 s to converge to their best solutions on several hard instances. Furthermore, the proposed algorithm reaches the best results within 0.5 s in half of the instances, whereas PBS, GRASP/PR, and SAT-sol take 100 to 5000 s on most instances. For datasets u1060, r1323 and u1817, VWDT is highly stable for it reaches such results on all 20 independent runs.

The datasets pmed, sTSP, u1060, rl1323, and u1817 have already been closed and VWDT can obtain the optimal solutions on all instances within a short time. Fig. 1 illustrates the overall advantage of VWDT on the closed instances, where we only choose several best-performing algorithms, including GRASP/PB, PBS, SAT-sol, and VWTS for comparison. In these three charts, the height of each bar stands for the number of instances whose computational time is no more than the time specified by the corresponding x value. As we can see, the computing time of VWDT is no more than 0.05 s on the ORLIB instances and the small TSPLIB instances. However, GRASP/PR, PBS, and SAT-sol may spend hundreds of seconds to converge on several hard small instances.

Table 7 reports the computational results on the most challenging dataset pcb3038 from TSPLIB. VNS and GRASP+PS only tackle the instances where $p \geq 50$. Gon+/CDSh and GRASP+PS did not report their computational time. From Table 7, we can observe that, VWTS and VWDT significantly improve the best known results on 10 instances where $p = 50, 100, 150, 200, 250, 300, 350, 400, 450, 500$. Furthermore, VWDT further improves the best results obtained by VWTS on four instances where $p = 50, 100, 150, 200$. Among the improved results, the results for $p = 450, 500$ have been proven to be optimal. For the solution quality, our VWDT algorithm outperforms Gon+/CDSh, VNS, and GPASP+PS algorithms on all instances of pcb3038. Furthermore, we can also observe from Table 3 that the success rate of obtaining the new records is still 100%. In addition to the new upper bounds, the total computational time of the VWDT is much shorter than PBS and GRASP/PR. Compared with VWTS, our VWDT not only improves the quality of the solutions, but also is almost twice as fast as VWTS in terms of the total computational time. The small p-values (<0.05) indicate that there are significant differences between our best results and those of all the reference algorithms (except for VWTS).

Tables 8 and 9 report the computational results obtained by EM(Z*), TSA, GMA (Irawan et al., 2016), GDSh, Gon+ (Garcia-Diaz et al., 2019),

[5] The detailed results are reported in the supplemental materials.

Table 4
Computational results on u1060 dataset from TSPLIB.

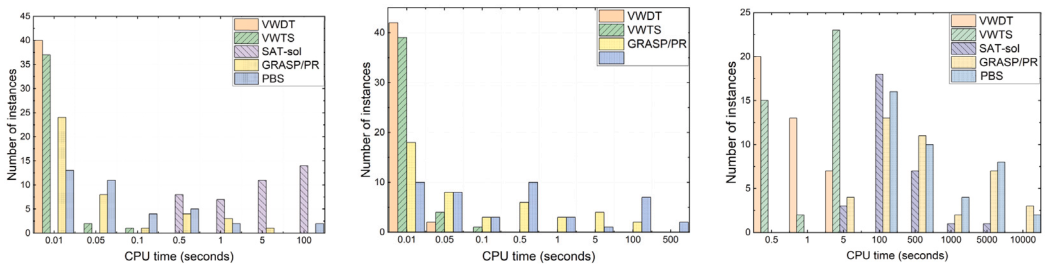
Ins.	n	p	BKS	ELP (Ilhan et al., 2002)		VNS (Mladenović et al., 2003)		CDSh+Garcia-Diaz et al. (2019)	GRASP+PS (Ferone et al., 2017)	PBS (Pullan, 2008)		GRAS/PR (Yin et al., 2017)		SAT-sol (Liu et al., 2020)		VWTS (Zhang et al., 2020)			VWDT		
				f_{best}	CPU	f_{best}	CPU^{0}	f_{best}	f_{best}	f_{best}	CPU	f_{best}	CPU	f_{best}	CPU	f_{best}	CPU-q^*	CPU	f_{best}	CPU-q^*	CPU
u1060	1060	10	2273.08*	*2273*	10.63	2280.09	94.93	2475.60	2301.70	*2273.08*	77.41	*2273.08*	1.31	*2273.08*	1.29	*2273.08*	$< \varepsilon$	0.20	*2273.08*	$< \varepsilon$	0.10
u1060	1060	20	1580.80*	*1581*	557.27	1611.95	20.49	1698.71	1650.34	*1580.80*	369.60	*1580.80*	14.88	*1580.80*	8.03	*1580.80*	0.05	0.65	*1580.80*	0.03	0.45
u1060	1060	30	1207.77*	*1208*	59.78	1220.41	373.46	1299.08	1302.94	*1207.77*	20.65	*1207.77*	3.19	*1207.77*	6.43	*1207.77*	0.09	1.08	*1207.77*	0.02	0.40
u1060	1060	40	1020.56*	*1021*	73.42	1050.45	279.75	1139.49	1118.59	*1020.56*	26.75	*1020.56*	3.26	*1020.56*	9.17	*1020.56*	0.01	0.29	*1020.56*	0.01	0.30
u1060	1060	50	904.92*	*905*	76.83	922.14	477.18	1000.70	950.66	*904.92*	130.66	*904.92*	218.85	*904.92*	14.91	*904.92*	0.10	1.02	*904.92*	0.05	0.81
u1060	1060	60	781.17*	*781*	46.74	806.52	446.89	906.22	860.49	*781.17*	57.80	*781.17*	7.75	*781.17*	9.80	*781.17*	0.01	0.42	*781.17*	0.02	0.45
u1060	1060	70	710.75*	*711*	27.08	721.37	422.73	790.13	790.13	710.76	61.41	*710.75*	116.91	*710.75*	8.46	*710.75*	0.01	0.45	*710.75*	0.01	0.33
u1060	1060	80	652.16*	*652*	12.04	670.53	398.84	721.37	720.94	*652.16*	79.65	*652.16*	316.57	*652.16*	14.41	*652.16*	0.05	0.47	*652.16*	0.03	0.41
u1060	1060	90	607.87*	*608*	7.62	640.23	111.08	671.17	667.55	607.88	35.39	*607.87*	7.09	*607.87*	10.50	*607.87*	0.03	0.40	*607.87*	0.01	0.29
u1060	1060	100	570.01*	*570*	5.82	582.92	430.33	632.88	632.11	*570.01*	9.83	*570.01*	19.04	*570.01*	16.28	*570.01*	0.01	0.38	*570.01*	0.02	0.31
u1060	1060	110	538.84*	*539*	6.02	565.72	186.60	583.32	570.49	*538.84*	90.08	*538.84*	66.46	*538.84*	24.54	*538.84*	0.03	0.39	*538.84*	0.02	0.32
u1060	1060	120	510.27*	*510*	8.83	551.90	218.84	565.71	570.00	510.28	60.33	*510.27*	397.85	*510.27*	13.97	*510.27*	0.05	0.35	*510.27*	0.04	0.30
u1060	1060	130	499.65*	*500*	8.83	500.14	473.65	538.22	538.82	*499.65*	66.53	*499.65*	58.18	*499.65*	11.53	*499.65*	0.02	0.29	*499.65*	0.02	0.26
u1060	1060	140	452.46*	*452*	9.23	500.12	214.06	500.19	500.39	*452.46*	178.50	*452.46*	127.39	*452.46*	4.03	*452.46*	0.04	0.42	*452.46*	0.02	0.37
u1060	1060	150	447.01*	*447*	10.03	453.16	428.16	495.01	499.65	*447.01*	5.94	*447.01*	4.37	*447.01*	4.11	*447.01*	0.01	0.30	*447.01*	0.01	0.25
Avg.					61.34		305.13				84.70		90.87		10.50		0.03	0.47		0.02	0.36
p-value				1.00		6.10E−05		6.10E−05	6.10E−05	0.08		1.00		1.00		1.00					
#better/#equal/#worse				0/15/0		15/0/0		15/0/0	15/0/0	3/12/0		0/15/0		0/15/0		0/15/0					

Table 5
Computational results on rl1323 dataset from TSPLIB.

Ins.	n	p	BKS	ELP (Ilhan et al., 2002)		GRASP+PS (Ferone et al., 2017)	PBS (Pullan, 2008)		GRAS/PR (Yin et al., 2017)		VWTS (Zhang et al., 2020)			VWDT		
				f_{best}	CPU	f_{best}	f_{best}	CPU	f_{best}	CPU	f_{best}	CPU-q^*	CPU	f_{best}	CPU-q^*	CPU
rl1323	1323	10	3077.30*	*3077*	276.81	3110.57	*3077.30*	2667.90	*3077.30*	38.02	*3077.30*	0.02	0.62	*3077.30*	0.01	0.24
rl1323	1323	20	2016.40*	*2016*	96.28	2090.87	*2016.40*	339.59	*2016.40*	104.89	*2016.40*	0.15	2.00	*2016.40*	0.08	0.56
rl1323	1323	30	1631.50*	*1632*	180.53	1730.78	*1631.50*	672.68	*1631.50*	169.47	*1631.50*	0.19	2.22	*1631.50*	0.18	1.04
rl1323	1323	40	1352.36*	*1352*	601.77	1479.24	*1352.36*	163.66	*1352.36*	21.90	*1352.36*	0.42	2.66	*1352.36*	0.06	1.02
rl1323	1323	50	1187.27*	*1187*	1721.06	1300.00	*1187.27*	347.16	*1187.27*	119.63	*1187.27*	0.10	1.59	*1187.27*	0.04	0.69
rl1323	1323	60	1063.01*	*1063*	1829.38	1181.30	*1063.01*	4587.41	*1063.01*	4190.92	*1063.01*	0.40	1.83	*1063.01*	0.28	0.81
rl1323	1323	70	971.93*	*972*	349.03	1076.20	*971.93*	4162.63	*971.93*	6287.04	*971.93*	0.26	1.94	*971.93*	0.26	1.10
rl1323	1323	80	895.06*	*895*	84.25	988.87	*895.06*	4922.63	*895.06*	5265.81	*895.06*	0.06	1.69	*895.06*	0.06	1.03
rl1323	1323	90	832.00*	*832*	24.07	935.02	*832.00*	521.18	*832.00*	776.23	*832.00*	0.04	1.73	*832.00*	0.03	0.58
rl1323	1323	100	787.10*	*787*	24.07	886.85	789.70	1031.55	789.70	2010.67	*787.10*	0.31	1.69	*787.10*	0.08	0.81
Avg.					545.60			1941.64		1898.46		0.20	1.80		0.11	0.79
p-value				1.00		1.95E−03	0.32		0.32		1.00					
#better/#equal/#worse				0/10/0		10/0/0	1/9/0		1/9/0		0/10/0					

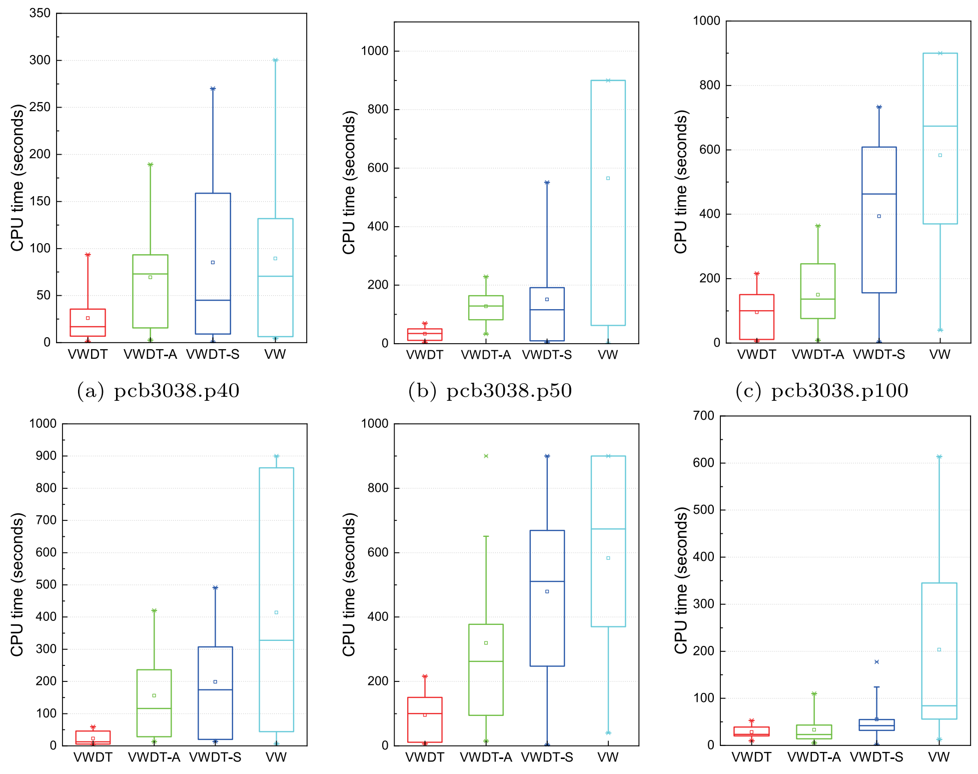
Table 6
Computational results on u1817 dataset from TSPLIB.

Ins.	n	p	BKS	ELP (Ilhan et al., 2002)		DBR2(Calik and Tansel, 2013)		CDSh+Garcia-Diaz et al. (2019)	GRASP+PS (Ferone et al., 2017)	PBS (Pullan, 2008)		GRASP/PR (Yin et al., 2017)		SAT-sol (Liu et al., 2020)		VWTS (Zhang et al., 2020)			VWDT		
				f_{best}	CPU	f_{best}	CPU^{0}	f_{best}	f_{best}	f_{best}	CPU	f_{best}	CPU	f_{best}	CPU	f_{best}	CPU-q^*	CPU	f_{best}	CPU-q^*	CPU
u1817	1817	10	457.91*	*458*	541.59	*458*	4.00	475.54	466.96	*457.91*	2979.81	*457.91*	604.53	*457.91*	42.61	*457.91*	1.85	1.78	*457.91*	0.39	0.83
u1817	1817	20	309.01*	310	986.90	*309*	278.49	338.89	330.20	*309.01*	5740.91	*309.01*	4068.06	*309.01*	80.93	*309.01*	2.92	8.56	*309.01*	0.82	2.10
u1817	1817	30	240.99*	250	3309.73	*241*	344.59	283.98	265.19	*240.99*	899.84	*240.99*	1239.97	*240.99*	123.61	*240.99*	1.52	5.23	*240.99*	0.45	2.32
u1817	1817	40	209.45*	210	1287.79	*209*	1221.86	236.22	232.25	209.46	108.56	*209.45*	308.29	*209.45*	47.09	*209.45*	0.91	1.41	*209.45*	0.09	0.77
u1817	1817	50	184.91*	187	1973.81	*185*	330.09	209.43	204.79	*184.91*	632.72	*184.91*	471.94	*184.91*	173.81	*184.91*	0.08	1.20	*184.91*	0.08	0.75
u1817	1817	60	162.64*	*163*	252.74	*163*	17.52	193.42	184.91	162.65	469.28	*162.64*	469.43	*162.64*	134.17	*162.64*	0.07	0.93	*162.64*	0.03	0.60
u1817	1817	70	148.11*	*148*	84.25	*148*	6.04	179.59	170.39	*148.11*	107.50	*148.11*	19.66	*148.11*	46.17	*148.11*	0.05	0.45	*148.11*	0.04	0.38
u1817	1817	80	136.77*	*137*	228.67	*137*	35.12	152.41	154.50	136.80	71.46	136.80	12.42	*136.77*	2313.89	*136.77*	3.00	1.66	*136.77*	0.44	0.63
u1817	1817	90	129.51*	*130*	1444.65	*129(?)*	7519.04	148.09	148.11	129.54	1660.96	*129.51*	3859.05	*129.51*	370.95	*129.51*	0.55	0.59	*129.51*	0.06	0.34
u1817	1817	100	126.99*	*127*	60.18	*127*	10.22	136.77	136.79	127.01	82.05	*126.99*	2.35	*126.99*	124.24	*126.99*	0.02	0.27	*126.99*	0.02	0.23
u1817	1817	110	109.25*	*109*	84.25	110	5.32	129.50	–	*109.25*	7719.04	*109.25*	6954.89	*109.25*	84.25	*109.25*	0.27	1.70	*109.25*	0.13	0.68
u1817	1817	120	107.76*	*108*	24.07	*107(?)*	3.99	126.99	–	107.78	44.89	*107.76*	5.25	*107.76*	54.48	*107.76*	0.02	0.25	*107.76*	0.01	0.21
u1817	1817	130	104.73*	108	746.19	*105*	335.03	113.59	–	107.75	6.28	107.75	7.04	*104.73*	941.17	*104.73*	1.20	9.17	*104.73*	0.80	2.63
u1817	1817	140	101.60*	105	806.37	*102*	4.62	107.79	–	101.61	2773.94	*101.60*	30.95	*101.60*	385.56	*101.60*	0.39	0.60	*101.60*	0.12	0.40
u1817	1817	150	91.60*	94	1131.33	*92*	6.61	107.75	–	101.60	175.99	92.44	1236.55	*91.60*	227.99	*91.60*	0.07	1.41	*91.60*	0.05	0.94
Avg.					864.17		674.84				1564.88		1286.03				0.86	2.35		0.23	0.92
p-value				1.73E−02		0.32		6.10E−05	5.07E−03	7.57E−03		0.11		1.00		1.00					
#better/#equal/#worse				7/8/0		1/12/0		15/0/0	10/0/0	9/6/0		3/12/0		0/15/0		0/15/0					



(a) pmed (b) sTSP (c) u1060, rl1323, and u1817

Fig. 1. Distribution of computational time on closed instances.



(a) pcb3038.p40 (b) pcb3038.p50 (c) pcb3038.p100
(d) pcb3038.p150 (e) pcb3038.p200 (f) pcb3038.p250

Fig. 2. Comparison on different tabu strategies.

Table 7
Computational results on pcb3038 dataset from TSPLIB.

Ins.	n	p	BKS	VNS (Mladenović et al., 2003)		DBR2(Calik and Tansel, 2013)		Con+/CDSh (Garcia-Diaz et al., 2019)	GRASP+PS (Ferone et al., 2017)	PBS (Pullan, 2008)		GRASP/PR (Yin et al., 2017)		VWTS (Zhang et al., 2020)			VWDT		
				f_{best}	CPU^0	f_{best}	CPU^0	f_{best}	f_{best}	f_{best}	CPU	f_{best}	CPU	f_{best}	CPU-q^*	CPU	f_{best}	CPU-q^*	CPU
pcb3038	3038	10	728.54*	–	–	*729*	176.23	806.98	–	*728.54*	3035.18	*728.54*	240.56	*728.54*	0.53	23.37	*728.54*	0.51	18.91
pcb3038	3038	20	493.04*	–	–	*493*	22740.76	582.60	–	*493.04*	544.83	*493.04*	1051.99	*493.04*	1.74	38.28	*493.04*	1.41	23.67
pcb3038	3038	30	393.50	–	–	397	76923.67	502.31	–	*393.50*	245.82	*393.50*	644.34	*393.50*	1.23	69.22	*393.50*	0.89	23.92
pcb3038	3038	40	336.42	–	–	337	72364.56	411.91	–	*336.42*	427.64	*336.42*	210.14	*336.42*	79.81	264.98	*336.42*	18.23	89.40
pcb3038	3038	50	297.83	317.00	578.81	300	91029.77	364.11	534.48	298.20	410.04	298.10	4686.77	298.04	78.34	307.51	***297.83***	33.85	201.46
pcb3038	3038	100	206.31	220.06	570.60	209	27292.39	327.39	399.49	206.63	317.12	207.06	6678.44	206.60	97.85	683.89	***206.31***	95.79	403.21
pcb3038	3038	150	164.40	174.83	52.99	–	–	306.91	331.62	164.77	239.32	165.00	5653.32	164.55	88.55	705.74	***164.40***	22.92	325.11
pcb3038	3038	200	140.06	157.88	747.00	141	33929.91	280.14	301.01	140.90	504.09	140.62	5021.13	140.09	59.09	360.98	***140.06***	101.48	371.67
pcb3038	3038	250	122.25	140.98	103.72	–	–	257.80	292.48	122.78	405.61	122.78	1985.10	*122.25*	24.37	205.55	*122.25*	15.56	90.85
pcb3038	3038	300	115.00	123.33	786.96	*115*	6185.96	243.50	261.28	115.25	301.20	115.73	3671.32	*115.00*	29.73	133.26	*115.00*	18.43	50.54
pcb3038	3038	350	104.68	118.02	264.60	–	–	233.34	258.82	104.81	451.68	104.81	3926.83	*104.68*	1.64	49.33	*104.68*	0.97	18.85
pcb3038	3038	400	96.88	107.65	904.88	97	11.48	212.04	249.78	97.51	347.83	97.80	6956.46	*96.88*	2.89	55.58	*96.88*	1.08	15.65
pcb3038	3038	450	88.55*	101.51	675.55	–	–	205.00	214.97	88.90	373.39	89.56	6161.75	*88.55*	0.51	49.46	*88.55*	0.49	16.58
pcb3038	3038	500	84.58*	94.37	937.70	*85*	5.23	196.65	209.35	85.00	244.25	85.09	3015.45	*84.58*	0.76	33.60	*84.58*	0.55	12.99
Avg.					562.28		33066.00				560.57		3564.54		33.36	212.91		22.30	118.77
p-value				1.95E−03		0.04		1.22E−04	1.95E−03	5.06E−03		5.06E−03		0.06					
#better/#equal/#worse				10/0/0		5/5/0		14/0/0	10/0/0	10/4/0		10/4/0		4/10/0					

Table 8
Computational results on mu1979, ca4663, and tz6117 datasets from world TSP.

Ins.	n	p	BKS	EM (Z*)		TSA (Irawan et al., 2016)		GMA (Irawan et al., 2016)		GDSh (Garcia-Diaz et al., 2019)	VWTS (Zhang et al., 2020)			VWDT		
				f_{best}	CPU	f_{best}	CPU	f_{best}	CPU	f_{best}	f_{best}	CPU-q^*	CPU	f_{best}	CPU-q^*	CPU
mu1979	1979	5	1876.83*	*1876.83*	54.76	*1876.83*	37.00	*1876.83*	79.56	–	*1876.83*	0.07	0.14	*1876.83*	0.07	0.11
mu1979	1979	10	1160.70*	*1160.70*	42.61	*1160.70*	30.71	*1160.70*	60.40	1160.79	*1160.70*	0.17	0.37	*1160.70*	0.05	0.10
mu1979	1979	15	867.52*	*867.52*	29.73	*867.52*	39.47	*867.52*	64.18	–	*867.52*	0.20	0.93	*867.52*	0.14	0.54
mu1979	1979	20	750.53*	*750.53*	31.56	*750.53*	71.10	*750.53*	67.29	768.29	*750.53*	0.26	1.08	*750.53*	0.03	0.66
mu1979	1979	25	638.79*	*638.79*	24.59	*638.79*	82.24	*638.79*	63.09	–	*638.79*	0.22	0.97	*638.79*	0.17	0.68
mu1979	1979	50	380.90*	*380.90*	24.72	382.70	129.24	*380.90*	129.38	400.34	*380.90*	0.20	1.06	*380.90*	0.07	0.49
mu1979	1979	75	284.80*	*284.80*	22.42	286.73	149.67	*284.80*	105.12	–	*284.80*	0.29	1.36	*284.80*	0.01	0.37
mu1979	1979	100	220.32*	*220.32*	36.58	226.50	184.22	222.09	154.56	235.08	*220.32*	0.95	6.87	*220.32*	0.44	3.20
ca4663	4663	5	16836.61*	*16836.61*	836.47	*16836.61*	126.46	*16836.61*	315.08	–	*16836.61*	1.62	7.57	*16836.61*	0.57	4.29
ca4663	4663	10	10498.81*	*10498.81*	511.52	*10498.81*	131.28	*10498.81*	292.41	12184.71	*10498.81*	1.16	18.92	*10498.81*	0.47	2.24
ca4663	4663	15	8295.93*	*8295.93*	377.92	*8295.93*	230.13	*8295.93*	372.31	–	*8295.93*	0.81	4.56	*8295.93*	0.26	2.10
ca4663	4663	20	7023.87*	*7023.87*	338.53	*7023.87*	370.11	*7023.87*	484.33	8414.57	*7023.87*	0.73	12.04	*7023.87*	0.33	7.70
ca4663	4663	25	5965.76*	*5965.76*	331.87	5966.51	368.82	*5965.76*	432.18	–	*5965.76*	3.72	37.87	*5965.76*	2.20	9.15
ca4663	4663	50	3955.06*	*3955.06*	412.34	3955.50	347.18	*3955.06*	338.25	4663.15	*3955.06*	0.56	6.75	*3955.06*	0.21	2.41
ca4663	4663	75	3069.32*	*3069.32*	466.88	3072.09	330.17	3072.09	283.01	–	*3069.32*	0.68	10.70	*3069.32*	0.35	3.53
ca4663	4663	100	2543.89*	*2543.89*	382.77	2545.53	383.33	2545.53	280.58	2874.45	*2543.89*	0.64	9.80	*2543.89*	0.31	5.52
tz6117	6117	5	2917.86*	*2917.86*	3021.76	*2917.86*	551.36	*2917.86*	1343.20	–	*2917.86*	1.92	28.64	*2917.86*	1.09	19.31
tz6117	6117	10	1902.12*	*1902.12*	9220.21	*1902.12*	371.38	*1902.12*	895.64	–	*1902.12*	3.94	41.14	*1902.12*	1.34	20.31
tz6117	6117	15	1527.98*	*1527.98*	29102.14	1529.38	737.43	1528.46	1158.97	–	*1527.98*	8.45	68.58	*1527.98*	2.48	22.41
tz611	6117	20	1278.30*	*1278.30*	20584.44	1278.62	1136.69	1279.30	1417.64	–	*1278.30*	4.07	32.00	*1278.30*	1.26	10.52
tz6117	6117	25	1152.05*	*1152.05*	294422.79	1153.16	947.31	1153.16	1168.50	–	*1152.05*	48.80	117.67	*1152.05*	16.12	62.53
tz6117	6117	50	776.21	N/A	N/A	806.23	646.19	806.23	916.10	–	***776.21***	9.16	48.13	***776.21***	2.81	27.08
tz6117	6117	75	622.72	N/A	N/A	663.53	594.17	663.74	729.12	–	***622.72***	31.89	67.43	***622.72***	11.32	38.59
tz6117	6117	100	529.67	N/A	N/A	579.75	627.09	566.18	696.23	–	***529.67***	40.95	114.67	***529.67***	16.98	76.54
Avg.					17156.03		359.28		493.63						2.46	13.35
p-value				1.00		6.50E−04		3.34E−03		7.81E−03	1.00					
#better/#equal/#worse				0/21/0		13/11/0		9/15/0		8/0/0	0/24/0					

Table 9
Computational results on sw24978, bm33708, and ch71009 datasets from world TSP.

Ins.	n	p	BKS	TSA (Irawan et al., 2016)		GMA (Irawan et al., 2016)		Gon+(Garcia-Diaz et al., 2019)	VWTS (Zhang et al., 2020)					VWDT				
				f_{best}	CPU	f_{best}	CPU	f_{best}	f_{best}	f_{avg}	CPU-q^*	CPU	Hit	f_{best}	f_{avg}	CPU-q^*	CPU	Hit
sw24978	24978	25	1238.39	1335.99	8.76	1329.37	1319.13	1550.35	***1238.39***	1241.8	201.34	3623.82	9/20	***1238.39***	***1238.39***	78.84	603.13	20/20
sw24978	24978	50	846.22	929.61	504.10	925.71	1520.68	1088.45	847.11	848.88	360.43	2559.96	3/20	***846.22***	***846.23***	411.20	1293.70	14/20
sw24978	24978	75	680.89	759.16	745.61	759.02	1095.73	862.32	681.02	683.38	317.23	2344.5	5/20	***680.89***	682.08	459.22	1287.13	3/20
sw24978	24978	100	574.70	686.48	528.98	685.77	909.84	741.89	577.17	577.89	640.53	2470.29	2/20	***574.7***	575.02	260.95	1042.67	7/20
bm33708	33708	25	1114.18	1183.80	588.19	1183.80	851.62	1414.40	1117.79	1117.95	301.44	3344.73	8/20	***1114.18***	***1114.18***	347.94	1717.97	18/20
bm33708	33708	50	743.30	823.27	944.72	823.78	1087.10	851.31	744.05	744.52	521.32	3081.34	3/20	***743.3***	744.29	640.80	3369.85	10/20
bm33708	33708	75	584.29	686.75	624.19	683.94	1121.05	766.84	585.47	587.4	425.42	2083.8	2/20	***584.29***	587.38	312.64	2396.90	7/20
bm33708	33708	100	502.49	594.73	447.82	593.48	1848.85	649.14	503.87	504.8	592.24	2369.18	3/20	***502.49***	504.36	777.81	4957.81	2/20
ch71009	71009	25	4205.77	4430.15	6358.05	4428.72	7649.10	5608.06	4211.82	4213.87	546.52	3290.74	3/20	***4205.77***	4208.12	392.31	3452.54	7/20
ch71009	71009	50	2928.22	3109.67	6167.86	3107.56	7642.38	3641.98	2935.54	2938.01	679.53	6978.92	2/20	***2928.22***	2931.06	649.64	6286.64	3/20
ch71009	71009	75	2319.78	2554.32	6115.02	2554.63	7630.42	2985.92	2326.84	2330.31	643.25	8924.42	1/20	***2319.78***	2323.79	461.40	5865.34	3/20
ch71009	71009	100	2013.67	2170.69	6083.75	2168.97	6913.58	2524.63	2029.79	2048.62	703.53	16795.49	1/20	***2013.67***	2031.97	416.74	7412.43	2/20
Avg.					2426.42		3299.12									434.12	3307.17	
p-value				4.88E−04		4.88E−04		4.88E−04	3.35E−03	4.88E−04								
#better/#equal/#worse				12/0/0		12/0/0		12/0/0	11/1/0	12/0/0								

Table 10
Overall computational results on all massive TSP datasets.

Dataset	Count	CIK (Contardo et al., 2019)			fCLH (Gaar and Sinnl, 2022)			VWDT		
		#best	#better	CPU	#best	#better	CPU	#best	CPU-q^*	CPU
P2	42	*42*	0	23.80	*42*	0	3.27	*42*	12.85	204.25
P3	42	*42*	0	43.76	*42*	0	8.76	*42*	18.91	392.59
P5	42	*42*	0	171.88	*42*	0	62.80	*42*	48.59	512.64
P10	42	*42*	0	1642.73	29	13	771.87	***42***	52.58	587.45
P15	42	36	7	17174.12	17	25	1194.03	***42***	46.50	650.12
P20	42	29	13	33031.00	10	32	1391.29	***42***	53.77	657.98
P25	42	21	21	46943.70	8	34	1469.47	***42***	58.17	681.28
P30	42	18	24	53204.92	8	34	1474.28	***42***	60.58	825.52

Table 11
Computational results on the improved massive TSP instances.

Ins.	p	LB	CIK (Contardo et al., 2019)		fCLH (Gaar and Sinnl, 2022)		VWDT		
			f_{best}	CPU	f_{best}	CPU	f_{best}	CPU-q^*	CPU
fyg28534	15	142	151	TL1	168	TL2	**144**	13.41	229.24
pla33810	15	110019	126933	TL1	137325	TL2	**111811**	74.24	1222.89
bby34656	15	155	164	TL1	189	TL2	**156**	214.63	1080.14
pba38478	15	160	163	TL1	208	TL2	**161**	38.53	1604.11
pla85900	15	142938	159484	TL1	185853	TL2	**147969**	230.64	4021.31
usa115475	15	5288	5419	TL1	6259	TL2	**5361**	105.86	4325.54
ara238025	15	441	458	TL1	511	TL2	**455**	450.23	5405.43
rl11849	20	2119	2273	TL1	2629	TL2	**2139**	63.73	759.07
usa13509	20	44740	46719	TL1	56610	TL2	**44744**	97.23	820.30
d15112	20	2581	2717	TL1	3359	TL2	**2615**	53.36	455.12
d18512	20	912	969	TL1	1218	TL2	**925**	35.45	526.78
fyg28534	20	118	130	TL1	154	TL2	**122**	87.42	2027.45
pla33810	20	91302	106076	TL1	118855	TL2	**94884**	58.65	2785.04
bby34656	20	128	138	TL1	159	TL2	**132**	77.15	822.20
pba38478	20	136	148	TL1	175	TL2	**140**	234.43	1017.09
ch71009	20	4798	5250	TL1	5868	TL2	**4842**	206.77	2611.83
pla85900	20	119643	136984	TL1	163300	TL2	**127046**	139.77	2910.99
sra104815	20	232	236	TL1	294	TL2	**233**	138.67	1422.17
usa115475	20	4453	4747	TL1	5855	TL2	**4691**	108.26	2053.14
ara238025	20	372	407	TL1	466	TL2	**396**	441.54	4957.25
pcb3038	25	433	470	TL1	545	TL2	**438**	2.55	10.42
rl5915	25	1823	1916	TL1	2201	TL2	**1825**	6.39	43.83
tz6117	25	1152	1258	TL1	1426	TL2	***1152***	7.22	28.66
ei8246	25	429	461	TL1	532	TL2	**433**	5.95	19.86
fi10639	25	1103	1173	TL1	1400	TL2	**1106**	39.25	272.97
rl11849	25	1838	2099	TL1	2506	TL2	**1864**	29.02	390.99
usa13509	25	38150	40578	TL1	46954	TL2	**38238**	54.64	648.30
brd14051	25	703	737	TL1	843	TL2	**709**	83.72	385.89
d15112	25	2233	2447	TL1	2877	TL2	**2299**	69.05	670.41
d18512	25	795	881	TL1	1029	TL2	**817**	34.33	424.39
sw24978	25	1233	1285	TL1	1567	TL2	**1238**	60.38	503.61
fyg28534	25	102	112	TL1	133	TL2	**107**	58.54	688.61
bm33708	25	1106	1140	TL1	1358	TL2	**1114**	207.92	926.10
pla33810	25	81800	88861	TL1	108074	TL2	**84095**	283.15	2491.55
bby34656	25	112	128	TL1	144	TL2	**117**	63.53	1643.76
pba38478	25	117	134	TL1	160	TL2	**123**	101.41	1379.08
ch71009	25	4145	4321	TL1	5491	TL2	**4206**	258.95	2008.95
pla85900	25	107527	124693	TL1	145695	TL2	**117019**	21.32	2911.59
sra104815	25	206	216	TL1	241	TL2	**211**	202.18	4046.17
usa115475	25	3808	4453	TL1	5347	TL2	**4262**	143.53	2243.14
ara238025	25	319	357	TL1	393	TL2	**340**	404.60	5064.79
pr2392	30	1379	1471	TL1	1765	TL2	**1387**	0.59	1.33
pcb3038	30	386	412	TL1	508	TL2	**394**	1.23	2.50
rl5915	30	1624	1853	TL1	2210	TL2	**1666**	2.31	65.57
rl5934	30	1658	1812	TL1	2116	TL2	**1665**	3.06	57.41
tz6117	30	1025	1142	TL1	1304	TL2	**1027**	3.84	22.35
ei8246	30	386	412	TL1	508	TL2	**393**	3.26	55.23
fi10639	30	974	1017	TL1	1251	TL2	**984**	17.67	148.76
rl11849	30	1649	1855	TL1	2255	TL2	**1716**	106.13	969.27
usa13509	30	34471	37306	TL1	45591	TL2	**35283**	67.76	763.77
brd14051	30	620	668	TL1	812	TL2	**632**	21.31	253.73
mo14185	30	732	767	TL1	936	TL2	**737**	73.93	355.40
d15112	30	2009	2254	TL1	2662	TL2	**2084**	201.46	982.72
d18512	30	712	786	TL1	963	TL2	**735**	104.22	740.40
sw24978	30	1096	1215	TL1	1514	TL2	**1114**	141.41	950.27
fyg28534	30	93	108	TL1	122	TL2	**97**	120.13	501.66
bm33708	30	968	1065	TL1	1221	TL2	**982**	56.86	1008.13
pla33810	30	71480	83187	TL1	95203	TL2	**76000**	440.41	3910.27
bby34656	30	102	133	TL1	137	TL2	**106**	37.86	1902.96
pba38478	30	107	125	TL1	149	TL2	**112**	54.62	2411.74
ch71009	30	3724	4098	TL1	4947	TL2	**3897**	81.69	3077.36
pla85900	30	93871	114244	TL1	127681	TL2	**100809**	31.15	3605.74
sra104815	30	186	206	TL1	216	TL2	**187**	290.71	2718.68
usa115475	30	3391	3874	TL1	4770	TL2	**3727**	141.43	3430.31
ara238025	30	288	368	TL1	386	TL2	**322**	430.53	5922.92
Avg.				TL1		TL2		114.48	1564.90

Table 12
Comparison of VWDT, VWDT-A, VWDT-S, and VW on instances pcb3038 with 10 – 500 centers.

Ins.	p	f_{best}	VWDT			VWDT-A			VWDT-S			VW		
			T_{best}	T_{avg}	Hit	T_{best}	T_{avg}	Hit	T_{best}	T_{avg}	Hit	T_{best}	T_{avg}	Hit
pcb3038	10	728.54	0.06	0.51	20/20	0.96	3.59	20/20	0.15	1.75	20/20	0.08	1.40	20/20
pcb3038	20	493.04	0.22	1.41	20/20	1.09	9.18	20/20	0.06	7.05	20/20	0.16	37.73	20/20
pcb3038	30	393.50	0.18	0.89	20/20	1.16	4.39	20/20	0.31	1.70	20/20	0.21	2.84	20/20
pcb3038	40	336.42	1.16	18.23	20/20	2.66	69.35	20/20	0.41	101.77	20/20	4.05	98.44	20/20
pcb3038	50	297.83	4.18	33.85	20/20	32.62	127.60	20/20	4.38	151.37	20/20	40.01	565.25	12/20
pcb3038	100	206.31	5.82	95.79	20/20	8.50	149.90	18/20	2.58	432.34	17/20	0.41	582.88	9/20
pcb3038	150	164.40	4.87	22.92	20/20	3.44	49.15	20/20	13.66	198.61	20/20	5.94	413.96	16/20
pcb3038	200	140.06	10.88	101.48	20/20	13.14	236.24	16/20	77.09	281.27	17/20	550.70	865.07	2/20
pcb3038	250	122.25	9.77	15.56	20/20	5.23	17.77	20/20	1.35	47.34	20/20	12.80	122.25	20/20
pcb3038	300	115.00	0.55	18.43	20/20	0.23	14.02	20/20	3.71	32.86	20/20	0.67	21.99	20/20
pcb3038	350	104.68	0.22	0.97	20/20	0.17	1.02	20/20	0.53	1.11	20/20	0.19	1.53	20/20
pcb3038	400	96.88	0.62	1.08	20/20	1.86	2.67	20/20	0.63	1.61	20/20	0.13	1.70	20/20
pcb3038	450	88.55	0.40	0.49	20/20	1.46	2.05	20/20	0.42	1.31	20/20	0.78	3.90	20/20
pcb3038	500	84.58	0.27	0.55	20/20	0.15	1.20	20/20	0.15	0.65	20/20	0.08	0.79	20/20
Avg.			2.80	22.30	100%	5.19	49.15	97.86%	7.53	90.05	97.86%	44.01	194.27	85.36%

VWTS (Zhang et al., 2020), and our proposed VWDT on 36 instances from World TSP. From Tables 8 and 9, we can observe that for the 15 open instances (tz6117 with $p = 50, 75, 100$, sw24978, bm33708 and ch71009) whose optimality is unknown, VWDT improves the best results on the remaining 11 instances (sw24978 with $p = 50, 75, 100$, bm33708 and ch71009) on the basis that VWTS has already improved the best results for 4 instances (tz6117 with $p = 50, 75, 100$, sw24978 with $p = 25$), and matches the optimal solutions for the remaining ones. For Table 9, the small p-values (<0.05) indicate that there are significant differences between our best results and those of all the reference algorithms. It is noteworthy that VWDT keeps a 100% success rate on these instances with less than 10,000 vertices (mu1979, ca4663, tz6117). Overall, VWDT outperforms TSA, GMA, Gon+, and VWTS algorithms in terms of both the solution quality and computational efficiency.

5.4.2. *Massive TSP set*

Table 10 shows the overall results on the massive set obtained by CIK (Contardo et al., 2019), fCLH (Gaar and Sinnl, 2022), and our VWDT. Column Count shows the number of instances in each dataset. Column #best presents the number of instances where the corresponding algorithms match the best known results among all the algorithms. It can be observed that VWDT obtains the best known results for all the instances, while no reference algorithm can obtain such as overall outcome. Compared to fCLH, our VWDT obtains 138 better, 198 equal, and no worse solutions. Compared with the best performing algorithm CIK, VWDT obtains 65 better, 272 equal, and no worse solutions. When the number of centers is small ($p \leq 5$), our VWDT takes slightly longer time than CIK and fCLH to obtain the best known results. However, as the number of centers increases ($p > 5$), the advantage of VWDT over CIK and fCLH becomes obvious in terms of solution quality and computational efficiency. Table 11 shows the details of the 65 improved instances.[6] TL1 and TL2 are the running time limits of CIK and fCLH, which are 24 h and 1800 s, respectively. Column LB gives the lower bound of each instance obtained by exact algorithms in the literature. Specifically, for instance tz6117 $p = 25$, the upper bound obtained by VWDT is equal to the lower bound, i.e., VWDT closes this instance.

In sum, these statistics show that our VWDT is highly effective and efficient for solving large scale p-center problem instances. Especially, when the number of centers is large, its performance for obtaining high-quality upper bounds is much better than the exact algorithms.

[6] We report the detailed results of all instances in the supplemental materials.

6. Analysis and discussions

6.1. *Importance of hybrid double-tabu strategy*

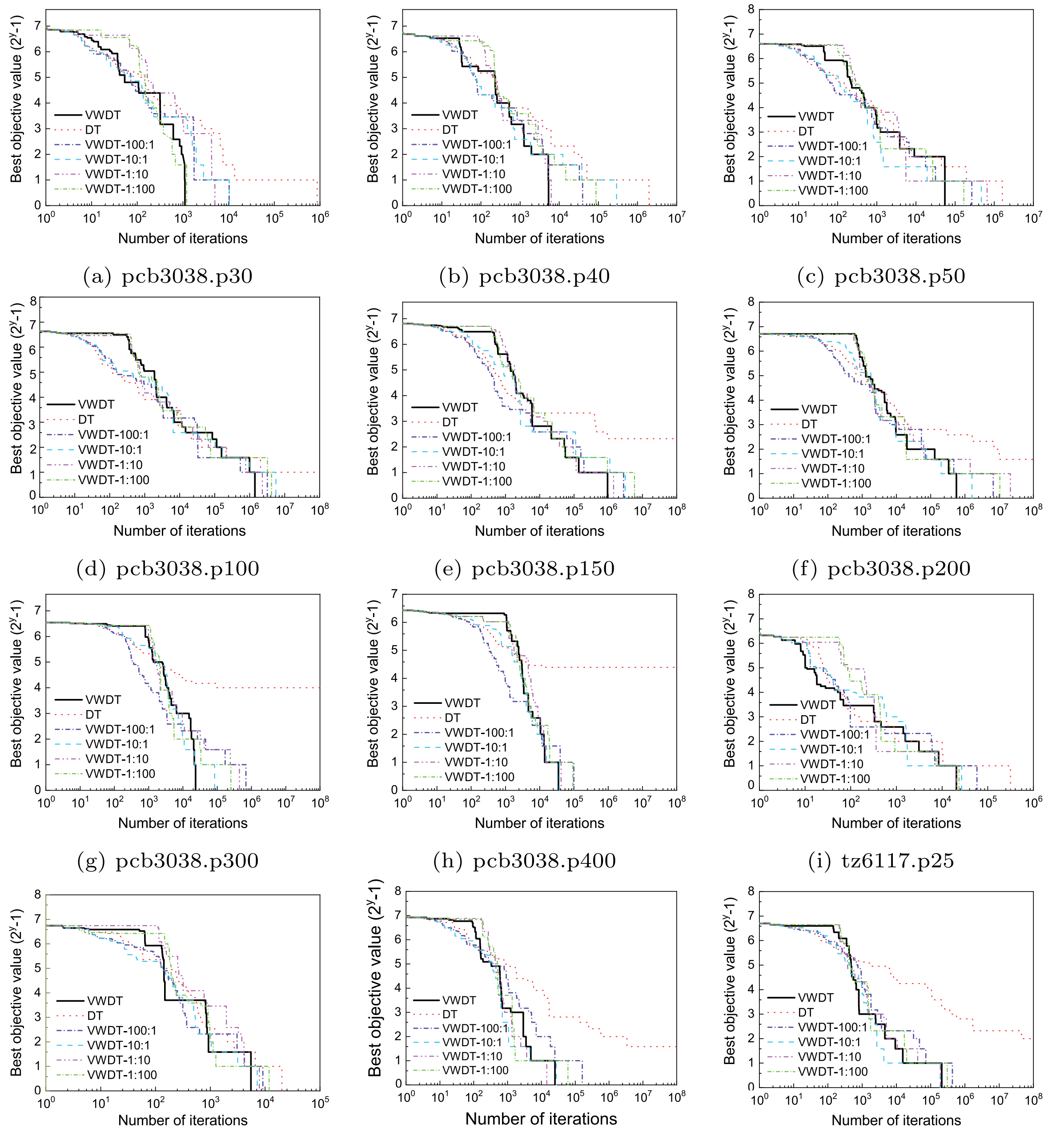
In our VWDT, the hybrid double-tabu strategy plays an important role for stably producing high-quality results. In order to verify the effectiveness of the hybrid double-tabu strategy, we conduct experiments on the hard large dataset pcb3038 from TSPLIB to compare the original version of VWDT with three simplified versions named VW, VWDT-S, and VWDT-A, respectively. These four versions share the same components except for the configuration of the tabu strategies. VW uses a local search without any tabu strategy. VWDT-S only employs the solution-based tabu strategy, and VWDT-A only enables the attribute-based tabu strategy. For each algorithm, we perform 20 independent runs on the pcb3038 dataset. For each instance, the time limit for each subproblem corresponding to each covering radius is 900 s.

Table 12 reports the experimental results produced by VWDT, VW, VWDT-S, and VWDT-A on pcb3038 dataset. Columns T_{best} and T_{avg} present the best and average CPU time to match the best known result f_{best}, respectively. Column Hit shows the success rate for reaching the best known result f_{best} under the given time limits. Furthermore, Fig. 2 illustrates the distributions of the computational time consumed by VWDT, VW, VWDT-A, and VWDT-S on six representative instances (pcb3038 with p = 40, 50, 100, 150, 200, 250) in a box-and-whisker plot.

Table 12 and Fig. 2 demonstrate that the hybrid double-tabu strategy outperforms other three strategies. In detail, Table 12 shows that VWDT-A and VWDT-S fail to always match the best known solutions on 2 instances (pcb3038 with $p = 100, 200$), and VW fails to do so on 4 ones (pcb3038 with $p = 50, 100, 150, 200$). Moreover, for the instances with $p = 100$ and $p = 200$, the hit rates of VWDT-A and VWDT-S are not 100%, while the hit rates of VW are even less than 50%. Thus, VWDT is more stable than VW, VWDT-A and VWDT-S for it has a higher hit rate and a better average deviation from the best found solution. As shown in Fig. 2, the hybrid double-tabu strategy obviously outperforms other variants on all 6 instances in terms of computational efficiency.

6.2. *Effectiveness of the vertex weighting technique*

As the tabu search proceeds, more solutions are recorded in the solution-based tabu list, which may narrow the feasible solution space into disconnected areas and may hinder the search for trajectory-based metaheuristics. In contrast, the vertex weighting technique helps the search to jump out of the local optima by altering the objective function, which plays a significant role in the proposed VWDT algorithm.



(a) pcb3038.p30 (b) pcb3038.p40 (c) pcb3038.p50

(d) pcb3038.p100 (e) pcb3038.p150 (f) pcb3038.p200

(g) pcb3038.p300 (h) pcb3038.p400 (i) tz6117.p25

(j) tz6117.p50 (k) tz6117.p75 (l) tz6117.p100

Fig. 3. Effectiveness of vertex weighting technique and performance of different ratios of initial weight to increment.

In order to justify the importance of the vertex weighting technique, we conduct experiments on 12 hard instances to compare the performance of VWDT with its simplified versions. Specifically, we disable the vertex weighting technique to obtain a new DT algorithm, and then compare it with the original VWDT on large instances pcb3038 with $p = 30, 40, 50, 100, 150, 200, 300, 400$ and tz6117 with $p = 25, 50, 75, 100$. DT works almost the same as VWDT, except that the vertex weights of DT are always one during the entire search procedure, i.e., lines 11–13 in Algorithm 1 are disabled.

In addition, we also analyze how different vertex weights updating schemes affect the effectiveness of the VWDT algorithm. The ratio of the initial weight to the increment controls the aggressiveness of the search. If the increment is too large, the penalty for the uncovered vertices will grow steeply, which makes the uncovered vertices be quickly covered in subsequent search. If the increment is too small, it may take a long time to reshape the landscape of the solution space and get out of the basin around the local optima. In order to find the best value of the increment, we test five ratios, which are 100:1, 10:1, 1:1, 1:10 and 1:100. Note that the weight scale is 1:1 in the version of the proposed VWDT algorithm, as described in Algorithm 1. For a fair comparison, we use the same random seed in each run, so that they start from the same initial solution on each instance, respectively. Fig. 3 illustrates the evolution of the number of uncovered vertices for the variants of VWDT with the ratios 100:1, 10:1, 1:1, 1:10, 1:100 and DT as the search proceeds. Each point (x, y) means that there are $2^y - 1$ vertices which are not covered by any center at 10^x iteration. We can observe that, DT can obtain better infeasible solutions than VWDT in the initial stage. However, as discussed above, the tabu search simply prohibits a candidate center from being a center, instead of directly preventing a vertex from being uncovered. One may imagine that, in extreme conditions, the search may never consider covering some hard-to-cover clients at all. As expected, VWDT outperforms DT after 10^4 iterations on all 12 instances. Fig. 3 shows that all the five variants of VWDT outperform DT on all the 12 instances, and the gaps expand as p gets larger. According to Fig. 3, in the initial stage, the higher the ratio of the initial weight to the increment is, the faster the algorithm converges. However, their convergence rate gradually decreases in the later stage of the search. VWDT with weight scale 1:1 obtains better overall results than other weight scales for 10 out of the 12 instances in terms of the rate of convergence, while obtaining slightly worse or comparable results to one or two algorithms for the remaining 2 instances (Fig. 3(k) and 3(l)), which highlights the effectiveness of the vertex weighting strategy. These observations confirm that the vertex weighting is essential for VWDT to obtain the competitive results.

7. Conclusion

In this paper, we study the p-center problem, and design a meta-heuristic algorithm, called vertex weighting-based double-tabu search (VWDT) for solving this classical and challenging NP-hard problem. We decompose the p-center problem into a series of decision subproblems, and solve each one of them by combining the hybrid double-tabu search with the vertex weighting technique. It improves the best known results on 84 out of 510 widely used benchmark instances in the literature, while matching the best records in the literature for all the remaining ones. The experimental results demonstrate that VWDT is highly competitive in terms of the solution quality and the computational efficiency. In addition, we carried out experiments to analyze the essential ingredients of our VWDT, such as the vertex weighting technique and double-tabu strategy. The experiments show that these two components play significant roles in reaching a trade-off between exploration and exploitation of the search, thus making VWDT powerful and robust.

The success of solving the p-center problem inspires us that, it would be appealing to investigate the combination of the solution-based and attribute-based tabu strategies with the weighting technique in the future. As these strategies are independent of the p-center problem, it seems promising to apply them to other optimization problems.

CRediT authorship contribution statement

Qingyun Zhang: Methodology, Software, Validation, Formal analysis, Investigation, Resources, Writing – original draft, Visualization. **Zhipeng Lü:** Conceptualization, Methodology, Supervision. **Zhouxing Su:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Project administration. **Chumin Li:** Writing – review & editing.

Data availability

The benchmark data have already been publicly available on ORLIB and TSPLIB.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 72101094 and the Special Project for Knowledge Innovation of Hubei Province under Grant 2022013301015175.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cor.2023.106373.

References

Al-Khedhairi, A., Salhi, S., 2005. Enhancements to two exact algorithms for solving the vertex p-center problem. J. Math. Model. Algorithms 4 (2), 129–147.

Amiri, A., 2006. Designing a distribution network in a supply chain system: Formulation and efficient solution procedure. European J. Oper. Res. 171 (2), 567–576.

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

Cai, S., Su, K., Luo, C., Sattar, A., 2013. NuMVC: An efficient local search algorithm for minimum vertex cover. J. Artificial Intelligence Res. 46, 687–716.

Cai, S., Su, K., Sattar, A., 2011. Local search with edge weighting and configuration checking heuristics for minimum vertex cover. Artificial Intelligence 175 (9–10), 1672–1696.

Calik, H., Tansel, B.C., 2013. Double bound method for solving the p-center location problem. Comput. Oper. Res. 40 (12), 2991–2999.

Caruso, C., Colorni, A., Aloi, L., 2003. Dominant, an algorithm for the p-center problem. European J. Oper. Res. 149 (1), 53–64.

Church, R., ReVelle, C., 1974. The maximal covering location problem. Pap. Reg. Sci. 32 (1), 101–118.

Contardo, C., Iori, M., Kramer, R., 2019. A scalable exact algorithm for the vertex p-center problem. Comput. Oper. Res. 103, 211–220.

Daskin, M.S., 2000. A new approach to solving the vertex p-center problem to optimality: Algorithm and computational results. Commun. Oper. Res. Soc. Jpn. 45 (9), 428–436.

Daskin, M.S., 2013. Center problems. In: Daskin, M.S. (Ed.), Network and Discrete Location: Models, Algorithms, and Applications, Second Edition. John Wiley & Sons, Ltd, pp. 193–234.

Elloumi, S., Labbé, M., Pochet, Y., 2004. A new formulation and resolution method for the p-center problem. INFORMS J. Comput. 16 (1), 84–94.

Ferone, D., Festa, P., Napoletano, A., Resende, M.G., 2017. A new local search for the p-center problem based on the critical vertex concept. In: 11th Int. Conf. Learn. Intell. Optim. LION 11, Springer, pp. 79–92.

Floyd, R.W., 1962. Algorithm 97: shortest path. Commun. ACM 5 (6), 345.

Gaar, E., Sinnl, M., 2022. A scaleable projection-based branch-and-cut algorithm for the p-center problem. European J. Oper. Res. 303 (1), 78–98.

Gao, C., Yao, X., Weise, T., Li, J., 2015. An efficient local search heuristic with row weighting for the unicost set covering problem. European J. Oper. Res. 246 (3), 750–761.

Garcia-Diaz, J., Menchaca-Mendez, R., Menchaca-Mendez, R., Hernández, S.P., Pérez-Sansalvador, J.C., Lakouari, N., 2019. Approximation algorithms for the vertex k-center problem: Survey and experimental evaluation. IEEE Access 7, 109228–109245.

Garcia-Diaz, J., Sanchez-Hernandez, J.J., Menchaca-Mendez, R., Menchaca-Méndez, R., 2017. When a worse approximation factor gives better performance: A 3-approximation algorithm for the vertex k-center problem. J. Heuristics 23 (5), 349–366.

Glover, F., 1989. Tabu search-part I. ORSA J. Comput. 1 (3), 190–206.

Gonzalez, T.F., 1985. Clustering to minimize the maximum intercluster distance. Theoret. Comput. Sci. 38, 293–306.

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

Hochbaum, D.S., Shmoys, D.B., 1985. A best possible heuristic for the k-center problem. Math. Oper. Res. 10 (2), 180–184.

Ilhan, T., Ozsoy, F., Pinar, M., 2002. An Efficient Exact Algorithm for the Vertex p-Center Problem and Computational Experiments for Different Set Covering Subproblems. Technical Report, Bilkent University, Department of Industrial Engineering.

Irawan, C.A., Salhi, S., Drezner, Z., 2016. Hybrid meta-heuristics with VNS and exact methods: application to large unconditional and conditional vertex p-centre problems. J. Heuristics 22 (4), 507–537.

Kariv, O., Hakimi, S.L., 1979. An algorithmic approach to network location problems. I: The p-centers. SIAM J. Appl. Math. 37 (3), 513–538.

Lai, X., Hao, J.K., Glover, F., Lü, Z., 2018. A two-phase tabu-evolutionary algorithm for the 0–1 multidimensional knapsack problem. Inform. Sci. 436, 282–301.

Liao, C., Ting, C., 2018. A novel integer-coded memetic algorithm for the set k-cover problem in wireless sensor networks. IEEE Trans. Cybern. 48 (8), 2245–2258.

Liu, X., Fang, Y., Chen, J., Su, Z., Li, C., Lü, Z., 2020. Effective approaches to solve P-center problem via set covering and SAT. IEEE Access 8, 161232–161244.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T., 2016. The irace package: Iterated racing for automatic algorithm configuration. Oper. Res. Perspect. 3, 43–58.

Luo, C., Su, K., Cai, S., 2012. Improving local search for random 3-SAT using quantitative configuration checking. In: Proc. 20th Eur. Conf. Artif. Intell.. ECAI 2012, IOS Press, pp. 570–575.

Martinich, J.S., 1988. A vertex-closing approach to the p-center problem. Naval Res. Logist. 35 (2), 185–201.

Minieka, E., 1970. The m-center problem. SIAM Rev. 12 (1), 138–139.

Mladenović, N., Brimberg, J., Hansen, P., A.Moreno-Pérez, J., 2007. The p-median problem: A survey of metaheuristic approaches. European J. Oper. Res. 179 (3), 927–939.

Mladenović, N., Labbé, M., Hansen, P., 2003. Solving the p-center problem with tabu search and variable neighborhood search. Networks 42 (1), 48–64.

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

Reinelt, G., 1991. TSPLIB–A traveling salesman problem library. ORSA J. Comput. 3 (4), 376–384.

Rigas, E.S., Ramchurn, S.D., Bassiliades, N., 2018. Algorithms for electric vehicle scheduling in large-scale mobility-on-demand schemes. Artificial Intelligence 262, 248–278.

Robič, B., Mihelič, J., 2005. Solving the k-center problem efficiently with a dominating set algorithm. J. Comput. Inform. Technol. 13 (3), 225–234.

Salhi, S., Al-Khedhairi, A., 2010. Integrating heuristic information into exact methods: The case of the vertex p-centre problem. J. Oper. Res. Soc. 61 (11), 1619–1631.

Shmoys, D.B., 1995. Computing near-optimal solutions to combinatorial optimization problems. Comb. Optim. 20, 355–397.

Toregas, C., Swain, R., ReVelle, C., Bergman, L., 1971. The location of emergency service facilities. Oper. Res. 19 (6), 1363–1373.

Voudouris, C., Tsang, E.P.K., 2003. Guided local search. In: Glover, F., Kochenberger, G.A. (Eds.), Handbook of Metaheuristics. Springer US, Boston, MA, pp. 185–218. http://dx.doi.org/10.1007/0-306-48056-5_7.

Wang, Y., Lü, Z., Su, Z., 2021. A two-phase intensification tabu search algorithm for the maximum min-sum dispersion problem. Comput. Oper. Res. 135, 105427.

Wang, Y., Wu, Q., Glover, F., 2017. Effective metaheuristic algorithms for the minimum differential dispersion problem. European J. Oper. Res. 258 (3), 829–843.

Xia, L., Yin, W., Dong, J., Wu, T., Xie, M., Zhao, Y., 2010. A hybrid nested partitions algorithm for banking facility location problems. IEEE Trans. Autom. Sci. Eng. 7 (3), 654–658.

Yin, A.H., Zhou, T.Q., Ding, J.W., Zhao, Q.J., Lv, Z.P., 2017. Greedy randomized adaptive search procedure with path-relinking for the vertex p-center problem. J. Comput. Sci. Tech. 32 (6), 1319–1334.

Zhang, X., Li, B., Cai, S., Wang, Y., 2021. Efficient local search based on dynamic connectivity maintenance for minimum connected dominating set. J. Artificial Intelligence Res. 71, 89–119.

Zhang, Q., Lü, Z., Su, Z., Li, C., Fang, Y., Ma, F., 2020. Vertex weighting-based tabu search for p-center problem. In: Proc. 29th Int. Joint Conf. Artif. Intell. IJCAI 2020, pp. 1481–1487.