Computers & Operations Research 149 (2023) 106023

Contents lists available at ScienceDirect

Computers and Operations Research

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

Exploiting flat subspaces in local search for *p*-Center problem and two fault-tolerant variants

Seyed R. Mousavi

Coventry University, UK

A R T I C L E I N F O

Keywords:
Flat subspace
Heuristic function
Local search
Metaheuristic
Move operation
p-Center problem
Search space

A B S T R A C T

In this paper, local search algorithms are proposed for the *p*-Center, α-Neighbour *p*-Center and *p*-Next Center facility location problems. The α-Neighbour *p*-Center and *p*-Next Center problems may be viewed as two fault-tolerant variants of the *p*-Center problem. The algorithm proposed for *p*-Center outperforms the most recent state-of-the-art metaheuristic for this problem using standard datasets. The proposed algorithm for *p*-Next Center also outperforms an existing, more complex, state-of-the-art metaheuristic for this problem. The algorithm proposed for α-Neighbour *p*-Center is the first metaheuristic for this problem, to the best of the author's knowledge. The proposed algorithms share a common design, which is the integration of the first-improvement local search with strategies to exploit flat subspaces in the search space. The overall success of this design paradigm motivates further investigation about its properties and applications to similar NP-hard optimisation problems.

1. Introduction

One of the well-studied facility location problems is the *p*-Center (*pC*) problem, which is the problem of selecting a pre specified number of vertices in a graph as facility centers such that the maximum distance from a client vertex to its closest facility is minimised (Hakimi, 1964; Hakimi, 1965; Minieka, 1970). Because of its min–max property, it may be used to model real-world applications where the high service cost of a demand point is not compensated by the low service costs of other demand points. It also has applications in solving location covering problems. Applications cited in the literature for such problems and their variants include locating emergency service points such as police stations, ambulances, and fire brigades in Çalık et al. (2019), television transmitters, warning sirens, and sprinkler systems in Suzuki and Drezner (1996), mobile base stations mounted on unmanned aerial vehicles in Lyu et al. (2016), delivery drone battery depos in Liu (2022), and mobile sinks in wireless sensor networks in Solmaz et al. (2014), among others. Callaghan (2016) presents a brief review of the previous studies using real data, which include, among others, locating geriatric and diabetic health care clinics in Spain (Pacheco and Casado, 2005), emergency warning sirens in Dublin, Ohio (Murray et al., 2008), and urgent relief distribution centres for earthquake injured residents in Taiwan (Lu, 2013). Other studies using real data include cell phone towers in northern Orange County, California (Drezner and Drezner, 2014) and emergency rescue stations in the high-speed railway network in China (Wang et al., 2022).

Among numerous exact algorithms proposed for this problem are Daskin (1995, 2000), Elloumi et al. (2004), Al-Khedhairi and Salhi (2005), Chen and Chen (2009), Çalık and Tansel (2013) and Contardo et al. (2019). However, because of its NP-hardness (Kariv and Hakimi, 1979), guaranteeing to find optimal solutions requires super-polynomial running time unless P = NP. Therefore, several inexact, mostly metaheuristic, algorithms have also been proposed to address the problem in affordable time but with no optimality guarantee.

Mladenović et al. (2003) proposed Variable Neighbourhood Search (VNS) and Tabu Search (TS) for the problem. Hassin et al. (2003) proposed a local search strategy where solutions are compared lexicographically and applied this strategy to the *pC* problem. A Scatter Search (SS) metaheuristic was proposed by Pacheco and Casado (2005). Pullan (2008) devised a heuristic used in a Memetic Algorithm (MA). Davidović et al. (2011) introduced a variant of the Bee Colony Optimization (BCO) and showed its superiority to the standard BCO for the problem. Other bee colony metaheuristics were proposed by Yurtkuran and Emel (2014) and Jayalakshmi and Singh (2018). Ferone et al. (2017) proposed a Greedy Randomized Adaptive Search (GRASP) for the problem. Yin et al. (2017) used the local search devised in Pullan (2008) with a modified tabu strategy within a GRASP metaheuristic with Path Relinking (PR). Two more heuristics for the problem were evaluated by Yadav and

E-mail address: seyed.mousavi@coventry.ac.uk.

https://doi.org/10.1016/j.cor.2022.106023
Received 3 October 2020; Received in revised form 10 June 2022; Accepted 10 September 2022
Available online 17 September 2022

Prakash (2020). The problem may also be formulated as the Set Covering or Boolean Satisfiability problems (Caruso et al., 2003; Liu et al., 2020).

A generalisation of the *pC* problem is the α-Neighbour *p*-Center (*αNpC*) problem, $1 \leqslant \alpha \leqslant p$, whose objective is to minimize the maximum distance between a client and its α-closest facility center. The case $\alpha = 1$ is equivalent to *pC*, and the case $\alpha > 1$ may be viewed as a fault-tolerant variant of *pC* (by catering for the potential failure of up to $\alpha - 1$ facilities). This problem was first introduced and shown NP-hard by Krumke (1995) and further explored by Chaudhuri et al. (1998), Khuller et al. (2000) and Chen and Chen (2013). Two variants of this problem (*continuous* and *variable αNpC*) were introduced by Chen and Chen (2013) and Callaghan et al. (2019).

Recently, another NP-hard fault-tolerant variant of the *pC* problem, called *p*-Next Center (*pNC*), was introduced by Albareda-Sambola et al. (2015) where each client is assigned to two facility centers, a primary and a secondary, so that the latter can be used as the backup in case the former becomes unavailable, e.g. in natural disasters. The primary center is required to be a closest center to the client. A key assumption in this problem is that the potential failure of the primary center is not known in advance. That is, a client first pays the cost of reaching their primary center and, if unavailable, will pay the extra cost of reaching their backup center. Therefore, in case of failure, the total cost to the client will be the sum of these two costs. The goal is to locate the centers such that the maximum of these total costs is minimised. For discussion on the potential applications of this problem, the reader is referred to Albareda-Sambola et al. (2015). Two metaheuristics and their hybrid were proposed for this problem by López-Sánchez et al. (2019).

Among other similar facility location problems are Probabilistic *p*-Center (Martínez-Merino et al., 2017), α-All-Neighbour *p*-Center (Khuller et al., 2000), Capacitated *p*-Center (Kramer et al., 2020), and *p*-Median (Mladenović et al., 2007) problems.

To the best of the author's knowledge, the current state-of-the-art metaheuristics for *p*C are those proposed by Pullan (2008) and Yin et al. (2017). No metaheuristic has yet been proposed for *αNpC*, to the best of the author's knowledge, and the current state-of-the-art metaheuristic for *pNC* is the hybrid metaheuristic proposed by López-Sánchez et al. (2019).

The main contributions of this paper are new state-of-the-art metaheuristics for the *pC*, *αNpC*, and *pNC* problems. More specifically, metaheuristics are proposed to outperform the state-of-the-art metaheuristics of Yin et al. (2017) and López-Sánchez et al. (2019) for *pC* and *pNC*, respectively, and another metaheuristic as the (first) state-of-the-art metaheuristic for *αNpC*.

To that end, the first-improvement local search is used together with strategies to exploit the *flat* subspaces of the search space due to the max–min nature of these problems. Such strategies are not used in the current state-of-the-art metaheuristics for *pC* and *pNC*.

In particular, two strategies are used to benefit from flat subspaces. The first strategy is to accept not only downhill moves but also flat moves that are 'promising' according to some heuristic function. Let $f(.)$ be the objective function and P and P_1 be two neighbours in the search space with the same objective value $f(P) = f(P_1)$. This strategy is to replace the objective function $f(.)$ with a more accurate heuristic function $h(.)$ to evaluate these points and decide on the potential move from P to P_1. To that end, a heuristic function is used to take into account properties additional to those already captured by the objective function. Therefore, P and P_1, which have the same objective value, may now have different heuristic values $h(P) \neq h(P_1)$. This means, while such neighbours are flat in the search space with respect to $f(.)$, they may no longer be so with respect to $h(.)$. For this reason, this strategy is called the *unflattening strategy* in this paper. Using this strategy, we can potentially distinguish between flat moves and accept those likely to be beneficial. That is, although the objective value is not immediately improved by such moves, it is likely to be improved in subsequent moves. More specifically, the intuition for using this strategy is to potentially reduce the expected running time needed to improve the objective value. However, it requires an effective, but not computationally expensive, heuristic function. This strategy was used in Mousavi et al. (2012) within the local search phase of a GRASP algorithm to address the Far From Most Strings problem. In Mousavi and Esfahani (2012), a similar approach was used in both the construction and the local search phases of a GRASP algorithm for the Closest String problem. This strategy was also reported useful in a stand-alone local search algorithm and the local search phase of a GRASP algorithm proposed for the *pC* problem by Hassin et al. (2003) and Ferone et al. (2017), respectively, though neither of them is a recent state-of-the-art for the problem.

The second strategy is to always accept flat moves (in addition to downward moves). That is, the move from P and P_1 is accepted when P_1 has a better or the same objective value $f(P_1) \leqslant f(P)$. The intuition for accepting flat moves in the proposed algorithms is to increase diversification without sacrificing intensification. Intensification aims at utilising the neighbourhood of the best solution discovered, whereas diversification aims at exploring other parts of the search space to potentially discover better solutions. In general, these two mechanisms are contradictory in the sense that diversification requires moving to a point whose objective value is (most likely) worse than that of the best point found so far. However, flat moves promote diversification without reducing the objective value. This strategy is known in the literature as the *Improving and Equal* (*IE*) (also, *Improving or Equal*) move-acceptance hyper-heuristic (Misir et al., 2009; Burke et al., 2013; Jackson et al. 2018). This strategy is called the *IE strategy* in this paper. This strategy is also combined with the unflattening strategy in the proposed algorithms by accepting moves to neighbours with better or the same *heuristic* values.

The current state-of-the-art metaheuristics for *pC* and *pNC* do not use these strategies. The state-of-the-art metaheuristic proposed by Yin et al. (2017) for the *pC* problem is a GRASP algorithm with the PR operator. The GRASP part consists of a construction and a local search phase. The local search is that proposed by Pullan (2008), in a memetic algorithm, but with a modified tabu strategy. This pair of construction and local search phases is run independently several times. The PR procedure is intended to keep a list of the best solutions found in these independent runs and use them to further improve the solutions obtained at each run. The adopted local search in this metaheuristic uses the best-improvement strategy. This means that the algorithm rejects all identified flat moves unless no better move can be found. That is, the *IE* strategy is not employed. The local search procedure does not use extra information to evaluate flat neighbours either. On the contrary, it uses even less information to determine the "best" move. More specifically, to evaluate the quality of the neighbours, it uses a heuristic function that is less accurate (hence faster to compute) than the objective function. That is, the heuristic function uses less information than that used by the objective function. This is in contrast to the strategy used in the proposed metaheuristics, which is to use a heuristic function that is more accurate than the objective function by using extra information to distinguish between flat neighbours.

The state-of-the-art metaheuristic of López-Sánchez et al. (2019) for *pNC* is a hybrid of GRASP and Basic VNS (BVNS). The local search procedure in this metaheuristic resides in its VNS component, which serves as the local search phase of GRASP. This local search procedure is the standard local search with the first-improvement strategy, which only accepts moves that improve the objective value. That is, no flat move is accepted. No heuristic is used in this metaheuristic to distinguish between flat neighbours either.

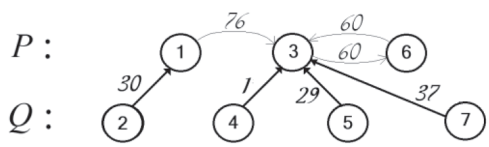
As confirmed by the experimental results, these state-of-the-art metaheuristics are outperformed by the proposed algorithms.

The rest of the paper is organised as follows. Section 2 provides basic notations and the formal definitions of the problems. The proposed algorithms are described in Section 3. Section 4 reports the experimental results, and Section 5 concludes the paper.

(a)

	1	2	3	4	5	6	7
1:	0	30	76	77	105	136	113
2:	30	0	46	47	75	106	83
3:	76	46	0	1	29	60	37
4:	77	47	1	0	28	59	36
5:	105	75	29	28	0	31	8
6:	136	106	60	59	31	0	39
7:	113	83	37	36	8	39	0

(b)



(c)

	Vertex i	Primary Center (its cost)	Secondary Center (its cost)	f_i
	1	1(0)	3(76)	76
P:	3	3(0)	6(60)	60
	6	6(0)	3(60)	60
	2	1(30)	3(76)	**106**
Q:	4	3(1)	6(60)	61
	5	3(29)	6(60)	89
	7	3(37)	6(60)	97
			$f(P)$	= **106**

Fig. 1. Analysis of the candidate solution $P = \{1, 3, 6\}$ in Example 1. (a) The shortest distance between each pair of vertices. Note the symmetry of the table. (b) Assignment of vertices to their closest centers. (c) The primary and secondary centers, their costs (inside parentheses), the f_i values, and the final objective value $f(P)$.

2. Basic notations and formal definitions

Let $V = \{v_1, \cdots, v_n\}$ be a set of vertices and $d_{ij} \geqslant 0$ be the (shortest) distance between vertices v_i and v_j. Then, given a distance matrix $D = [d_{ij}]_{n\times n}$ and an integer $p \in \{2, \cdots, n-1\}$, the p-Center, α-Neighbour p-Center, where $\alpha \in \{1, \cdots, p\}$, and p-Next Center problems are the problems of finding a set $P \subset V$ such that $|P| = p$ and the objective value $f(P) = \max\limits_{1\leqslant i\leqslant n}\{f_i(P)\}$ is minimised, where $f_i(P)$ is the *cost of vertex* v_i defined differently for these problems as follows. For the p-Center problem, the cost of a vertex is its distance to a nearest vertex in P. For the α-Neighbour p-Center problem, the cost of a vertex in P is zero and the cost of any other vertex is defined as its distance to an α-nearest member of P. Finally, for the p-Next Center problem, the cost of a vertex $v_i, i = 1, \cdots, n$, is defined as the following.

$$f_i(P) = \min_{v_j\in P}\{d_{ij}\} + \min_{\substack{j'\in \operatorname{argmin}_{v_j\in P}\{d_{ij}\} \\ k\neq j', v_k\in P}} \{d_{j'k}\}$$

Let $f^{(r)}(P), r = 1, \cdots, n$, denote the rth largest value among $f_i(P), i = 1, \cdots, n$. Then, the objective function f (for all these problems) is equal to $f^{(1)}$. For brevity, f_i and $f^{(r)}$ are used to denote, respectively, $f_i(P)$ and $f^{(r)}(P)$, $i, r = 1, \cdots, n$, when no ambiguity arises. The set $V\backslash P$ is also denoted by Q.

A candidate solution, or for short a solution, is a set P of p vertices which represent the *facility centers*. Such a vertex may simply be called a facility or a center. Other vertices (i.e. members of Q) are called *clients*. A candidate solution corresponds to a point in the search space, so these terms are used interchangeably. The optimal solution is a candidate solution with the minimum objective value. A neighbour P_1 of a candidate solution P is a candidate solution obtained by replacing a member of P with a member of Q, which means $|P\cap P_1| = p-1$ (equivalently, $|P\cup P_1| = p+1$). The set of the neighbours of P is denoted by $N(P)$. A candidate solution $P_1 \in N(P)$ is a *flat neighbour* of P (with respect to f) if $f(P) = f(P_1)$.

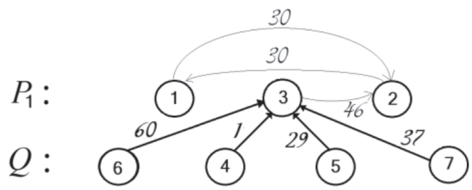
The p-Center and α-Neighbour p-Center problems are well-studied in the literature. To elaborate on the p-Next Center problem, an example is now presented which is derived from the first data file `pmed1` used in the experimental results with a reduced number of vertices.

Example 1. Assume we want to open 3 facility centers for which there are 7 candidate locations (vertices) labelled with 1 to 7. The shortest distances between all pairs of vertices are given in Fig. 1.a.

For this problem instance, $n = 7$ and $p = 3$. Any set of 3 vertices is a candidate solution, e.g. $P = \{1, 3, 6\}$, for which $Q = \{2, 4, 5, 7\}$. For this candidate solution, f_1 to f_7 are now calculated. For this purpose, the closest center to each vertex $i = 1, \cdots, 7$ is identified. Fig. 1.b displays the vertices in two rows. The top row presents the facility centers, i.e. the members of P, and the bottom row presents those in Q. An arrow from a node $i \in Q$ to a facility node $j \in P$ means that j is the primary center for i, i.e. a closest center to i. The weight of the arrow is the distance d_{ij}. For example, the arrow weighted 30 from 2 to 1 indicates that the primary center for 2 is 1, away by 30 units. In this example, the closest center to each node is unique, which is not always the case. An arc from a node $i \in P$ to another node $j \in P$ means that j is the *secondary* center for i; its primary center is itself. For example, the arc from 1 to 3 means that node 3, away by 76 units, is the secondary center for node 1. In addition, it is the secondary center for node 2 whose primary center is 1. Similarly, the primary and the secondary centers for the nodes 4, 5 and 7 are the centers 3 and 6, respectively.

The information provided in Fig. 1.b is now used to calculate the values f_1 to f_7. For each center $i \in P$, f_i is simply the weight of its outgoing arc, i.e. the cost of reaching its secondary center. Therefore, $f_1 = 76$ and $f_3 = f_6 = 60$. However, for a node $i \in Q$, f_i is the sum of two costs, the weight of its outgoing arrow to a center j and the weight of the arc outgoing from j. For example, $f_2 = 30 + 76 = 106$. Similarly, $f_4 = 1 + 60 = 61$, $f_5 = 29 + 60 = 89$, and $f_7 = 37 + 60 = 97$. Fig. 1.c. summarises the primary and the secondary centers and the f_i value for each node $i = 1, \cdots, 7$. The objective value $f(P)$ is the largest value among $f_i, i = 1, \cdots, 7$, which is 106, shown in the last row. The second largest value in this example is $f^{(2)} = 97$. An optimal solution for this problem instance (not shown in the figure) is $P^* = \{2, 3, 7\}$ with objective value $f(P^*) = 76$.

(a)



(b)

	Vertex i	Primary Center (its cost)	Secondary Center (its cost)	f_i
P_1:	1	1(0)	2(30)	30
	3	3(0)	2(46)	46
	2	2(0)	1(30)	30
Q:	6	3(60)	2(46)	**106**
	4	3(1)	2(46)	47
	5	3(29)	2(46)	75
	7	3(37)	2(46)	83
			$f(P_1)$	= **106**

Fig. 2. Analysis of the candidate solution $P_1 = \{1,3,2\}$ for the problem instance in Example 1. This candidate solution is a neighbour of $P = \{1,3,6\}$ analysed in Fig. 1. (a) Assignment of vertices to their closest centers. (b) The primary and secondary centers, their costs (inside parentheses), the f_i values, and the final objective value $f(P_1)$.

3. Proposed algorithms

This section presents the algorithms proposed for the *pNC*, *αNpC* and *pC* problems. These algorithms are based on the first-improvement local search (Blum and Roli, 2003) integrated with the unflattening and *IE* strategies.

Definition. A function h from the set of candidate solutions to the set of real numbers is called f-consistent if for all candidate solutions P and P_1, $f(P) < f(P_1) \Rightarrow h(P) < h(P_1)$.

f-consistent heuristic functions are used to integrate the unflattening strategies with the local search algorithms for the *pNC*, *αNpC* and *pC* problems. The f-consistency of such a heuristic allows for the incorporation of the unflattening strategy in the local search by simply replacing the objective function f with the heuristic function h.

The combination of the unflattening and *IE* strategies in the proposed algorithms is defined as the acceptance of moves to neighbours with the same objective value and the same heuristic value in addition to those accepted by the unflattening strategy alone. That is, a move from P to P_1 is accepted iff P_1 has either a better objective value or the same objective value *and* a better or the same heuristic value, i.e.

$$f(P_1) < f(P) \text{ OR } (f(P_1) = f(P) \text{ AND } h(P_1) \leqslant h(P)).$$

However, by the f-consistency of h, this condition is simplified to $h(P_1) \leqslant h(P)$.

The replacement of the "$\leqslant$" operator with "$<$" in these conditions would yield the acceptance criteria for the unflattening strategy alone.

3.1. Proposed algorithm for pNC

The algorithm proposed for *pNC* is the basic local search (*BLS*) integrated with the unflattening and *IE* strategies. Here, *BLS* is the classic first-improvement local search within a random-restart loop. At each iteration of the loop, it starts with a random solution and keeps moving to improved neighbours, on the first-improvement basis, until it reaches a local minima. Before presenting the proposed pseudocode, the unflattening heuristic function is described.

Let c be an integer greater than $2 \times d_{max}$, where $d_{max} = \max_{1 \leqslant i,j \leqslant n} \{d_{ij}\}$. Then, the proposed heuristic function h_K, $1 \leqslant K \leqslant n$, is defined as the following.

$$h_K(P) = \sum_{r=1}^{K} c^{K-r} f^{(r)}(P) \tag{1}$$

Recall that $f^{(r)}(P)$ is the rth largest value among $f_i(P), i = 1, \cdots, n$. Also note that $h_1(P) = f(P)$. Both the distinguishing power and the computational cost of $h_K(P)$ increase with K.

In the following, for simplicity and without loss of generality, the distance values are assumed to be scaled (based on the desired accuracy) to integers. For example, for centimetre-level accuracy, if the distance between vertex i and vertex j is 2 meters, then $d_{ij} = 200$. As a result, the objective and the heuristic values will also be integers.

Proposition 1. *The heuristic function h_K given in* (1) *is f-consistent.*

Proof. It is proved by induction on K. The base case holds trivially because $h_1 = f^{(1)} = f$. Assume h_K is f-consistent for some $K = k_1$, $1 \leqslant k_1 < n$. It is now proved so for $K = k_1 + 1$.

$$\begin{aligned} f(P) < f(P_1) &\Rightarrow h_{k_1}(P) < h_{k_1}(P_1) && (\text{by the induction hypothesis}) \\ &\Rightarrow (h_{k_1}(P_1) - h_{k_1}(P)) \geqslant 1 && (\text{because heuristic values are integers}) \\ &\Rightarrow c(h_{k_1}(P_1) - h_{k_1}(P)) \geqslant c && (\text{because } c > 0) \end{aligned} \tag{2}$$

On the other hand, $f^{(r)}(P) \leqslant 2 \times d_{max}$, $1 \leqslant r \leqslant n$, which implies $c > f^{(k_1+1)}(P)$. This in turn implies $c > f^{(k_1+1)}(P) - f^{(k_1+1)}(P_1)$ because $f^{(k_1+1)}(P_1)$ is nonnegative. Using this result together with (2) yields the following.

$$\begin{aligned} f(P) < f(P_1) &\Rightarrow c(h_{k_1}(P_1) - h_{k_1}(P)) > f^{(k_1+1)}(P) - f^{(k_1+1)}(P_1) \\ &\Rightarrow c\, h_{k_1}(P) + f^{(k_1+1)}(P) < c\, h_{k_1}(P_1) + f^{(k_1+1)}(P_1) \\ &\Rightarrow h_{k_1+1}(P) < h_{k_1+1}(P_1) \left(\text{by the recursion } h_{k_1+1} = c\, h_{k_1} + f^{(k_1+1)}\right) \blacksquare \end{aligned}$$

The value $h_K(P)$ may be viewed as a number in radix c with K digits $f^{(1)}$ to $f^{(K)}$ because c is greater than $f^{(r)}$, $1 \leqslant r \leqslant K$. Therefore, if the most significant digit $f^{(1)}(P)$ of $h_K(P)$ is less than that of $h_K(P_1)$, then the whole number $h_K(P)$ is less than $h_K(P_1)$. If their $f^{(1)}$ digits are equal, their $f^{(2)}$ digits determine which number is less, unless these digits are also equal in which case their $f^{(3)}$ digits matter, and so on.

It is not hard to see that Proposition 1 also holds for decimal objective and heuristic values by appropriately scaling up the constant c.

Let A_K be the algorithm obtained by replacing the objective function f in *BLS* with h_K, $K > 1$. The f-consistency of h_K means that A_K is consistent with *BLS* in accepting downhill and rejecting uphill moves. The only difference is that any move from P to $P_1 \in N(P)$ with the same objective value but a better heuristic value is rejected as a flat move in *BLS* but is accepted as a downhill move (with respect to h_K) in A_K.

Example 2. Consider the problem instance and the candidate solution $\mathrm{P} = \{1,3,6\}$ in Example 1. Consider the neighbour $\mathrm{P}_1 = \{1,3,2\}$ obtained by replacing center 6 in P with center 2. It is now shown that the move from P to P_1 is rejected in BLS but is accepted in A_K, $K > 1$. Fig. 2 analyses P_1.

As can be seen in Fig. 2, $f(P_1) = 106 = f(P)$, which means P and P_1 are flat neighbours with respect to f. Therefore, *BLS* rejects the move from P to P_1. Assume, for simplicity and without loss of generality, that the algorithm A_2 is used, i.e. the heuristic function is $h_2 = cf^{(1)} + f^{(2)}$, where c is an integer greater than $2 \times d_{max}$. Here, $d_{max} = 136$ (please see the distance matrix in Fig. 1.a). So, $c = 273$ can be used. Then, although the $f^{(1)}$ values of P and P_1 are the same (106), their heuristic values will be different because of their different $f^{(2)}$ values (97 and 83, respectively). In particular, $h_2(P) = 273 \times 106 + 97 = 29035$ and $h_2(P_1) = 273 \times 106 + 83 = 29021$. Therefore, P_1 has a better (smaller) heuristic value and A_2 accepts the move from P to P_1.

Algorithm 1 presents the proposed pseudocode. It receives, as inputs, the matrix D of the shortest distances $d_{ij}, 1 \leqslant i, j \leqslant n$ and the number p of facility centers. Its parameter K determines the unflattening heuristic function h_K. It uses arrays to represent P and Q. Its main loop (line 1) repeats the same process (lines 2–17) until its termination condition is met. The termination condition could be based on a fixed number of iterations, fixed running time, target solution quality or any combination of these, among other options. At each iteration, it starts with a random initial solution P (line 2) and executes a **while** loop (lines 4–17) until no downward move is performed. At each iteration of the **while** loop, it first resets the *improved* flag. Then, it goes through every pair ($i \in \{1, \cdots, p\}, j \in \{1, \cdots, q\}$), where $q = n - p$, (lines 6–7) to generate a neighbour P_1 of P obtained by replacing the facility center $P[i]$ with $Q[j]$ (without changing P and Q) (line 8). Next, it compares the heuristic values of P_1 and P (line 9). If P_1 has a better value, it moves to P_1 and sets the *improved* flag (line 11). If the heuristic values are equal, it also accepts the move (line 13) but does not set the flag. This means, while both downward and flat moves (with respect to h_K) are accepted, they are treated differently. Because this flag is only set for the downward (not flat) moves and the **while** loop in line 4 stops if it is not set, the *IE* strategy does not result in infinite cycling. Eventually, the best solution found in all the iterations is returned (line 19).

This algorithm is identical to *BLS* except that:

(i) It incorporates the unflattening strategy by replacing the objective function f in *BLS* with the heuristic function h_K in line 9.
(ii) It incorporates the *IE* strategy by using the **else** branch in lines 12–13.

This algorithm is called B_K, where parameter K determines the heuristic function h_K, and A_K is used to refer to the corresponding algorithm without the *IE* strategy (i.e. without lines 12–13). Note that *BLS* is equivalent to A_1.

Algorithm 2 calculates the heuristic value $h_K(P)$ for a given candidate solution P and a value of the parameter K. It uses other data relevant to the problem instance, such as the distance matrix and the constant c, as global variables. The first **for** loop in lines 1–4 calculates f_i for all facility centers $v_i \in P$. Recall that such a center is its own primary center and only needs a backup center. The second **for** loop, lines 5–14, calculates f_i for each client vertex $v_i \in Q$. In line 15, the k largest values $f^{(1)}$ to $f^{(K)}$ among f_i, $v_i \in V$, are identified, which are then used to calculate the heuristic value (*hValue*) in lines 16–19 by the following recursion.

$$h_K(P) = \begin{cases} c \times h_{K-1}(P) + f^{(K)}(P) & K > 1 \\ f^{(1)}(P) & K = 1 \end{cases}$$

This algorithm is efficient. The first **for** loop (lines 1–4) runs in $O(p^2)$. The second **for** loop (lines 5–14) runs in $O(qp)$, so the calculation of all the f_i values, $i = 1, \cdots, n$, (lines 1–14) takes $O(np)$. It takes $O(Kn)$ to calculate $f^{(r)}, r = 1, \cdots, K$ (line 15). These values are then used to calculate *hValue* in $O(K)$ (lines 16–19). Therefore, the whole running time is $O(n(p + K))$ which is linear in each of the variables n, p and K. It is $O(np)$ for any fixed value of K.

Algorithm 1
Algorithm B_K, which is *BLS* equipped with unflattening (for K>1) and *IE* strategies, proposed for *pNC*.

Algorithm B_K

Inputs: Matrix $D = [d_{ij}]_{n \times n}$ of shortest distances
Integer $p \in \{2, \cdots, n-1\}$
Parameter: Integer $K \in \{1, \cdots, n\}$
1 **while** Termination_Condition not met **do**
2 Initialise P with p random vertices and put the rest in Q
3 *improved* = **true**
4 **while** *improved* = **true do**
5 *improved* = **false**
6 **for** $i = 1$ to p **do**
7 **for** $j = 1$ to $n - p$ **do**
8 P_1 = solution obtained by replacing $P[i]$ with $Q[j]$
9 **if** $h_K(P_1) < h_K(P)$ **then** //the use of h_K implements unflattening strategy
10 swap $P[i]$ and $Q[j]$ //move from P to P_1
11 *improved* = **true**
12 **else if** $h_K(P_1) = h_K(P)$ **then** //the else branch implements *IE* strategy
13 $P = P_1$ and update Q
14 **end if**
15 **end for**
16 **end for**
17 **end while**
18 **end for**
19 **return** best solution found in all iterations

Algorithm 2
The algorithm used to calculate the heuristic value $h_K(P)$ for a given solution P and a value of K.

Algorithm h_K

Input: Candidate solution P
Parameter: Integer $K \in \{1, \cdots, n\}$
1: **for** each $v_i \in P$ **do**
2: //calculate f_i
3: $f_i = \min_{\substack{v_k \in P \\ k \neq i}} \{d_{ik}\}$
4: **end for**
5: **for** each $v_i \in Q$ **do**
6: //calculate f_i
7: *minPrimary* $= f_i = \infty$
8: **for** each $v_j \in P$ **do**
9: **if** $d_{ij} <$ *minPrimary* **or** ($d_{ij} =$ *minPrimary* **and** $d_{ij} + f_j < f_i$) **then**
10: *minPrimary* $= d_{ij}$
11: $f_i = d_{ij} + f_j$
12: **end if**
13: **end for**
14: **end for**
15: $f^{(1)}$ to $f^{(K)}$ = the first to the kth largest values among $f_i, v_i \in V$
16: *hValue* $= f^{(1)}$
17: **for** $r = 2$ to K **do**
18: *hValue* $= c \times$ *hValue* $+ f^{(r)}$
19: **end for**
20: **return** *hValue*

Algorithm 3
Algorithm B_K-*αNpC*, with unflattening (for$K>1$) and *IE* strategies, proposed for *αNpC*.

Algorithm B_K-αNpC

Inputs: Matrix $D = [d_{ij}]_{n\times n}$ of shortest distances
Integer $p \in \{2, \cdots, n-1\}$
Parameters: Integers $\alpha \in \{1, \cdots, p\}$ and $K \in \{1, \cdots, n\}$
1 **while** Termination_Condition not met **do**
2 Initialise P with p random vertices and update data structures accordingly
3 TB = {} //initilaize the tabu list
4 $flg\_moved = true$
5 $ls\_best\_h = h_K\text{-}\alpha NpC(P)$
6 **while** $flg\_moved = true$ **do**
7 SCL = {} //SCL is the set of candidate pairs for swap
8 v_c = a random member of C
9 $w = F^{\alpha}_{v_c}$
10 $inx = N^{-1}_{v_c,w}$
11 **for** each $t \in \{1, \cdots, inx\}$ **do** //in a random order
12 $v_j = N_{v_c,t}$
13 **if** $d_{cj} < D^{\alpha}_{v_c}$ **then**
14 **for** $v_i \in P$ **do** //in a random order
15 $P_1 = P \cup \{v_j\} \backslash \{v_i\}$
16 **if** $h_K\text{-}\alpha NpC(P_1) < h_K\text{-}\alpha NpC(P)$ **or** ($h_K\text{-}\alpha NpC(P_1) = h_K\text{-}\alpha NpC(P)$ **and** $(v_j, v_i) \notin$ TB) **then**
17 $P = P_1$ and update data structures //move
18 **if** $h_K\text{-}\alpha NpC(P) < ls\_best\_h$ **then**
19 $ls\_best\_h = h_K\text{-}\alpha NpC(P)$
20 TB = {}
21 **else**
22 TB = TB $\cup (v_j, v_i)$
23 **end if**
24 continue with next iteration of **while** loop (line 6)
25 **else**
26 update SCL with (v_j, v_i) if it is not tabued and there is no better candidate in SCL
27 **end if**
28 **end for**
29 **end if**
30 **end for**
31 **if** SCL $\neq$ {} **then**
32 (v_j, v_i) = select a pair from SCL randomly
33 $P = P \cup \{v_j\} \backslash \{v_i\}$ and update data structures //move
34 TB = TB $\cup (v_j, v_i)$
35 **else**
36 $flg\_moved = false$
37 **end if**
38 **end while**
39 **end while**
40 **return** best solution found in all iterations

3.2. Proposed algorithm for αNpC

Algorithm 3 presents the proposed metaheuristic for *αNpC*. It is called B_K-*αNpC*, with two parameters α and K, and A_K-*αNpC* is used to refer to the same algorithm but without the *IE* strategy. It uses similar (but not identical) notations used in the literature of the *pC* problem (Pullan, 2008; Mladenović et al., 2003). N_v, where $v \in V$, is the list of all vertices sorted ascendingly by their distances from v, with ties broken arbitrarily. $N_{v,i}$, $i = 1, \cdots, n$, is the vertex at index i in N_v. Another list N^{-1}_v is used to keep the index of each vertex u in N_v, i.e. $N^{-1}_{v,u} = i \Leftrightarrow N_{v,i} = u$. These data structures are static, i.e. fixed for a given problem instance, whereas the following data structures are dynamic and change with solution P. F^1_v is the first facility center (i.e. that with the smallest index) in N_v. Therefore, F^1_v is a nearest facility to v. (Note that F^1_v is defined for every vertex $v \in V$ and not only clients). v is said to be *assigned* to F^1_v as its designated facility and D^1_v is used to denote their distance. These notations are generalised to $F^r_v, r = 1, \cdots, \alpha$, to denote the rth facility in N_v and D^r_v to denote its distance to v. The set of *critical* vertices, i.e. those whose costs are equal to the current objective value, is denoted by C.

The algorithm uses the heuristic function given in (1), already used for the *pNC* problem. Recall from Section 2 that the vertex cost $f_i(P), i = 1, \cdots, n$, is defined differently for different problems. It is not hard to see that this heuristic function is also f-consistent for *αNpC* (even if a smaller coefficient $c > d_{max}$ would be used). To avoid confusion with the function h_K presented in Algorithm 2, a different name h_K-*αNpC* is used to denote the function invoked in B_K-*αNpC* to calculate heuristic values. They only differ in the calculation of the vertex costs $f_i, i = 1, \cdots, n$. In particular, h_K-*αNpC* calculates the cost f_i of a client vertex $v_i, i = 1, \cdots, n$, as its distance to the αth facility in N_{v_i}. This calculation takes $O(\alpha n/p)$ average time (because p out of n vertices in N_{v_i} are facilities) and is performed for $q = |Q|$ client vertices. Recall from Section 2 that the cost of a facility is 0 for *αNpC*. Because the calculation of the heuristic value from the cost values $f_i, i = 1, \cdots, n$, takes $O(Kn)$ (as analysed in Section 3.1 for Algorithms 3), the average time complexity of h_K-*αNpC* is $O(\alpha qn/p + Kn)$.

As shown in Algorithm 3, B_K-*αNpC* is a random restart first-improvement local search with possible upward moves. To avoid cycling in the search space, it also employs tabu strategies. The main **while** loop (lines 1–39) of the algorithm runs until its termination condition (in line 1) is met. Once it terminates, the best solution found will be returned (line 40). At each iteration, it starts with a random solution P (line 2), resets the tabu list denoted as TB (line 3), and initialises two temporary variables *flg_moved* and *ls_best_h* (lines 4–5). The former is used in the condition of the main local search loop (line 6), which runs until no move is performed. The latter records the best (smallest) heuristic value obtained during the local search. At each iteration of the local search (lines 6–38), the algorithm creates an empty list SCL (for *Swap Candidate List*), which is used to keep the list of the "best" moves among those rejected (by the **if** condition in line 16) but not forbidden by the tabu mechanism. For such a move, it keeps the pair $(v_j, v_i) \in Q \times P$ in SCL (line 26), where v_j and v_i are the client and facility vertices, respectively, that will be swapped if the move is performed. If all the moves are rejected, then a pair (if any) in SCL will be selected randomly (line 32), the corresponding move is performed (line 33), and the swapped pair is marked as tabu (line 34). In case all the rejected moves are forbidden by the tabu mechanism, SCL will remain empty and *flg_moved* will be reset (line 36) to terminate the local search loop.

The local search does not explore all neighbours. It excludes neighbours that cannot reduce the cost of a critical vertex. This requirement is implemented by selecting a (random) critical vertex v_c (line 8) and choosing the new facility among clients v_j such that $d_{cj} < D^{\alpha}_{v_c}$ (line 9–13). For such a client v_j, every facility v_i in P is considered for potential replacement (line 14). The move to the corresponding neighbour P_1 (constructed in line 15) will be accepted if it is either downward (with respect to h_K-*αNpC*) or flat but not tabued (line 16). Furthermore, if the resulting heuristic value is better than the best one found so far during the local search (verified in line 18), the tabu list will be reset (line 20); otherwise, the swapped pair (v_j, v_i) will be added to the tabu list (line 22).

3.3. Proposed algorithm for pC

Although the algorithm proposed in Section 3.2 for *αNpC* can readily be used for *pC* by setting its parameter α to 1, a more specialised algorithm is presented in this section which outperforms the existing state-of-the-art metaheuristics for this problem. This algorithm is called B_h-*pC* and the name A_h-*pC* is reserved to mean the same algorithm without the *IE* strategy. *B-pC* and *A-pC* are also used, respectively, to refer to the same algorithms but without the unflattening strategy.

B_h-*pC* uses the data structures C, N_v, N^{-1}_v, F^r_v and $D^r_v, r = 1, 2, v \in V$, already explained in Section 3.2. It uses additional data structures for more efficient implementation. For example, for each facility v, it keeps the sets $Z^r_v = \{u \in V : F^r_u = v\}$, $r = 1, 2$. It also differs from B_K-*αNpC* in other aspects including its unflattening heuristic function and tabu strategy. For brevity, its high level pseudocode is presented in Appendix A and its main features are outlined in this section.

B_h-pC uses the unflattening function previously used in Ferone et al. (2017). However, its theoretical properties are further explored here and several propositions are established for its efficient calculation. In the following, n_X, where X is a finite set, is the cardinality $|X|$ of X. B_h-pC considers the number of critical vertices to compare flat neighbours, by using the following heuristic function.

$$h(P) = n \times f(P) + n_C \tag{3}$$

Proposition 2. *The heuristic function given in* (3) *is f-consistent.*

The proof is similar to the proof of Proposition 1 and is omitted for brevity.

Let P be the current solution and $P_1 = P \cup \{v_j\} \backslash \{v_i\}$ be its neighbour obtained by swapping the pair $(v_j \in Q, v_i \in P)$ of client and facility vertices. The acceptance criteria for the potential move from P to P_1 requires the verification of the conditions $h(P_1)<h(P)$ and $h(P_1) = h(P)$ (in the same manner used in line 16 of Algorithm 3). A crucial idea in B_h-pC is to verify these conditions *without* calculating $h(P_1)$ and $f(P_1)$ by using the existing data structures and the current heuristic $h(P)$ and objective $f(P)$ values, which are only updated when a move is performed. A method is now proposed to verify these conditions in $O(n/p + n_C)$ average time. Using this method, we do not need to iterate through all the clients, which would take $\Omega(n - p)$ time.

Let C_1 be the set of critical vertices corresponding to P_1. By (3) and Proposition 2, we have:

$$h(P_1)<h(P) \quad \Leftrightarrow \quad f(P_1)<f(P) \quad \vee \quad (f(P_1) = f(P) \quad \wedge \quad n_{C_1} < n_C) \tag{4}$$

and

$$h(P_1) = h(P) \Leftrightarrow f(P_1) = f(P) \bigwedge \mathrm{n}_{\mathrm{C}_1} = \mathrm{n}_{\mathrm{C}}. \tag{5}$$

Using (4) and (5), we can verify the conditions $h(P_1)<h(P)$ and $h(P_1) = h(P)$ by verifying other conditions $f(P_1)<f(P)$ and $f(P_1) = f(P)$ and calculating n_{C_1} when $f(P_1) = f(P)$. Note that n_{C_1} is not needed if $f(P_1) \neq f(P)$. To illustrate how to verify the latter conditions without computing $f(P_1)$, the set of vertices is split into three disjoint sets V_1, V_2 and V_3. The first set V_1 is the set of vertices currently assigned to v_i as their closest facility, i.e. $V_1 = Z^1_{v_i}$. Recall that v_i is the current facility that will be replaced with v_j if the move to P_1 is accepted. V_2 is the set of critical vertices not assigned to v_i, i.e. $V_2 = C \backslash Z^1_{v_i}$. Finally, V_3 contains the remaining vertices, i.e. $V_3 = V \backslash (V_1 \cup V_2)$. Some, but not all, of these sets may be empty. As a result of the potential move from P to P_1, the only vertices whose costs may increase are members of V_1. Inspired by Pullan (2008), let m be the maximum cost of the vertices in V_1 if the move is accepted. Also, let $A = \{v_k \in V_1 : \min\{D^2_{v_k}, d_{kj}\} = f(P)\}$ and $B = \{v_k \in V_2 : d_{kj} \geqslant f(P)\}$.

Proposition 3. $f(P_1)>f(P) \quad \Leftrightarrow \quad m > f(P)$.

Proposition 4. $f(P_1)<f(P) \quad \Leftrightarrow \quad m < f(P) \ \wedge \ n_B = 0$.

The proofs of Propositions 3–4 are omitted for brevity and their simplicity.

Proposition 5. $f(P_1) = f(P) \Rightarrow n_{c_1} = n_A + n_B$.

Proof. By the assumption $f(P_1) = f(P)$ and the definition of V_3, every vertex in C_1 must be a member of V_1 or V_2. Therefore, $C_1 = Y_1 \cup Y_2$, where $Y_1 = \{v_k \in V_1 : D^1_{v_k}(P_1) = f(P)\}$, $Y_2 = \{v_k \in V_2 : D^1_{v_k}(P_1) = f(P)\}$, and $D^1_{v_k}(P_1)$ denotes the resulting cost of the node v_k if the move to P_1 is performed. However, if the move is performed, the new cost $D^1_{v_k}(P_1)$ of each vertex v_k in V_1 will be the minimum of its distances to its current secondary facility $F^2_{v_k}$ and the new facility v_j, hence $Y_1 = A$. Similarly, the new cost of each vertex in V_2 will be the minimum of its current cost $f(P)$ and its distance d_{kj} to the new facility v_j, which means $Y_2 = B$. Therefore, $C_1 = A \cup B$. Because A and B are disjoint, $n_{C_1} = n_A + n_B$. ∎

Proposition 6. *The conditions* $h(P_1)\langle h(P)$ *and* $h(P_1) = h(P)$ *are verified in* $O(n/p + n_C)$ *average time.*

Proof. The average number of vertices assigned to a facility is $O(n/p)$. Therefore, it takes $O(n/p)$ average time to iterate through Z_{v_i} and calculate m and n_A. Similarly, it takes $O(n_C)$ to iterate through C and calculate n_B. Given m and n_B, by Propositions 3–4, it takes $O(1)$ to verify the conditions $f(P_1)>f(P)$ and $f(P_1)\langle f(P)$. If neither of these conditions holds, then it takes another $O(1)$ to calculate n_{C_1} using Proposition 5. The proof is concluded by applying (4) and (5). ∎

The tabu mechanism in B_h-pC is different from that used in B_K-αNpC. Inspired by Yin et al. (2017), it is *dynamic*, i.e. a prohibited move becomes eligible once a number *tt* (for *tabu tenure*) of other moves are performed. More specifically, when a flat or upward move (with respect to *h-pC*) is performed by replacing a facility v_i with v_j, then these vertices cannot be swapped again during the next $tt(v_i, v_j)$ moves, where $tt(v_i, v_j)$ is initially 1 and is doubled every time they are swapped, capped by 0.1 $\times (n - p) \times p$. The tabu tenure $tt(v_j, v_i)$ is also set to $tt(v_i, v_j)$. In addition, for moves selected from SCL, the new facility v_j cannot be replaced immediately in the next move by any vertex.

The eligibility of potential moves is verified by the function *promising* (line 18 of Algorithm 4, Appendix A). It rejects moves that are tabued or are worse than those in SCL. In contrary to B_K-αNpC, which always selects a "best" candidate among the rejected moves, B_h-pC selects one of the best three candidates, stored in SCL, such that the selection probability of kth best candidate, $k = 1,2$, is twice as that of the $(k+1)$th one.

The maximum number of iterations of the local search's **while** loop (line 8) is initially $(n - p) \times p$ but grows each time by 10 percent (by line 47) to increase intensification (versus diversification) for more challenging instances, which require more local exploration. In addition, except for the first time, the initialisation of P (line 3) at each iteration of the main **while** loop (line 2) is not necessarily random. More specifically, it is reset to either a random solution, the best solution found in the last round of the local search, or the best solution found since the beginning, equiprobably.

Finally, it is important to note that properly implementing and updating the dynamic data structures used in B_h-pC is essential for it to achieve its best performance.

4. Experimental results

To observe the performance of the presented algorithms, they were implemented in Java. The source codes are available (as Supplementary materials). The experiments were performed on a laptop with Intel® Core(TM) i5-6200, 2.3 GHz CPU and 8 GB of RAM. The reported running times are CPU (not wall clock) times in seconds, rounded up to two decimal places. In this section, the proposed algorithms for pNC, αNpC, and pC refer to B_K, B_K-αNpC, and B_h-pC, $\alpha = 2$, $K = 3$, respectively.

4.1. Experimental results for *p*NC

The same datasets as used in Albareda-Sambola et al. (2015) and López-Sánchez et al. (2019) were used in the experiments. That is, 132 instances were obtained from the pmed1 to pmed8 data files. These data files, originally used for the purpose of the p-Median problem, are available in the OR-Library (Beasley, 1990a,b). Each data file includes a weighted graph $G = (V, E, w)$. Such a graph was first converted to a complete graph $G_c = (V, E_c, w_c)$ such that $w_c(v_i, v_j) = d_{ij}$ for each edge $(v_i, v_j) \in E_c$, where d_{ij} is the shortest distance between the vertices v_i and v_j in G. The Floyd–Warshall algorithm was used for this purpose. Then, for each pair (n, p) defined in Albareda-Sambola et al. (2015), the problem instance (G_n, p) was generated, where G_n is the subgraph of G_c induced by its first n vertices. Please see Albareda-Sambola et al. (2015) for more details on generating the instances.

Three experiments were conducted. First, algorithms A_K and B_K, $K = 1,3,5$, were evaluated to assess the impact of the unflattening and *IE*

Table 1
Impact of the unflattening and *IE* strategies on BLS.

				Without *IE* strategy							With *IE* strategy					
Instance				*BLS* (A_1)			A_3		A_5		B_1		B_3		B_5	
Filename	*n*	*p*	BKOV	t_{avg}	t_{std}	$n_{timeout}$	t_{avg}	t_{std}	t_{avg}	t_{std}	t_{avg}	t_{std}	t_{avg}	t_{std}	t_{avg}	t_{std}
pmed1	10	5	*84*	0	0	0	0	0	0	0	0	0	0	0	0	0
pmed1	20	5	*120*	0	0	0	0	0	0	0	0	0	0	0	0	0
pmed1	20	10	*95*	0	0.01	0	0	0	0	0	0	0	0	0	0	0
pmed1	30	5	*126*	0	0	0	0	0.01	0	0	0	0	0	0	0	0
pmed1	30	10	*95*	0.01	0.02	0	0	0	0	0	0	0.01	0	0	0	0
pmed1	40	5	*144*	0.14	0.11	0	1.68	1.53	1.34	0.78	0.94	0.9	1.21	1.11	1.08	0.94
pmed1	40	10	*111*	0.16	0.21	0	0.08	0.03	0.07	0.03	0.15	0.14	0.06	0.06	0.08	0.06
pmed1	40	20	*89*	0	0.01	0	0	0.01	0	0.01	0	0	0	0.01	0	0.01
pmed1	50	10	*110*	5.01	4.51	0	4.71	3.56	4.58	2.81	40.39	42.17	28.34	35.05	5.63	2.9
pmed1	50	20	*89*	1.37	1.32	0	0.06	0.03	0.06	0.05	0.11	0.09	0.02	0.02	0.06	0.06
pmed2	10	5	*121*	0	0	0	0	0	0	0	0	0	0	0	0	0
pmed2	20	5	*147*	0	0	0	0	0	0	0	0	0	0	0	0	0
pmed2	20	10	*99*	0	0	0	0	0	0	0.01	0	0.01	0	0	0.01	0.01
pmed2	30	5	*169*	0.01	0.01	0	0	0.01	0.01	0.01	0	0.01	0	0.01	0.01	0.01
pmed2	30	10	*110*	0.02	0.01	0	0	0	0	0.01	0	0	0	0.01	0	0
pmed2	40	5	*164*	0.01	0.01	0	0	0	0	0	0	0	0	0	0	0
pmed2	40	10	*112*	0.06	0.04	0	0.02	0.01	0.03	0.02	0.05	0.05	0.01	0.01	0.03	0.03
pmed2	40	20	*96*	0.05	0.04	0	0.01	0.01	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.01
pmed2	50	10	*140*	0.05	0.06	0	0.02	0.02	0.02	0.02	0.01	0.01	0.01	0.01	0.02	0.01
pmed2	50	20	*99*	0.35	0.28	0	0.03	0.03	0.03	0.02	0.04	0.04	0.02	0.01	0.02	0.02
pmed3	10	5	*77*	0	0	0	0	0	0	0	0	0	0	0	0	0
pmed3	20	5	*145*	0	0	0	0	0	0	0	0	0	0	0	0	0
pmed3	20	10	*77*	0	0	0	0	0	0	0	0	0	0	0	0	0
pmed3	30	5	*157*	0	0	0	0	0	0	0.01	0	0	0	0	0	0
pmed3	30	10	*122*	0	0.01	0	0	0	0	0	0	0	0	0	0	0.01
pmed3	40	5	*157*	0	0.01	0	0.01	0.01	0.01	0.01	0	0.01	0	0.01	0	0.01
pmed3	40	10	*105*	0.1	0.06	0	0.01	0.01	0.02	0.01	0.08	0.06	0.02	0.02	0.02	0.02
pmed3	40	20	*77*	0.09	0.07	0	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0	0.01
pmed3	50	10	*125*	0.59	0.48	0	0.05	0.06	0.05	0.03	0.05	0.04	0.02	0.03	0.02	0.01
pmed3	50	20	*87*	0.1	0.06	0	0.09	0.06	0.1	0.06	0.03	0.03	0.03	0.02	0.03	0.01
pmed4	10	5	*126*	0	0	0	0	0	0	0	0	0	0	0	0	0
pmed4	20	5	*139*	0	0	0	0	0	0	0	0	0	0	0	0	0
pmed4	20	10	*125*	0	0	0	0	0.01	0	0	0	0.01	0	0.01	0.01	0.01
pmed4	30	5	*173*	0	0	0	0	0	0	0	0	0	0	0	0	0
pmed4	30	10	*122*	0	0	0	0	0.01	0	0.01	0	0	0	0	0	0.01
pmed4	40	5	*175*	0	0	0	0	0	0	0.01	0	0	0	0	0	0
pmed4	40	10	*122*	0.16	0.21	0	0.16	0.13	0.13	0.11	0.06	0.07	0.05	0.03	0.06	0.04
pmed4	40	20	*85*	0.02	0.02	0	0	0.01	0	0.01	0.01	0.01	0.01	0.01	0	0.01
pmed4	50	10	*126*	0.02	0.01	0	0.01	0.01	0.01	0.01	0.03	0.03	0.01	0.01	0.01	0.01
pmed4	50	20	*91*	0.91	0.72	0	0.06	0.04	0.03	0.02	0.04	0.04	0.02	0.02	0.02	0.01
pmed1	60	10	*112*	0.51	0.33	0	0.13	0.1	0.18	0.2	0.2	0.15	0.09	0.06	0.14	0.17
pmed1	60	20	*91*	1.3	1.01	0	0.05	0.03	0.03	0.02	0.11	0.06	0.03	0.02	0.02	0.01
pmed1	60	30	*89*	0.09	0.07	0	0.02	0.01	0.03	0.02	0.03	0.01	0.02	0.01	0.02	0.01
pmed1	70	10	*119*	7.07	5.45	0	2.15	1.63	1.51	1.01	0.75	0.52	1.47	1.2	1.73	1.58
pmed1	70	20	*99*	0.96	0.74	0	0.14	0.14	0.07	0.05	0.07	0.07	0.09	0.06	0.08	0.07
pmed1	70	30	*73*	31.33	21	0	0.12	0.11	0.07	0.03	0.1	0.06	0.08	0.04	0.07	0.03
pmed1	80	10	*133*	1.45	1.66	0	0.25	0.15	0.21	0.16	0.11	0.09	0.2	0.2	0.11	0.09
pmed1	80	20	*105*	26.15	14.34	0	0.4	0.37	0.36	0.24	0.53	0.48	0.1	0.07	0.17	0.16
pmed1	80	30	*91*	11.85	7.76	0	0.17	0.12	0.1	0.07	0.22	0.18	0.08	0.03	0.1	0.05
pmed1	90	10	*133*	3.2	5.96	0	0.52	0.36	0.5	0.58	0.28	0.21	0.21	0.14	0.29	0.21
pmed1	90	20	*108*	80.62	38.78	8	2.13	0.97	2.04	2.24	0.97	0.95	0.34	0.23	0.2	0.12
pmed1	90	30	*91*	92.83	21.52	9	0.99	1.28	0.45	0.53	2.83	2.64	0.21	0.12	0.26	0.19
pmed1	90	50	*70*	88.13	29	8	0.15	0.07	0.09	0.02	0.51	0.51	0.15	0.08	0.15	0.11
pmed1	100	10	*133*	1.51	1.82	0	0.43	0.39	0.29	0.26	0.88	0.69	0.28	0.24	0.24	0.21
pmed1	100	20	*108*	83.1	34	8	1.99	1.19	0.93	0.87	1.47	1.44	0.31	0.25	0.22	0.11
pmed1	100	30	*97*	88.07	26.82	7	0.3	0.2	0.29	0.24	1.14	0.84	0.15	0.07	0.21	0.12
pmed1	100	50	*74*	100	0	10	0.51	0.32	0.28	0.13	0.98	0.59	0.4	0.18	0.31	0.14
pmed2	60	10	*140*	0.15	0.04	0	0.07	0.04	0.08	0.09	0.05	0.05	0.04	0.03	0.08	0.05
pmed2	60	20	*99*	31.22	25.79	0	0.43	0.23	0.23	0.14	0.5	0.42	0.07	0.04	0.06	0.04
pmed2	60	30	*96*	0.2	0.14	0	0.02	0.01	0.02	0.01	0.02	0.01	0.03	0.01	0.02	0.02
pmed2	70	10	*138*	0.5	0.41	0	0.17	0.15	0.15	0.1	0.06	0.05	0.08	0.08	0.05	0.05
pmed2	70	20	*102*	1.54	2.04	0	0.06	0.03	0.02	0.01	0.04	0.03	0.04	0.02	0.03	0.02
pmed2	70	30	*96*	0.49	0.43	0	0.04	0.03	0.04	0.01	0.04	0.02	0.05	0.02	0.04	0.03
pmed2	80	10	*138*	9.02	7.05	0	1.62	1.42	1.43	1.17	1.99	1.34	0.89	0.96	0.98	0.71
pmed2	80	20	*109*	55.32	37.85	4	0.91	1.11	0.43	0.34	0.96	1.07	0.12	0.06	0.21	0.2
pmed2	80	30	*97*	85.8	27.26	7	1.69	1.34	1.31	1.01	1.05	1.92	0.21	0.16	0.51	0.35
pmed2	90	10	*140*	0.8	0.99	0	0.13	0.15	0.06	0.04	0.3	0.28	0.11	0.07	0.12	0.09
pmed2	90	20	*109*	83.3	33.98	8	2.97	2.72	0.36	0.23	1.4	1.19	0.47	0.49	0.45	0.24
pmed2	90	30	*97*	87.24	17.84	5	0.36	0.28	0.26	0.19	0.23	0.21	0.14	0.07	0.2	0.12
pmed2	90	50	*96*	0.14	0.07	0	0.06	0.02	0.08	0.02	0.06	0.01	0.09	0.02	0.11	0.04
pmed2	100	10	*135*	7.75	6.55	0	0.52	0.64	0.47	0.52	0.08	0.09	0.17	0.09	0.19	0.13

(*continued on next page*)

Table 1 (*continued*)

				Without *IE* strategy							With *IE* strategy					
Instance				*BLS* (A_1)			A_3		A_5		B_1		B_3		B_5	
Filename	n	p	BKOV	t_{avg}	t_{std}	$n_{timeout}$	t_{avg}	t_{std}	t_{avg}	t_{std}	t_{avg}	t_{std}	t_{avg}	t_{std}	t_{avg}	t_{std}
pmed2	100	20	*109*	77.58	31.61	6	1.68	1.56	1.18	0.77	1.53	1.05	0.71	0.43	0.64	0.5
pmed2	100	30	*96*	93.01	14.94	8	0.66	0.46	0.93	1.12	1.11	0.89	0.15	0.1	0.16	0.09
pmed2	100	50	*96*	0.37	0.27	0	0.08	0.02	0.1	0.03	0.09	0.02	0.11	0.03	0.15	0.05
pmed3	60	10	*124*	11.16	11.95	0	0.61	0.57	0.73	0.65	0.71	0.52	0.55	0.46	0.43	0.56
pmed3	60	20	*97*	0.58	0.6	0	0.13	0.16	0.06	0.03	0.07	0.05	0.09	0.1	0.05	0.03
pmed3	60	30	*73*	10.03	8.36	0	0.11	0.11	0.1	0.06	0.15	0.09	0.04	0.02	0.06	0.04
pmed3	70	10	*121*	46.56	36.33	3	18.8	17.6	21.93	20.53	24.09	23.42	18.32	14.65	11.32	8.64
pmed3	70	20	*97*	4.86	4.63	0	0.17	0.1	0.08	0.05	0.1	0.06	0.05	0.04	0.05	0.03
pmed3	70	30	*82*	70.75	32.72	4	0.52	0.49	0.25	0.22	1.3	1.15	0.09	0.06	0.07	0.05
pmed3	80	10	*121*	78.08	21.58	4	25.78	23.58	34.39	31.53	34.43	22.96	27.19	16.9	28.3	31.5
pmed3	80	20	*93*	39.69	24.84	1	0.49	0.47	0.5	0.41	0.64	0.32	0.09	0.05	0.09	0.04
pmed3	80	30	*86*	38.43	26.94	1	0.16	0.08	0.07	0.03	0.35	0.36	0.08	0.04	0.07	0.02
pmed3	90	10	*148*	0.81	0.91	0	0.31	0.24	0.18	0.22	0.11	0.08	0.13	0.1	0.13	0.08
pmed3	90	20	*105*	38.88	38.31	1	1.28	0.8	0.38	0.23	1.51	1.25	0.47	0.44	0.23	0.11
pmed3	90	30	*93*	33.74	35.23	1	0.22	0.22	0.17	0.1	0.25	0.16	0.09	0.03	0.1	0.04
pmed3	90	50	*93*	0.12	0.06	0	0.06	0.03	0.08	0.03	0.06	0.01	0.1	0.03	0.08	0.01
pmed3	100	10	*151*	1.18	0.92	0	0.7	0.81	0.76	0.87	2.13	2.33	0.81	0.75	0.68	0.74
pmed3	100	20	*113*	54.56	37.28	2	5.28	4.2	4.88	5.48	2.57	1.72	1.04	1.18	0.95	1.01
pmed3	100	30	*93*	92.7	21.9	9	0.27	0.17	0.12	0.05	0.21	0.09	0.15	0.08	0.09	0.04
pmed3	100	50	*93*	0.25	0.24	0	0.08	0.04	0.11	0.04	0.09	0.03	0.15	0.05	0.17	0.05
pmed4	60	10	*135*	0.04	0.05	0	0.02	0.02	0.02	0.03	0.02	0.02	0.01	0.01	0.03	0.02
pmed4	60	20	*93*	0.69	0.39	0	0.03	0.02	0.05	0.04	0.06	0.04	0.02	0.01	0.03	0.02
pmed4	60	30	*79*	31.82	25.03	0	0.05	0.04	0.06	0.05	0.44	0.41	0.05	0.02	0.04	0.03
pmed4	70	10	*146*	0.44	0.33	0	0.22	0.22	0.09	0.08	0.27	0.18	0.12	0.14	0.11	0.15
pmed4	70	20	*102*	2.94	2.28	0	0.18	0.1	0.1	0.07	0.09	0.08	0.06	0.05	0.06	0.05
pmed4	70	30	*85*	1.01	0.84	0	0.04	0.02	0.04	0.01	0.14	0.07	0.06	0.04	0.04	0.03
pmed4	80	10	*146*	1.12	1.14	0	0.28	0.3	0.14	0.12	0.61	0.62	0.24	0.16	0.14	0.08
pmed4	80	20	*114*	61.29	35.85	3	10	8.76	9.46	10.66	19.51	21.51	9.49	9.51	5.08	3.68
pmed4	80	30	*91*	21.21	14.98	0	0.22	0.15	0.14	0.1	0.48	0.46	0.08	0.05	0.1	0.05
pmed4	90	10	*147*	1.09	0.89	0	0.11	0.14	0.13	0.13	0.17	0.18	0.05	0.03	0.07	0.04
pmed4	90	20	*112*	89.72	20.57	8	12.73	12.16	10.52	9.19	4.16	2.39	1.86	1.52	0.84	0.61
pmed4	90	30	*92*	100	0	10	2.01	1.37	1.44	0.73	1.66	2.36	1.03	1.15	0.52	0.2
pmed4	90	50	*82*	7.06	5.79	0	0.11	0.08	0.16	0.08	0.24	0.22	0.11	0.04	0.2	0.15
pmed4	100	10	*147*	3.38	1.81	0	0.4	0.33	0.17	0.21	0.2	0.25	0.23	0.37	0.16	0.1
pmed4	100	20	*119*	12.44	11.63	0	0.22	0.29	0.38	0.46	0.85	0.62	0.49	0.41	0.22	0.16
pmed4	100	30	*96*	100	0	10	3.95	2.84	1.35	0.88	2.62	2.03	0.83	0.74	0.53	0.33
pmed4	100	50	*82*	94.72	15.84	9	0.34	0.26	0.3	0.19	1.02	1.14	0.34	0.19	0.28	0.13
pmed6	150	20	*79*	100	0	10	2.16	1.47	1.78	1.43	3.54	3.95	0.95	0.66	0.86	0.64
pmed6	150	30	*71*	100	0	10	3.7	3.04	3.3	2.07	2.32	1.44	0.84	0.4	0.73	0.51
pmed6	150	50	*62*	78.91	38.53	7	0.7	0.28	0.43	0.12	0.97	0.56	0.48	0.1	0.34	0.05
pmed6	150	80	*56*	2.83	2.39	0	0.45	0.14	0.4	0.01	0.48	0.09	0.45	0.03	0.41	0.06
pmed6	200	20	*79*	100	0	10	9.32	6.26	4.43	5.23	39.68	20.66	3.1	2.22	4.02	3.22
pmed6	200	30	*72*	100	0	10	5.98	9.28	2.39	2.31	6.88	5.46	1.99	1.5	2.63	1.65
pmed6	200	50	*68*	100	0	10	2.81	1.67	2.44	2.21	1.98	0.92	2.24	1.14	2.2	0.94
pmed6	200	80	*54*	91.47	25.6	9	2.88	1.53	1.96	1.01	2.67	0.8	1.61	0.77	2.03	1.68
pmed7	150	20	*69*	71.58	39.03	6	0.24	0.16	0.27	0.12	0.35	0.2	0.2	0.1	0.15	0.07
pmed7	150	30	*62*	100	0	10	3.08	2.91	0.86	0.5	3.25	1.82	0.7	0.43	0.66	0.27
pmed7	150	50	*59*	63.56	41.52	5	0.43	0.16	0.51	0.26	0.4	0.11	0.49	0.21	0.82	0.3
pmed7	150	80	*59*	1.05	0.51	0	0.5	0.16	0.44	0.03	0.43	0.07	0.48	0.08	0.79	0.28
pmed7	200	20	*73*	100	0	10	1.46	1.01	0.95	0.66	14.41	19.79	0.93	0.68	0.72	0.53
pmed7	200	30	*68*	100	0	10	1.17	0.74	0.89	0.47	3.43	2.74	0.79	0.48	0.69	0.31
pmed7	200	50	*63*	100	0	10	2.09	1.45	1.46	0.51	2.16	0.78	1.68	0.64	1.63	0.52
pmed7	200	80	*52*	100	0	10	3.08	2.42	1.6	0.81	5.02	2.26	1.57	0.6	1.83	0.82
pmed8	150	20	*74*	100	0	10	7.24	4.32	3.68	4.01	14.12	9.12	1.19	1.38	1.4	1.4
pmed8	150	30	*61*	100	0	10	4.37	2.04	1.3	0.9	5.17	3.29	0.88	0.47	0.92	0.63
pmed8	150	50	*58*	86.81	27.07	8	0.68	0.46	0.6	0.32	0.66	0.24	0.58	0.16	0.77	0.39
pmed8	150	80	*58*	0.96	0.62	0	0.46	0.13	0.48	0.11	0.45	0.08	0.67	0.16	0.66	0.19
pmed8	200	20	*84*	100	0	10	1.82	1.55	1.26	0.74	8.24	6	0.56	0.29	0.85	0.49
pmed8	200	30	*77*	100	0	10	1.8	1.39	0.95	1.08	4.25	2.64	0.98	0.59	1.1	0.39
pmed8	200	50	*68*	100	0	10	2.1	1.79	1.69	0.88	2.14	1.05	1.96	0.98	1.7	0.49
pmed8	200	80	*68*	7.91	9.03	0	1.21	0.5	1.4	0.61	1.08	0.19	1.38	0.3	1.85	1.02
Avg.			**103.01**	**31.20**	**7.96**	**2.57**	**1.32**	**1.10**	**1.11**	**0.96**	**2.18**	**1.77**	**0.98**	**0.80**	**0.73**	**0.58**

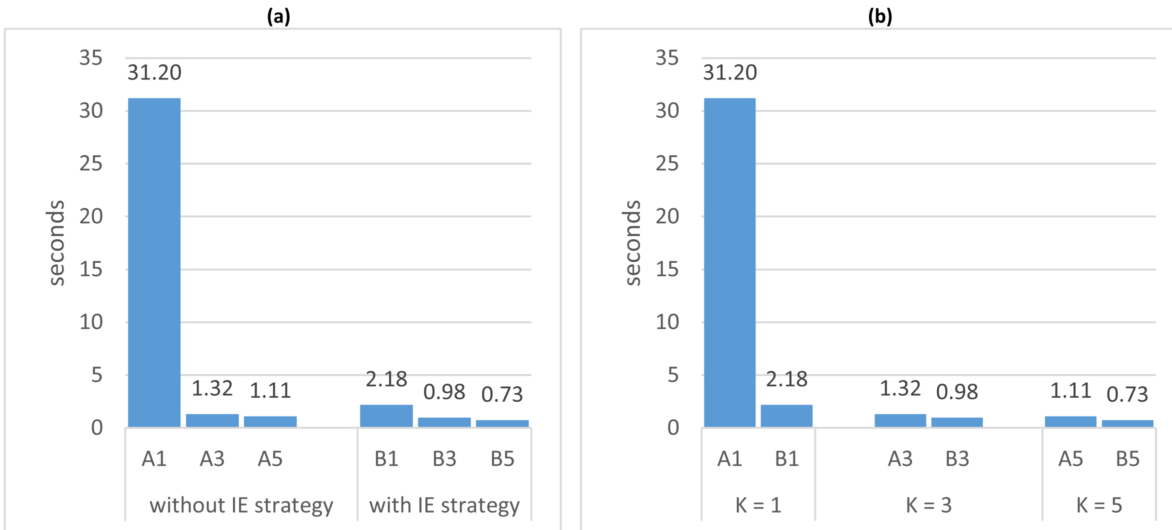


Fig. 3. (a) Impact of the unflattening strategy on the average running times of A_1 and B_1. (b) Impact of the *IE* strategy on the average running times of A_K, $K = 1,2,3$.

strategies on BLS. Then, B_3 was compared with the state-of-the-art GRASP-VNS in López-Sánchez et al. (2019). Finally, an experiment was conducted to observe the variability of the solution quality in different iterations of the local search of B_3.

4.1.1. Evaluation of the unflattening and IE strategies on BLS

The first experiment was conducted to compare the time required by algorithms A_K and B_K, $K = 1,3,5$, to achieve solutions with better or the same objective values as the Best Known Objective Values (BKOVs) previously obtained by Albareda-Sambola et al. (2015) or López-Sánchez et al. (2019). More specifically, each of these algorithms was run 10 times on each instance and for each of them but A_1 (BLS), the time spent to reach/exceed BKOV was recorded. Because A_1, as a naïve metaheuristic, required hours to achieve such an objective value for some (usually large) instances, a lower bound on its actual time was considered by applying a time limit of 100 s at each run. This lower bound was, on average, still sufficiently large to confirm the superiority of the other algorithms to BLS, as reported in Table 1. The reason for applying this time limit was to avoid (approximately) weeks of unnecessarily running of BLS.

The first three columns in Table 1 show the name of the data file (used to generate the instance) and the numbers of vertices n and centers p, respectively. The fourth column displays BKOV. For the first 40 instances, BKOVs are optimal, obtained by exact algorithms (Albareda-Sambola et al., 2015). For the remaining instances, these values were obtained by GRASP-VNS (López-Sánchez et al., 2019), which does not guarantee optimal values. Columns 5–6 calculate the average and standard deviation of the running time of BLS. Column 7 reports the number of times out of 10 where BLS failed to reach/exceed BKOV within the time limit. The subsequent columns present the average and standard deviation of the running times of the other algorithms. The last row calculates the average values over all the instances.

As can be seen in Table 1, on average, every algorithm performs significantly better than BLS. This means that the application of the unflattening and/or *IE* strategies to BLS has been successful. On average (shown in the last row), the lower bound on the actual running time of BLS (31.20 s) is more than the actual running times of the other algorithms by an order of magnitude.

The results also show that the joint application of both strategies has been more useful than their individual applications in reducing the average running time of BLS. The average running times of A_3 and B_1 are 1.32 and 2.18 s, respectively, which are reduced to 0.98 s by B_3. Similarly, the average running times of A_5 (1.11) and B_1 (2.18) are further reduced by B_5 (0.73). As depicted in Fig. 3, the average running time is reduced by applying the unflattening strategy to A_1 and B_1 (Fig. 3.a) and by applying the *IE* strategy to A_K, $K = 1,2,3$ (Fig. 3.b).

It was observed in the experiments (not reported in Table 1) that BLS, when run without a time limit, did not reach/exceed BKOV even after an hour, for several instances, e.g. pmed6 with $n = 150$, $p = 20$, while B_3 and B_5 did so within a few second. Obviously, such a basic local search is not expected to perform satisfactorily for NP-hard optimisation problems. However, the important observation is that the application of the unflattening and/or *IE* strategies significantly improves it without much modifications (via lines 9 and/or 12 in Algorithm 1). The following section shows that these small modifications convert the naïve BLS to an algorithm, B_3, that outperforms the state-of-the-art hybrid GRASP-VNS metaheuristic.

4.1.2. Comparison of the proposed algorithm with the state-of-the-art GRASP-VNS

Algorithm B_3 was run on each instance for the same amount of (scaled) time as reported for GRASP-VNS in López-Sánchez et al. (2019). For each run, the best objective value and the time spent to find it were recorded. It was run 10 times per instance and the average and the standard deviation of these values were calculated, reported in Table 2.

The first four columns in Table 2 are the same as those in Table 1. Columns 5–6 present the results of GRASP-VNS, which was run once per instance in López-Sánchez et al. (2019). In particular, Column 5 presents the objective values obtained by GRASP-VNS as reported in López-Sánchez et al. (2019). These objective values equal BKOVs (Column 4) except for two instances, namely pmed1, $n = 50$, $p = 10$ and pmed2, $n = 10$, $p = 5$. For these instances, BKOVs, obtained by exact algorithms of Albareda-Sambola et al. (2015), are 110 and 121, but those obtained by GRASP-VNS of López-Sánchez et al. (2019) are 111 and 128, respectively. Column 6 presents the running times of GRASP-VNS reported in López-Sánchez et al. (2019) divided by 2.60, which is the ratio of the average mark of the CPU used here to that used in López-Sánchez et al. (2019), obtained from CPU Benchmarks (2022). The remaining columns report the results for B_3. In particular, the next four columns report the average and standard deviation of the objective values obtained and the times spent to find them. The subsequent column reports the number of runs (out of 10) where the obtained objective value was better or the same as that reported in Column 5 for GRASP-VNS. The last column reports the best objective value found in all 10 runs.

As can be seen in Table 2, the average objective values obtained by B_3 are better than those obtained by GRASP-VNS in 35 cases and worse in 11 cases (shown in bold). It can also be seen that the relative performance of B_3 is usually better for larger instances in Table 2, which

Table 2
Comparison of B_3 and the hybrid GRASP-VNS metaheuristic.

Instance				GRASP-VNS		B_3					
Filename	n	p	BKOV	OV	t_{scaled}	OV_{avg}	OV_{std}	t_{avg}	t_{std}	$n_{achieved}$	OV_{best}
pmed1	10	5	84	84	0.01	84	0	0	0	10	84
pmed1	20	5	120	120	0.02	120	0	0	0	10	120
pmed1	20	10	95	95	0.05	95	0	0	0	10	95
pmed1	30	5	126	126	0.03	126	0	0	0	10	126
pmed1	30	10	95	95	0.11	95	0	0	0	10	95
pmed1	40	5	144	**144**	0.09	147.1	2.022	0.02	0.02	2	144
pmed1	40	10	111	**111**	0.22	111.4	0.8	0.04	0.03	8	111
pmed1	40	20	89	89	0.64	89	0	0	0	10	89
pmed1	50	10	110	111	0.36	111	0	0.03	0.03	10	111
pmed1	50	20	89	89	1.24	89	0	0.04	0.03	10	89
pmed2	10	5	121	128	0.00	**121**	0	0	0	10	121
pmed2	20	5	147	147	0.02	147	0	0	0	10	147
pmed2	20	10	99	99	0.04	99	0	0	0	10	99
pmed2	30	5	169	169	0.04	169	0	0	0	10	169
pmed2	30	10	110	110	0.14	110	0	0	0	10	110
pmed2	40	5	164	164	0.08	164	0	0	0	10	164
pmed2	40	10	112	112	0.22	112	0	0.01	0.01	10	112
pmed2	40	20	96	96	0.65	96	0	0.01	0.01	10	96
pmed2	50	10	140	140	0.32	140	0	0	0.01	10	140
pmed2	50	20	99	99	1.26	99	0	0.01	0.01	10	99
pmed3	10	5	77	77	0.00	77	0	0	0	10	77
pmed3	20	5	145	145	0.02	145	0	0	0	10	145
pmed3	20	10	77	77	0.04	77	0	0	0	10	77
pmed3	30	5	157	157	0.04	157	0	0	0	10	157
pmed3	30	10	122	122	0.12	122	0	0	0	10	122
pmed3	40	5	157	157	0.08	157	0	0	0	10	157
pmed3	40	10	105	105	0.22	105	0	0.02	0.02	10	105
pmed3	40	20	77	77	0.65	77	0	0.01	0.01	10	77
pmed3	50	10	125	125	0.33	125	0	0.02	0.02	10	125
pmed3	50	20	87	87	1.07	87	0	0.04	0.05	10	87
pmed4	10	5	126	126	0.00	126	0	0	0	10	126
pmed4	20	5	139	139	0.02	139	0	0	0	10	139
pmed4	20	10	125	125	0.04	125	0	0	0	10	125
pmed4	30	5	173	173	0.03	173	0	0	0	10	173
pmed4	30	10	122	122	0.11	122	0	0	0	10	122
pmed4	40	5	175	175	0.07	175	0	0	0	10	175
pmed4	40	10	122	**122**	0.21	122.1	0.3	0.07	0.05	9	122
pmed4	40	20	85	85	0.67	85	0	0	0	10	85
pmed4	50	10	126	126	0.36	126	0	0	0.01	10	126
pmed4	50	20	91	91	1.31	91	0	0.02	0.02	10	91
pmed1	60	10	112	112	0.48	112	0	0.11	0.09	10	112
pmed1	60	20	91	91	1.95	**89**	0	0.13	0.14	10	89*
pmed1	60	30	89	89	4.42	89	0	0	0	10	89
pmed1	70	10	119	**119**	0.71	124	5	0.28	0.21	5	119
pmed1	70	20	99	99	2.98	99	0	0.07	0.05	10	99
pmed1	70	30	73	73	7.25	73	0	0.06	0.05	10	73
pmed1	80	10	133	133	0.94	**131.4**	1.96	0.25	0.25	10	129*
pmed1	80	20	105	105	4.48	**102**	0	1.63	1.21	10	102*
pmed1	80	30	91	91	11.22	**85**	0	0.72	0.71	10	85*
pmed1	90	10	133	133	1.28	133	0	0.21	0.13	10	133
pmed1	90	20	108	108	5.79	**107**	0	1.03	1.15	10	107*
pmed1	90	30	91	91	14.73	**87**	0	0.53	0.35	10	87*
pmed1	90	50	70	70	39.89	70	0	0.06	0.05	10	70
pmed1	100	10	133	133	1.68	133	0	0.13	0.12	10	133
pmed1	100	20	108	108	7.43	108	0	0.33	0.21	10	108
pmed1	100	30	97	97	19.81	**93.6**	0.663	9.7	6.33	10	93*
pmed1	100	50	74	74	55.12	**70**	0	0.53	0.36	10	70*
pmed2	60	10	140	140	0.55	140	0	0.03	0.02	10	140
pmed2	60	20	99	99	2.35	99	0	0.06	0.04	10	99
pmed2	60	30	96	96	5.23	96	0	0.01	0.01	10	96
pmed2	70	10	138	138	0.76	138	0	0.06	0.06	10	138
pmed2	70	20	102	102	3.40	102	0	0.04	0.04	10	102
pmed2	70	30	96	96	7.94	96	0	0.02	0.02	10	96
pmed2	80	10	138	**138**	1.07	140	2.449	0.37	0.24	6	138
pmed2	80	20	109	109	4.77	109	0	0.17	0.1	10	109
pmed2	80	30	97	97	11.86	97	0	0.24	0.15	10	97
pmed2	90	10	140	140	1.37	**138.8**	0.98	0.27	0.22	10	138*
pmed2	90	20	109	109	6.35	**108.7**	0.458	1.15	0.94	10	108*
pmed2	90	30	97	97	15.97	**96**	0	0.1	0.07	10	96*
pmed2	90	50	96	96	40.22	96	0	0.01	0.01	10	96
pmed2	100	10	135	135	1.53	135	0	0.08	0.05	10	135
pmed2	100	20	109	109	7.23	**107.3**	0.458	2.27	2.15	10	107*

(continued on next page)

Table 2 (*continued*)

Instance			**BKOV**	**GRASP-VNS**		**B_3**					
Filename	**n**	**p**		**OV**	**t_{scaled}**	**OV_{avg}**	**OV_{std}**	**t_{avg}**	**t_{std}**	**$n_{achieved}$**	**OV_{best}**
pmed2	100	30	96	96	19.33	96	0	0.2	0.24	10	96
pmed2	100	50	96	96	55.82	96	0	0.02	0.03	10	96
pmed3	60	10	124	**124**	0.57	124.3	0.458	0.22	0.19	7	124
pmed3	60	20	97	97	1.98	97	0	0.04	0.06	10	97
pmed3	60	30	73	73	4.17	73	0	0.05	0.03	10	73
pmed3	70	10	121	**121**	0.83	127	0	0.04	0.04	0	121
pmed3	70	20	97	97	3.33	97	0	0.08	0.06	10	97
pmed3	70	30	82	82	7.53	82	0	0.11	0.08	10	82
pmed3	80	10	121	**121**	1.11	125.8	1.99	0.22	0.25	1	121
pmed3	80	20	93	93	4.63	93	0	0.13	0.08	10	93
pmed3	80	30	86	86	11.51	**84**	0	0.25	0.26	10	84*
pmed3	90	10	148	148	1.08	148	0	0.15	0.07	10	148
pmed3	90	20	105	105	4.85	105	0	0.48	0.53	10	105
pmed3	90	30	93	93	12.89	93	0	0.05	0.03	10	93
pmed3	90	50	93	93	34.58	93	0	0.02	0.01	10	93
pmed3	100	10	151	**151**	1.37	151.3	0.458	0.58	0.34	7	151
pmed3	100	20	113	113	6.59	**111.1**	1.814	1.5	1.44	10	109*
pmed3	100	30	93	93	17.28	93	0	0.09	0.05	10	93
pmed3	100	50	93	93	47.99	93	0	0.01	0.01	10	93
pmed4	60	10	135	135	0.56	**134.5**	0.5	0.13	0.16	10	134*
pmed4	60	20	93	93	2.09	93	0	0.02	0.02	10	93
pmed4	60	30	79	79	5.34	79	0	0.02	0.02	10	79
pmed4	70	10	146	146	0.78	146	0	0.1	0.08	10	146
pmed4	70	20	102	102	3.24	102	0	0.05	0.03	10	102
pmed4	70	30	85	85	7.37	85	0	0.03	0.03	10	85
pmed4	80	10	146	**146**	1.08	146.1	0.3	0.12	0.11	9	146
pmed4	80	20	114	**114**	4.63	114.5	0.5	1.69	1.77	5	114
pmed4	80	30	91	91	11.30	**90**	0	0.27	0.26	10	90*
pmed4	90	10	147	147	1.40	147	0	0.09	0.08	10	147
pmed4	90	20	112	112	6.33	112	0	1.8	1.3	10	112
pmed4	90	30	92	92	16.74	92	0	0.7	0.75	10	92
pmed4	90	50	82	82	39.59	82	0	0.1	0.11	10	82
pmed4	100	10	147	147	1.81	147	0	0.2	0.13	10	147
pmed4	100	20	119	119	8.42	**118**	0	0.52	0.38	10	118*
pmed4	100	30	96	96	22.24	96	0	0.5	0.34	10	96
pmed4	100	50	82	82	59.30	82	0	0.15	0.09	10	82
pmed6	150	20	79	79	13.02	**77**	0	4.18	3.74	10	77*
pmed6	150	30	71	71	29.69	**65.9**	0.3	11.56	5.6	10	65*
pmed6	150	50	62	62	76.99	**56**	0	0.51	0.38	10	56*
pmed6	150	80	56	56	184.23	56	0	0.16	0.29	10	56
pmed6	200	20	79	79	19.15	**77.1**	0.3	6.63	3.68	10	77*
pmed6	200	30	72	72	57.80	**66.9**	0.539	30.12	15.92	10	66*
pmed6	200	50	68	68	189.83	**49.2**	0.4	77.89	46.24	10	49*
pmed6	200	80	54	54	442.74	**49**	0	0.57	0.49	10	49*
pmed7	150	20	69	69	8.81	**67.4**	0.49	2.09	2.24	10	67*
pmed7	150	30	62	62	25.42	**59**	0	0.78	0.89	10	59*
pmed7	150	50	59	59	79.36	59	0	0.13	0.09	10	59
pmed7	150	80	59	59	159.63	59	0	0.03	0.01	10	59
pmed7	200	20	73	73	16.94	**68**	0	5.14	4.52	10	68*
pmed7	200	30	68	68	49.71	**60.9**	0.943	18.85	11.07	10	60*
pmed7	200	50	63	63	194.24	**50**	0	5.04	5.25	10	50*
pmed7	200	80	52	52	466.79	**46**	0	1.33	0.96	10	46*
pmed8	150	20	74	74	9.14	**73.1**	0.7	2.42	2.38	10	72*
pmed8	150	30	61	61	24.94	**58**	0	1.71	0.8	10	58*
pmed8	150	50	58	58	74.84	58	0	0.23	0.17	10	58
pmed8	150	80	58	58	165.50	58	0	0.05	0.04	10	58
pmed8	200	20	84	84	15.85	**82.5**	1.285	5.88	5.8	10	81*
pmed8	200	30	77	77	46.98	**70**	0.632	15.96	13.62	10	69*
pmed8	200	50	68	68	151.90	68	0	0.79	0.94	10	68
pmed8	200	80	68	68	438.48	68	0	0.16	0.17	10	68
Avg.			**103.01**	**103.07**	**27.83**	**102.22**	**0.20**	**1.70**	**1.14**	**9.61**	**101.94**

suggests that it is more scalable than GRASP-VNS. Using one-tailed paired t-test, the null hypothesis that the average objective value obtained by B_3 (Column 6) is not better than that of GRASP-VNS (Column 5) is rejected with a p-value < 0.0003.

During the experiment, B_3 found objective values better than BKOVs for 34 instances, indicated by asterisks in the last column of Table 2.

4.1.3. *Solution quality in different iterations*

This section reports the results of an experiment conducted to observe the quality of solutions found in different iterations of the proposed algorithm, B_3.

First, the termination condition of its main loop (line 1 of Algorithm 1) was adjusted to iterate 40 times. Then, it was run on each instance with probability 0.3 (so, approximately 30 % of the instances were

Table 3
Solution quality in different iterations.

No.	Filename	n	p	BOV	#Iterations BOV found	%Average deviation
1	pmed1	20	5	120	27	4.90
2	pmed1	40	20	89	39	0.31
3	pmed1	50	10	111	9	9.91
4	pmed1	50	20	89	8	5.65
5	pmed2	20	5	147	18	7.98
6	pmed2	30	5	171	16	3.27
7	pmed2	40	10	112	5	21.23
8	pmed3	10	5	77	**40**	**0**
9	pmed3	30	5	157	10	5.56
10	pmed3	30	10	122	17	7.56
11	pmed3	40	5	157	4	8.68
12	pmed3	40	20	77	15	19.38
13	pmed4	30	5	173	7	7.47
14	pmed4	30	10	122	22	2.01
15	pmed4	40	20	85	34	4.09
16	pmed4	50	20	91	20	7.66
17	pmed1	60	20	89	5	8.06
18	pmed1	60	30	89	31	3.71
19	pmed1	70	20	99	15	6.52
20	pmed1	90	50	70	23	15.68
21	pmed1	100	30	94	**1**	8.16
22	pmed2	60	10	140	3	5.36
23	pmed2	70	30	96	29	1.17
24	pmed2	80	10	143	2	6.43
25	pmed2	90	30	96	17	2.76
26	pmed2	90	50	96	**40**	**0**
27	pmed2	100	10	135	4	7.07
28	pmed3	60	10	124	**1**	13.99
29	pmed3	60	20	97	9	13.76
30	pmed3	70	20	97	19	4.12
31	pmed3	80	20	93	14	8.20
32	pmed3	100	10	152	4	7.93
33	pmed3	100	30	93	31	2.18
34	pmed4	60	10	135	15	3.50
35	pmed4	80	10	147	7	5.80
36	pmed4	100	30	96	5	11.43
37	pmed4	100	50	82	14	3.69
38	pmed6	150	20	77	2	7.60
39	pmed6	200	20	78	3	5.38
40	pmed6	200	30	67	**1**	15.41
41	pmed6	200	50	50	6	**29.70**
42	pmed8	150	50	58	35	4.35
43	pmed8	150	80	58	36	3.62
44	pmed8	200	30	72	11	10.28
45	pmed8	200	50	68	30	5.74
Avg.					**15.64**	**7.49**

used). The objective value obtained at each iteration and the best objective value (BOV) among them were recorded. Then, he number of iterations out of 40 where BOV was found was counted and the average deviation percentage was calculated as $\frac{1}{40}\sum_{i=1}^{40}\left(\frac{\mathrm{OV}_i-\mathrm{BOV}}{\mathrm{BOV}}\right)\times 100$, where OV_i is the objective value obtained in iteration i. The results are presented in Table 3.

Table 3 indicates that the number of iterations where BOV is found and the average deviation of the solution quality from BOV can vary significantly from one instance to another. The minimum deviation (0) is observed for instance no. 8 and 26, where BOV was found in all 40 iterations. The maximum deviation (29.70) is reported for instance no. 41 where BOV was found in 6 iterations. The minimum number of iterations BOV was found is 1 (for three instances). As shown in the last row, the average deviation percentage over all the instances is 7.49 %, and the average number of iterations that obtained BOV is 15.64, i.e. 39 %.

4.2. Experimental results for αNpC

The algorithms A_K-*αNpC* and B_K-*αNpC*, for $\alpha = 2$ and $K = 1, 3$, were run on the standard pmed1 to pmed40 instances in the OR-Library (Beasley, 1990a,b). Each pmed data file includes a weighted graph and the (default) number of required facility centers p. Each graph was converted to a complete graph using the Floyd–Warshall algorithm, and the resulting distance matrix together with the specified number of centers p was used as the input instance.

Because of no known prior objective values for these instances of the *αNpC* problem, initial experiments were performed using the algorithms with various settings and longer running time to obtain relatively high quality BKOVs. The conditions of the **while** loops (lines 1 and 6 in Algorithm 3) were adjusted such that each algorithm terminated upon achieving/exceeding BKOV or exceeding a time limit of 10,000 s (which did not happen). The algorithms were run 10 times per instance and the average and standard deviation of the objective value were observed.

The results are presented in Table 4. The first three columns show the instance filename and the numbers of vertices and facilities. Column 4 presents the target value. The subsequent columns report the average (t_{avg}) and the standard deviation (t_{std}) of the running times of the algorithms.

As indicated by Table 4, both the unflattening and *IE* strategies improved the base algorithm A_1-*αNpC*. The average running time (shown in the last row) for algorithms A_3-*αNpC* and B_1-*αNpC* are 37.98 and 28.47 s, respectively, whereas it is 70.55 s for A_1-*αNpC*. The algorithm B_3-*αNpC* further reduces the average running time to 5.09 s, while its standard deviation (6.07) is less than those of the other three algorithms.

4.3. Experimental results for pC

Finally, experiments were performed to evaluate the algorithms *A-pC*, A_h-*pC*, *B-pC*, B_h-*pC*, and the GRASP-PR state-of-the-art metaheuristic of Yin et al. (2017). Because the CPU used in Yin et al. (2017) was not specified (except its clock speed of 3.4 GHz), the running time reported in Yin et al. (2017) could not be scaled (using CPU benchmarks) to compare with the running time of the proposed algorithm which was run on a different CPU (Intel® Core$^{(\mathrm{TM})}$ i5-6200, 2.3 GHz). The source code of GRASP-PR used in Yin et al. (2017) was not available either. Therefore, GRASP-PR was implemented (in Java), and it was further optimised, improving its speed by an order of magnitude. However, the running time of this optimized implementation of GRASP-PR was observed to be, on average, significantly longer than that reported in Yin et al. (2017), which could be explained by the difference in the CPUs used here and there. Therefore, to show that the superiority of the proposed algorithm to GRASP-PR is not because of using a different implementation, the running times reported in Yin et al. (2017) are still used in this section as the baseline to evaluate the proposed algorithm.

The same problem instances as used in Yin et al. (2017) were used, namely 40 pmed instances (Beasley, 1990a,b) and 84 TSP instances (Reinelt, 1991; Universität Heidelberg, 2018). The pmed instances are pmed1 to pmed40 already used in Section 4.2, and the TSP instances are pr226, pr264, pr299, pr439, pcb442, kroA200, kroB200, lin318, gr202, d493, d657, u1060, rl1323, and u1817 used with various numbers of facilities p. Following the convention in Yin et al. (2017), these instances are split into two groups of 44 (small) and 40 (large) instances.

The algorithms were adjusted to run until finding solutions with better or the same objective values as BKOVs (Elloumi et al., 2004; Pullan, 2008; Yin et al., 2017; Liu et al., 2020) or exceeding a time limit of 10,000 s (which did not happen except for the last large TSP instance). The reason for terminating the algorithms upon achieving BKOVs is that these values are either optimum or very challenging to improve. In particular, these values are known optimum for all 40 pmed instances and 30 large (u1060 and ul817) TSP instances (Elloumi et al., 2004; Liu et al., 2020). For the remaining (44 small and 10 large TSP) instances, their rounded values are also optimum (Pullan, 2008; Yin et al., 2017), but the actual decimal values have not (yet) been proved so, though challenging to improve.

Some experimental settings not explicitly specified in Yin et al. (2017) are based on Pullan (2008), because Yin et al. (2017) used the results in Pullan (2008) to evaluate the GRASP-PR algorithm. In

Table 4
Comparison of the algorithms for the *αNpC* problem on the `pmed` instances.

Instance				A_1-αNpC		A_3-αNpC		B_1-αNpC		B_3-αNpC	
Filename	n	p	BKOV	t_{avg}	t_{std}	t_{avg}	t_{std}	t_{avg}	t_{std}	t_{avg}	t_{std}
pmed1	100	5	*150*	0.01	0.01	0.01	0.01	0.02	0.01	0.01	0.01
pmed2	100	10	*121*	0.23	0.13	0.12	0.06	0.14	0.11	0.20	0.15
pmed3	100	10	*121*	0.30	0.18	0.50	0.58	0.29	0.21	0.26	0.35
pmed4	100	20	*97*	13.91	11.67	5.09	4.30	7.91	5.91	8.19	8.90
pmed5	100	33	*63*	0.05	0.03	0.03	0.03	0.02	0.01	0.02	0.02
pmed6	200	5	*99*	0.07	0.04	0.03	0.03	0.05	0.04	0.03	0.03
pmed7	200	10	*80*	0.12	0.10	0.09	0.09	0.04	0.02	0.09	0.07
pmed8	200	20	*70*	0.09	0.09	0.15	0.16	0.03	0.03	0.03	0.03
pmed9	200	40	*49*	1.47	1.07	0.57	0.42	0.20	0.13	0.73	0.71
pmed10	200	67	*28*	0.81	0.80	0.69	0.36	0.22	0.13	0.62	0.36
pmed11	300	5	*68*	0.01	0.01	0.01	0.01	0.00	0.01	0.00	0.01
pmed12	300	10	*60*	1.54	1.18	0.95	0.82	0.50	0.67	0.27	0.32
pmed13	300	30	*43*	7.85	6.78	1.38	1.29	1.60	1.54	2.07	1.72
pmed14	300	60	*34*	2.85	3.34	1.72	1.66	1.17	1.26	0.93	1.35
pmed15	300	100	*23*	28.21	28.80	15.05	6.92	7.54	4.23	6.86	7.12
pmed16	400	5	*52*	0.47	0.51	0.27	0.26	0.31	0.24	0.24	0.27
pmed17	400	10	*45*	0.14	0.04	0.07	0.05	0.04	0.03	0.04	0.03
pmed18	400	40	*34*	271.85	315.90	63.81	57.36	81.64	62.22	17.76	33.81
pmed19	400	80	*25*	2.50	1.11	2.11	0.82	0.13	0.10	0.17	0.09
pmed20	400	133	*19*	7.99	3.12	10.76	6.96	1.48	1.60	1.24	1.35
pmed21	500	5	*45*	14.97	20.71	8.58	6.80	6.78	8.31	1.20	1.09
pmed22	500	10	*44*	2.22	1.96	0.60	0.51	0.75	0.61	0.42	0.43
pmed23	500	50	*27*	206.68	291.96	44.48	19.73	95.31	88.62	11.18	8.77
pmed24	500	100	*20*	6.83	2.52	6.80	2.26	0.29	0.21	0.54	0.59
pmed25	500	167	*15*	410.91	236.79	217.29	251.20	474.10	504.19	33.68	40.18
pmed26	600	5	*43*	0.52	0.30	0.73	0.64	0.45	0.21	0.24	0.19
pmed27	600	10	*36*	0.84	0.50	0.30	0.32	0.14	0.09	0.09	0.06
pmed28	600	60	*22*	55.78	41.25	15.78	9.16	1.26	0.79	0.59	0.27
pmed29	600	120	*17*	10.49	2.81	13.97	7.61	0.21	0.08	0.32	0.09
pmed30	600	200	*13*	57.23	28.83	118.54	78.51	4.42	3.28	2.89	2.13
pmed31	700	5	*34*	0.32	0.16	0.10	0.06	0.10	0.06	0.05	0.03
pmed32	700	10	*33*	6.27	4.84	0.99	0.81	0.32	0.26	0.21	0.17
pmed33	700	70	*19*	444.58	676.33	236.96	129.34	89.58	117.77	10.28	11.14
pmed34	700	140	*14*	913.70	1014.86	484.27	552.41	355.59	419.45	97.77	117.88
pmed35	800	5	*34*	2.92	2.70	0.52	0.60	0.43	0.27	0.54	0.82
pmed36	800	10	*31*	5.82	9.45	1.67	1.54	1.47	0.77	0.25	0.12
pmed37	800	80	*19*	23.47	7.57	20.99	7.34	0.14	0.06	0.12	0.07
pmed38	900	5	*33*	0.22	0.18	0.07	0.10	0.07	0.06	0.09	0.09
pmed39	900	10	*26*	4.45	3.27	0.97	0.94	0.44	0.36	0.18	0.18
pmed40	900	90	*16*	313.36	209.11	242.25	167.38	3.72	1.82	3.04	1.64
Avg.				**70.55**	**73.28**	**37.98**	**32.99**	**28.47**	**30.64**	**5.09**	**6.07**

particular, the reported running time excludes the time needed to read the input file and populate the static data structures and, if BKOV is not achieved in a run, the reported time is that at which the smallest objective value was found. The number of runs per instance is also adopted from Pullan (2008), which is 100 for pmed and small TSP instances and 10 for large TSP instances. This is because, the latter instances require, on average, much more time.

Table 5 presents the results for the pmed instances. The first three columns show the instance filename and the numbers of vertices and facilities. The next column shows the target objective values (OPT), which are optimum (Pullan, 2008; Yin et al., 2017). Columns 5–6 present the average (t_{avg}) and the standard deviation (t_{std}) of the running time of GRASP-PR as reported in Yin et al. (2017). The subsequent columns report these statistics for the implemented GRASP-PR (ImpGRASP-PR) and the other algorithms.

Table 5 indicates that the application of the *IE* and unflattening strategies are useful because all three algorithms *A_h-pC, B-pC, and B_h-pC* are faster, on average, than *A-pC*. The average running times of these three algorithms are 0.16, 0.21 and 0.13 s, respectively, whereas it is 0.25 s for *A-pC*. The results also suggest that the application of both strategies is better than the application of each of them individually for this dataset. Compared to GRASP-PR, all four algorithms perform worse, on average. However, the performance of the proposed algorithm *B_h-pC,* with *both* the *IE* and unflattening strategies, is competitive to that of GRASP-PR. In particular, *B_h-pC* has a slightly worse average running time (0.13 versus 0.12) but a better average standard deviation (0.15 versus 0.24). The average running time of the implemented GRASP-PR is 0.37 s, which is longer than that of GRASP-PR in Yin et al. (2017) by approximately-three times..

Similarly, Table 6 presents the results for the small `TSP` dataset. The results indicate no significant difference among *A-pC, A_h-pC, B-pC* and *B_h-pC,* while all of them outperform the GRASP-PR state-of-the-art. The average running times of all three algorithms *A_h-pC, B-pC* and *B_h-pC,* which use *IE* and/or unflattening strategies, are the same (0.23) but are better than those of *A-pC* (0.26) and GRASP-PR (1.02) by 12 % and 77 %, respectively. The average running time of the implemented GRASP-PR is 7.49 s, which is longer than that of GRASP-PR in Yin et al. (2017) by more than seven times.

The results for large TSP instances are reported in Table 7. Because some of these instances require significantly more running time, only *B_h-pC,* with both *IE* and unflattening strategies, was run on these instances. The first three columns show the instance filename and the numbers of vertices and facilities. The fourth column is the target value BKOV. The subsequent four columns report the results obtained by GRASP-PR, namely the best objective value, the average and the standard deviation of the running time and the average deviation percentage. The last four columns present the same information for *B_h-pC.*

Table 7 shows that *B_h-pC* outperforms the GRASP-PR state-of-the-art with respect to both solution quality and average running time. However, GRASP-PR has a better average standard deviation. *B_h-pC* finds solutions of higher quality for two instances of u1817 (with $p = 80$, 150). The average running time of *B_h-pC* is less than that of GRASP-PR by

Table 5
Comparison of the algorithms for the *pC* problem on `pmed` instances.

Instance				GRASP-PR		ImpGRASP-PR		*A-pC*		*A$_h$-pC*		*B-pC*		*B$_h$-pC*	
Filename	**n**	***p***	**OPT**	t_{avg}	t_{std}	t_{avg}	t_{std}	t_{avg}	t_{std}	t_{avg}	t_{std}	t_{avg}	t_{std}	t_{avg}	t_{std}
pmed1	100	5	127	0.00	0.00	0.01	0.07	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
pmed2	100	10	98	0.00	0.00	0.01	0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01
pmed3	100	10	93	0.01	0.02	0.34	0.34	0.02	0.02	0.02	0.02	0.03	0.03	0.02	0.02
pmed4	100	20	74	0.00	0.01	0.05	0.07	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
pmed5	100	33	48	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
pmed6	200	5	84	0.00	0.00	0.01	0.01	0.00	0.01	0.00	0.01	0.01	0.01	0.00	0.01
pmed7	200	10	64	0.00	0.00	0.04	0.11	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.01
pmed8	200	20	55	0.00	0.01	0.02	0.03	0.00	0.01	0.00	0.01	0.01	0.01	0.00	0.01
pmed9	200	40	37	0.00	0.00	0.01	0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01
pmed10	200	67	20	0.00	0.00	0.00	0.01	0.01	0.01	0.00	0.01	0.01	0.01	0.01	0.01
pmed11	300	5	59	0.01	0.03	1.85	2.46	0.00	0.01	0.00	0.01	0.09	0.13	0.08	0.10
pmed12	300	10	51	0.00	0.00	0.01	0.01	0.00	0.01	0.00	0.01	0.01	0.01	0.01	0.01
pmed13	300	30	36	0.00	0.02	0.12	0.19	0.05	0.08	0.03	0.03	0.05	0.05	0.03	0.03
pmed14	300	60	26	0.00	0.00	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
pmed15	300	100	18	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.01
pmed16	400	5	47	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.01
pmed17	400	10	39	0.00	0.02	0.01	0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01
pmed18	400	40	28	0.01	0.07	0.23	0.43	0.05	0.05	0.04	0.06	0.05	0.05	0.07	0.07
pmed19	400	80	18	0.57	0.93	1.07	1.20	0.85	1.06	0.53	0.46	0.55	0.79	0.61	0.49
pmed20	400	133	13	0.03	0.07	0.95	1.17	0.77	1.06	1.02	1.58	0.53	0.78	0.92	1.44
pmed21	500	5	40	0.01	0.01	0.01	0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01
pmed22	500	10	38	0.24	1.02	0.42	0.68	0.07	0.06	0.03	0.03	0.05	0.05	0.03	0.03
pmed23	500	50	22	0.14	1.43	1.89	1.87	0.49	0.66	0.40	0.44	0.39	0.38	0.40	0.41
pmed24	500	100	15	0.04	0.03	0.09	0.10	0.14	0.11	0.09	0.09	0.06	0.04	0.06	0.04
pmed25	500	167	11	0.02	0.02	0.24	0.58	0.17	0.25	0.30	0.46	0.10	0.13	0.08	0.08
pmed26	600	5	38	0.00	0.00	0.02	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.01
pmed27	600	10	32	0.00	0.04	0.04	0.18	0.01	0.01	0.00	0.01	0.01	0.01	0.00	0.01
pmed28	600	60	18	0.04	0.05	0.11	0.17	0.14	0.13	0.06	0.06	0.11	0.09	0.13	0.17
pmed29	600	120	13	0.03	0.01	0.03	0.06	0.08	0.04	0.04	0.07	0.03	0.02	0.04	0.05
pmed30	600	200	9	0.35	0.23	1.73	1.78	1.05	1.00	2.62	2.48	0.55	0.58	0.94	1.03
pmed31	700	5	30	0.00	0.00	0.02	0.01	0.01	0.01	0.00	0.01	0.01	0.01	0.00	0.01
pmed32	700	10	29	0.05	0.31	0.08	0.08	0.06	0.05	0.01	0.01	0.09	0.08	0.02	0.02
pmed33	700	70	15	1.56	2.56	2.80	2.69	3.37	3.73	0.77	0.85	2.64	2.61	0.80	0.98
pmed34	700	140	11	0.04	0.01	0.02	0.01	0.09	0.03	0.01	0.01	0.02	0.01	0.02	0.02
pmed35	800	5	30	0.04	0.10	0.09	0.10	0.01	0.01	0.01	0.01	0.02	0.01	0.01	0.01
pmed36	800	10	27	0.55	0.68	1.29	2.22	0.11	0.10	0.03	0.03	0.14	0.13	0.05	0.06
pmed37	800	80	15	0.08	0.13	0.15	0.24	0.33	0.22	0.12	0.18	0.13	0.11	0.18	0.19
pmed38	900	5	29	0.03	0.00	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
pmed39	900	10	23	0.77	1.44	0.63	0.62	1.75	1.88	0.08	0.07	2.28	2.38	0.25	0.25
pmed40	900	90	13	0.25	0.27	0.24	0.34	0.52	0.24	0.25	0.25	0.23	0.14	0.34	0.34
Avg.				**0.12**	**0.24**	**0.37**	**0.45**	**0.25**	**0.27**	**0.16**	**0.18**	**0.21**	**0.22**	**0.13**	**0.15**

over 3 times. Its standard deviation is more than that of GRASP-PR by 40 %, approximately. The average deviation percentage of *B$_h$-pC* is 0.15 which is significantly more than that of GRASP-PR (0.01). Indeed, *B$_h$-pC* has zero deviation for all but the last instance. However, *B$_h$-pC* achieved the optimal objective value (91.6) for that instance, whereas GRASP-PR failed to do so. GRASP-PR could not achieve a zero deviation for another instance, rl1323 with $p = 80$, for which *B$_h$-pC* achieved zero deviation. The average running time of GRASP-PR is better than that of *B$_h$-pC* for 10 instances and worse for 30. The implemented GRASP-PR was not run on these instances as it would require several weeks of continuous running.

The average running time of the proposed algorithm, *B$_h$-pC*, over all the 124 instances was 81.41 s whereas it was 320.06 for GRASP-PR in Yin et al. (2017). Using one-tailed paired *t*-test, over all 124 instances, the null hypothesis that the average running time of the proposed algorithm (Column 15 in Tables 5–6 and Column 10 in Table 7) is not less than that of GRASP-PR (Column 5 in Tables 5–6 and Column 6 in Table 7) is rejected with a p-value < 0.01.

Further experiments were conducted to compare the performance of *A-pC, A$_h$-pC, B-pC* and *B$_h$-pC* on the large TSP instances but with a shorter time limit of 60 s (not reported). However, as was the case for the small TSP instances, no significant difference was observed among them.

Overall, the unflattening and *IE* strategies were observed to be influential on pmed but not on TSP instances. A hypothesis to explain this difference is that the percentage of the flat neighbours is relatively low for the TSP dataset. Recall that the whole purpose of the unflattening and *IE* strategies is to exploit the flat subspaces.

To test this hypothesis, another experiment was performed. The rationale behind this experiment was the observation that no flat move would ever be performed in the algorithms (with or without the unflattening and *IE* strategies) if all the distance values $d_{ij}, i, j = 1, \cdots, n$, were distinct. This is because the algorithm always selects, as the new facility, a client vertex that *reduces* the cost of a critical vertex. Therefore, the percentage of duplicate distance values should be related to the percentage of the flat neighbours in the search space. Based on this observation, the percentage of duplicate distance values for an instance was used as an indication of its flat subspace size. In particular, for each instance, 1000 pairs of distance values $(d_{i_1j_1}, d_{i_2j_2})$ were randomly selected and the percentage of cases where $d_{i_1j_1} = d_{i_2j_2}$ was calculated. Then, the average percentage values for the pmed, small TSP and large TSP datasets were calculated. The results were 2.31 %, 0.05 % and 0.02 %, respectively. These figures suggest that the relative number of potential flat moves for the pmed dataset is significantly greater than that of the TSP dataset, which is an explanation for the different levels of impact of the unflattening and *IE* strategies on these datasets.

4.4. Impact of the IE strategy on diversification

The algorithms *A-pC* and *B-pC* were applied to the first (pmed1, $n = 100$, $p = 5$) and the last (pmed40, $n = 900$, $p = 90$) instances of the pmed dataset to compare the diversity of the points they visited during their search. A visited point is a neighbour of the current point in the

Table 6
Comparison of the algorithms for the *pC* problem on small TSP instances.

Instance				GRASP-PR		ImpGRASP-PR		*A-pC*		A_h-*pC*		*B-pC*		B_h-*pC*	
Filename	**n**	***p***	**BKOV**	t_{avg}	t_{std}	t_{avg}	t_{std}	t_{avg}	t_{std}	t_{avg}	t_{std}	t_{avg}	t_{std}	t_{avg}	t_{std}
pr226	226	40	650.00	0.00	0.00	0.01	0.01	0.02	0.02	0.01	0.01	0.01	0.01	0.01	0.01
pr226	226	20	1365.65	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.01
pr226	226	10	2326.48	0.01	0.02	0.57	0.74	0.24	0.38	0.24	0.37	0.28	0.52	0.37	0.75
pr226	226	5	3720.55	0.00	0.02	0.73	1.24	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
pr264	264	40	316.23	0.00	0.01	0.00	0.01	0.01	0.02	0.00	0.01	0.00	0.01	0.00	0.01
pr264	264	20	514.78	0.00	0.01	0.10	0.26	0.01	0.01	0.00	0.01	0.01	0.02	0.01	0.01
pr264	264	10	850.00	0.00	0.00	0.15	0.41	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.02
pr264	264	5	1610.12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
pr299	299	40	355.32	0.05	0.08	0.27	0.28	0.21	0.23	0.20	0.21	0.18	0.19	0.21	0.18
pr299	299	20	559.02	0.14	0.15	1.39	1.68	0.02	0.02	0.02	0.02	0.04	0.04	0.03	0.04
pr299	299	10	888.84	0.30	0.33	4.10	4.78	0.03	0.04	0.02	0.03	0.02	0.03	0.03	0.03
pr299	299	5	1336.27	0.05	0.02	1.73	2.55	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
pr439	439	40	671.75	0.27	0.34	2.62	3.08	0.55	1.07	0.42	1.28	0.68	1.83	0.35	0.51
pr439	439	20	1185.59	0.02	0.05	0.26	0.61	0.02	0.07	0.01	0.02	0.01	0.02	0.01	0.01
pr439	439	10	1971.83	0.01	0.09	0.51	1.78	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01
pr439	439	5	3196.58	0.04	0.06	0.51	2.04	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
pcb442	442	40	316.23	0.02	0.04	0.06	0.22	0.02	0.04	0.01	0.02	0.01	0.01	0.01	0.03
pcb442	442	20	447.21	0.23	0.11	0.39	0.50	0.09	0.07	0.06	0.05	0.09	0.10	0.08	0.07
pcb442	442	10	670.82	0.06	0.26	1.22	2.28	0.02	0.02	0.01	0.01	0.01	0.02	0.01	0.01
pcb442	442	5	1024.74	0.08	0.18	5.91	7.96	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.03
kroA200	200	40	258.26	0.61	0.09	0.79	0.93	0.75	0.86	0.66	0.64	0.80	0.85	0.71	0.70
kroA200	200	20	389.31	0.05	0.09	0.50	0.66	0.03	0.03	0.02	0.03	0.02	0.02	0.03	0.08
kroA200	200	10	598.82	0.11	0.23	1.83	1.60	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.01
kroA203	200	5	911.41	0.02	0.11	1.45	1.67	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
kroB200	200	40	253.24	0.01	0.01	0.42	0.47	1.68	1.92	1.76	1.84	1.73	2.03	1.36	1.46
kroB200	200	20	382.28	0.00	0.00	0.24	0.35	0.02	0.03	0.02	0.02	0.02	0.04	0.01	0.03
kroB200	200	10	582.10	0.00	0.03	0.17	0.27	0.01	0.01	0.00	0.01	0.01	0.01	0.00	0.01
kroB200	200	5	897.67	0.00	0.00	0.04	0.19	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
lin318	318	40	315.92	0.00	0.00	0.05	0.10	0.05	0.12	0.05	0.11	0.03	0.06	0.04	0.07
lin318	318	20	496.45	0.86	1.34	47.25	48.09	0.14	0.14	0.15	0.11	0.16	0.15	0.18	0.15
lin318	318	10	743.21	0.06	0.38	7.98	9.35	0.07	0.06	0.05	0.05	0.07	0.06	0.05	0.05
lin318	318	5	1101.34	0.05	0.12	2.82	3.83	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01
gr202	202	40	2.97	0.00	0.02	0.05	0.06	0.03	0.02	0.02	0.02	0.02	0.02	0.02	0.02
gr202	202	20	5.57	0.00	0.00	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
gr202	202	10	9.33	0.00	0.00	0.03	0.03	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01
gr202	202	5	19.38	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
d493	493	40	206.02	6.95	5.96	34.30	33.15	1.83	1.64	1.80	1.51	1.46	1.39	1.71	1.63
d493	493	20	312.74	2.58	8.23	24.79	23.93	0.66	0.90	0.61	0.54	0.46	0.37	0.53	0.56
d493	493	10	458.30	1.27	1.21	13.29	16.77	0.07	0.07	0.06	0.06	0.06	0.05	0.05	0.06
d493	493	5	752.91	3.49	2.74	4.88	8.78	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
d657	657	40	249.52	22.34	7.36	138.43	118.24	4.38	4.30	3.30	3.39	3.17	2.86	3.71	3.81
d657	657	20	374.70	0.46	4.55	4.02	4.10	0.13	0.10	0.16	0.16	0.15	0.10	0.13	0.11
d657	657	10	574.74	4.03	3.31	25.67	35.10	0.27	0.33	0.26	0.24	0.34	0.35	0.30	0.30
d657	657	5	880.91	0.61	1.27	0.16	0.42	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.01
Avg.				**1.02**	**0.88**	**7.49**	**7.69**	**0.26**	**0.29**	**0.23**	**0.25**	**0.23**	**0.26**	**0.23**	**0.25**

search space examined as the destination of the next potential move, irrespective of whether the move is accepted.

To measure the diversity of the visited points, their average Hamming distance was used, where the Hamming distance of two points is defined here as their symmetric difference. The minimum Hamming distance is 0, when the points are identical, and the maximum is *p*, when they are disjoint.

Example 3. Let $\mathrm{S} = \{\mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3\}$ be a multiset of visited points, where $\mathrm{P}_1 = \{1,2,3\}$, $\mathrm{P}_2 = \{1,2,4\}$, and $\mathrm{P}_3 = \{2,4,5\}$ and $\mathrm{HD}(.,.)$ be the Hamming distance function. Then the diversity of S is calculated as $\mathrm{Diversity}(\mathrm{S}) = \frac{1}{3}(\mathrm{HD}(\mathrm{P}_1,\mathrm{P}_2) + \mathrm{HD}(\mathrm{P}_1,\mathrm{P}_3) + \mathrm{HD}(\mathrm{P}_2,\mathrm{P}_3)) = \frac{1}{3}(2+4+2) \approx 2.67$.

Several metrics (including a slightly different Hamming distance function) were used in the literature to measure the diversity of numerical solutions (Morrison and De Jong, 2001; Salleh et al., 2018; Morales-Castañeda et al., 2020). However, the solutions here are categorical (not numerical) because a number in a solution *P* here (e.g. in Example 3) represents a vertex. A slightly different Hamming distance function was also used in Pinheiro et al. (2005) to compare genome sequences, among categorical applications to name.

Before running the algorithms, line 3 of Algorithm 4 (Appendix A) was changed from stochastic (randomised) to deterministic so that both algorithms would start with the same initial solution (which was simply the set of the first *p* vertices as the facility centers). Please note that, even starting with the same initial solution, the algorithms may still traverse different paths in the search space because of using the *IE* strategy in *B-pC* and other stochastic parts of the algorithm (lines 14, 17, 39 in Algorithm 4).

First, for each instance, *B-pC* was run until it found BKOV and, at each visited point P_i, the diversity of the visited points (from the beginning) was calculated. In addition, the total number of visited points n_1 was recorded. Then, *A-pC* was run until it terminated (finding BKOV) or visited n_1 points and the diversity values were calculated in the same way. If it terminated earlier, visiting $n_2 < n_1$ points, then only the first n_2 diversity values calculated for *B-pC* were kept.

Fig. 4.a to 4.c show the results of three independent runs of the algorithms on the first instance. These results indicate no significant difference between the diversity of the points visited by these algorithms. In all three cases, the diversity values for both algorithms start from 0 and end at 5 ± 1. This means the *IE* strategy has not been effective in diversifying the visited pints for the algorithm *B-pC*. A potential explanation could be that the algorithm did not find enough number of flat neighbours to move to along its search path in the search space. This

Table 7
Comparison of Bh-pC and GRASP-PR for the *pC* problem on large TSP instances.

Instance				GRASP-PR				B_h-*pC*			
Filename	*n*	*p*	BKOV	OV_{best}	t_{avg}	t_{std}	OV_{dev}	OV_{best}	t_{avg}	t_{std}	OV_{dev}
u1060	1060	10	2273.08	2273.08	1.31	24.11	0	2273.08	0.13	0.10	0
u1060	1060	20	1580.8	1580.8	14.88	85.67	0	1580.80	1.30	0.80	0
u1060	1060	30	1207.77	1207.77	3.19	30.43	0	1207.77	0.32	0.15	0
u1060	1060	40	1020.56	1020.56	3.26	41.32	0	1020.56	0.32	0.28	0
u1060	1060	50	904.92	904.92	218.85	104.87	0	904.92	16.74	7.80	0
u1060	1060	60	781.17	781.17	7.75	89.55	0	781.17	5.17	5.54	0
u1060	1060	70	710.75	710.75	116.91	12.4	0	710.75	169.85	265.21	0
u1060	1060	80	652.16	652.16	316.57	38.53	0	652.16	105.68	73.58	0
u1060	1060	90	607.87	607.87	7.09	20.78	0	607.87	13.17	12.06	0
u1060	1060	100	570.01	570.01	19.04	8.29	0	570.01	26.20	39.68	0
u1060	1060	110	538.84	538.84	66.46	57.33	0	538.84	59.99	43.92	0
u1060	1060	120	510.27	510.27	397.85	18.7	0	510.27	203.01	210.82	0
u1060	1060	130	499.65	499.65	58.18	83.08	0	499.65	78.50	65.82	0
u1060	1060	140	452.46	452.46	127.39	55.86	0	452.46	40.16	26.90	0
u1060	1060	150	447.01	447.01	4.37	11.5	0	447.01	32.39	27.88	0
rl1323	1323	10	3077.3	3077.3	38.02	342.59	0	3077.30	0.66	0.69	0
rl1323	1323	20	2016.4	2016.4	104.89	129.04	0	2016.40	2.69	1.88	0
rl1323	1323	30	1631.5	1631.5	169.47	473.51	0	1631.50	36.09	25.12	0
rl1323	1323	40	1352.36	1352.36	21.9	184.9	0	1352.36	3.62	2.65	0
rl1323	1323	50	1187.27	1187.27	119.63	110.75	0	1187.27	5.97	4.19	0
rl1323	1323	60	1063.01	1063.01	4190.92	394.07	0	1063.01	182.66	160.52	0
rl1323	1323	70	971.93	971.93	6287.04	129.23	0	971.93	2022.28	1406.41	0
rl1323	1323	80	895.06	895.06	5265.81	384.5	0.09	895.06	800.70	537.44	0
rl1323	1323	90	832	832	776.23	453.29	0	832.00	39.58	47.85	0
rl1323	1323	100	787	789.7	2010.67	225.68	0	789.70	1180.79	940.54	0
ul817	1817	10	457.91	457.91	604.53	87.25	0	457.91	2.75	2.46	0
ul817	1817	20	309.01	309.01	4068.06	398.87	0	309.01	29.64	28.64	0
ul817	1817	30	240.99	240.99	1239.97	20.43	0	240.99	32.74	27.69	0
ul817	1817	40	209.45	209.45	308.29	67.34	0	209.45	51.68	47.06	0
ul817	1817	50	184.91	184.91	471.94	234.9	0	184.91	24.74	34.13	0
ul817	1817	60	162.64	162.64	469.43	55.98	0	162.64	8.89	8.77	0
ul817	1817	70	148.11	148.11	19.66	87.36	0	148.11	4.37	3.90	0
ul817	1817	80	136.77	136.8	12.42	110.4	0	**136.78**	176.87	132.62	0
ul817	1817	90	129.51	129.51	3859.05	287.61	0	129.51	65.53	49.77	0
ul817	1817	100	126.99	126.99	2.35	39.51	0	126.99	7.70	7.14	0
ul817	1817	110	109.25	109.25	6954.89	434.03	0	109.25	1777.02	2495.07	0
ul817	1817	120	107.76	107.76	5.25	19.34	0	107.76	4.94	3.75	0
ul817	1817	130	104.73	107.75	7.04	13.45	0	107.75	10.49	7.67	0
ul817	1817	140	101.6	101.6	30.95	137.31	0	101.60	177.20	175.45	0
ul817	1817	150	91.6	92.44	1236.55	23.97	0.16	**91.60**	2677.31	2619.35	6.1
Avg.			**729.81**	**729.97**	**990.95**	**138.19**	**0.01**	**729.95**	**252.00**	**238.78**	**0.15**

explanation is consistent with the observation of the number of flat moves made by *B-pC*, which were 2, 0, and 0, for Fig. 4.a–c, respectively.

Similarly, the results of three independent runs of the algorithm on the last instance are presented in Fig. 4.d–f. In contrary to the previous results for the first instance, significant increase in the diversity of the points visited by *B-pC* is now evident. The diversity value for *B-pC* increases sharply at the beginning and stays well above that of *A-pC* steadily. The number of flat moves performed by *B-pC* were 1270, 2179 and 2986, respectively, for Fig. 4.d–f. This means the search space corresponding to the last pmed instance has significantly more flat neighbours than that of the first instance.

Overall, these results suggest that the *IE* strategy can be beneficial in diversifying the visited subspace, but its impact also depends on the structure of the search space, among other potential factors.

4.5. Impact of the unflattening strategy on lengths of paths to improved solutions

To observe how the unflattening strategy can potentially reduce the expected length of the path to an improved solution in the search space, the algorithms *A-pC* and A_h*-pC* were applied to the pmed40 instance with $n = 900$ and $p = 90$. Recall that A_h*-pC* is the same as *A-pC* except that it also uses the unflattening strategy. The *Reset*() function (line 3 in Algorithm 4, Appendix A) was modified to yield the same initial point for both algorithms. The initial point was simply the set of the first *p* vertices as the facility centers.

Fig. 5.a shows the objective values of the first 1000 points visited by the algorithms. Both algorithms start with the same initial point with objective value 33. However, A_h*-pC* performs more downward moves, improving the objective value more frequently, than *A-pC*. Fig. 5.b focuses on point no. 3–7, where both algorithms visit points of the same objective values at point no. 3–5 but A_h*-pC* takes over and achieves an improved objective value at point no. 6.

The question here is why A_h*-pC* takes over and achieves an improved objective value (not solely an improved heuristic value) while both algorithms behave the same with respect to downward moves. That is, their only difference lies in the acceptance of flat moves (with improved heuristic values), which do not change the objective values, so what is the relation between such flat moves and downward moves?

To answer this question, let us first look at Fig. 5.c, which further focuses on point no. 4–6. To better understand what is happening, the vertical axis does not show the objective values but the normalised heuristic values, defined as $\hat{h}(P) = h(P)/n = f(n) + nC(P)/n$, where $nC(P)$ is the number of critical vertices for a point P. The reason for showing normalised heuristic values in Fig. 5.c, as opposed to the objective values, is that such a value encodes both the objective value as its integer part and the number of critical vertices as its decimal part product n. This is because the objective value is an integer (for any pmed instance) hence the decimal part is only due to the term nC/n. The decimal parts shown in Fig. 5.c are approximate, using three decimal places.

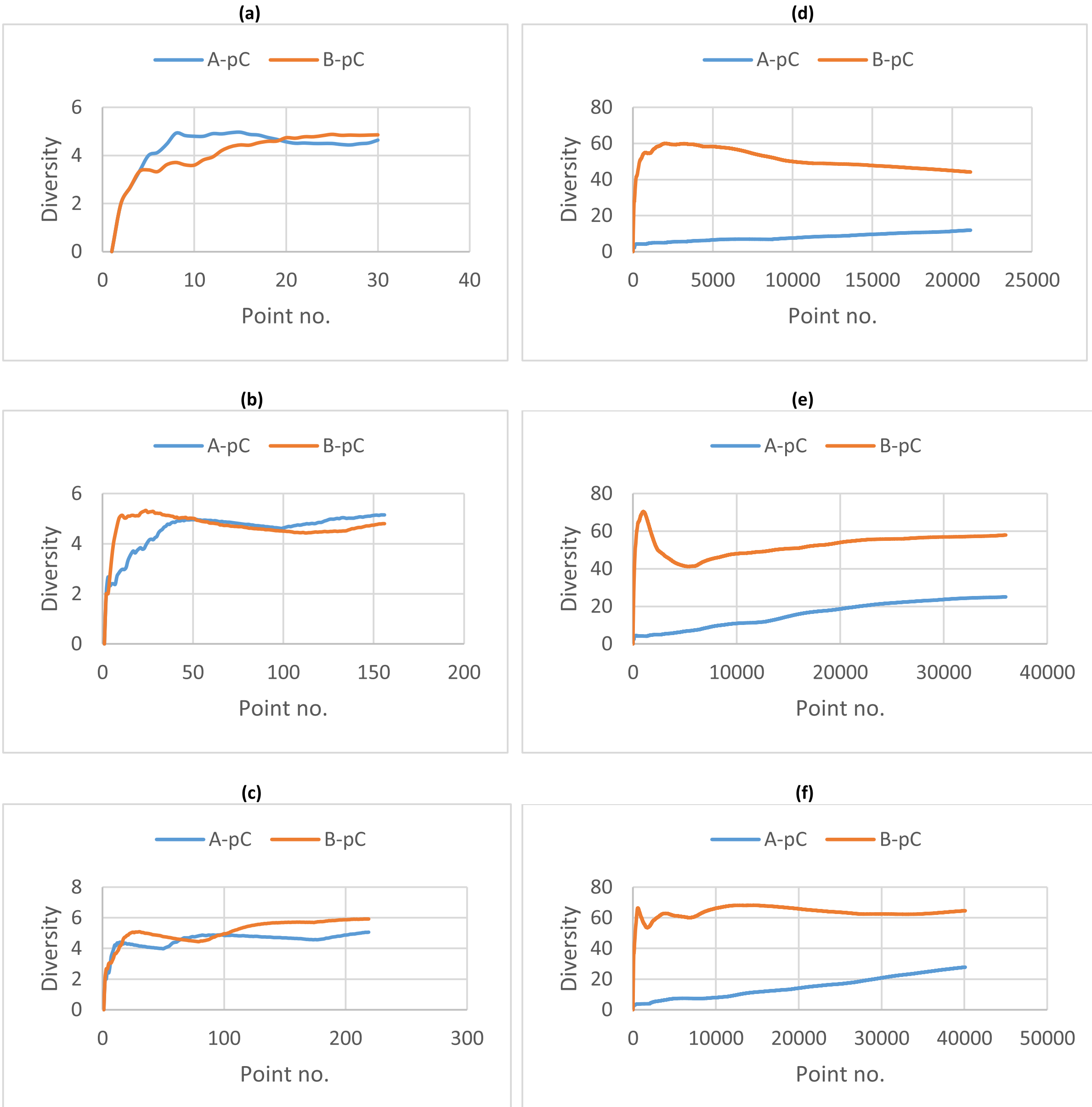


Fig. 4. Diversity of the visited points by algorithms *A-pC* and *B-pC* during their three runs on the first (a-c) and the last (d-f) pmed instances. At each point no. *i* on the horizontal axis, the diversity of the points from the beginning to point no. *i* is shown for each algorithm.

Fig. 5.c shows that, at point no. 4, the algorithms visit point(s) with the same normalised heuristic value 23.002. This implies that, for both algorithms, the objective value is 23 and the number of critical vertices is $nC \approx n \times 0.002 \approx 2$. Recall that $n = 900$ and the decimal values shown in Fig. 5.c are approximate. Next, at point no. 5, the algorithms visit different points still with the same objective value 23 but with different heuristic values. More specifically, the number of critical vertices for A_h-pC is now 1 whereas it is still 2 for *A-pC*. The flat move performed by A_h-pC has not (yet) been useful with respect to the objective value, although it has reduced the number of critical vertices. However, in the next point (point no. 6), A_h-pC achieves an improved objective value 22 (shown in Fig. 5.b but not in Fig. 5.c because of its vertical axis' scale) whereas A_h-pC stays with the same objective value 23. This scenario illustrates a case where a flat move that improves the heuristic value results in a subsequent downward move. But, is this by chance or because of the unflattening strategy?

Another experiment was performed to help with this question. A number of random points, called *basepoints*, were selected to examine their neighbours. More specifically, the algorithm *A-pC* was run on this pmed instance until 1000 basepoints were selected. To avoid selecting the first 1000 visited points only, each visited point was selected by probability 0.001 (so the algorithm visited, approximately, one million points). For each selected basepoint, the number of critical vertices, the number of neighbours with the same objective value and the number of neighbours with the same objective value but a better heuristic value were recorded. Fig. 6 presents the total number of neighbours with improved objective value and the total number of neighbours with the same objective but improved heuristic values grouped by the number of critical vertices nC of the basepoints for $2 \leqslant nC \leqslant 10$. The full results are presented in Table 8.

As can be seen in Fig. 6, for basepoints with >4 critical vertices, there is no neighbour with a better objective value, i.e. there is no downward move, whereas there are numerous flat moves that reduce the number of critical vertices. This mean, when at a point with $nC > 4$, the algorithm A_h-pC uses available flat moves to reduce the number of critical vertices whereas *A-pC* iterates through all the neighbours and founds no

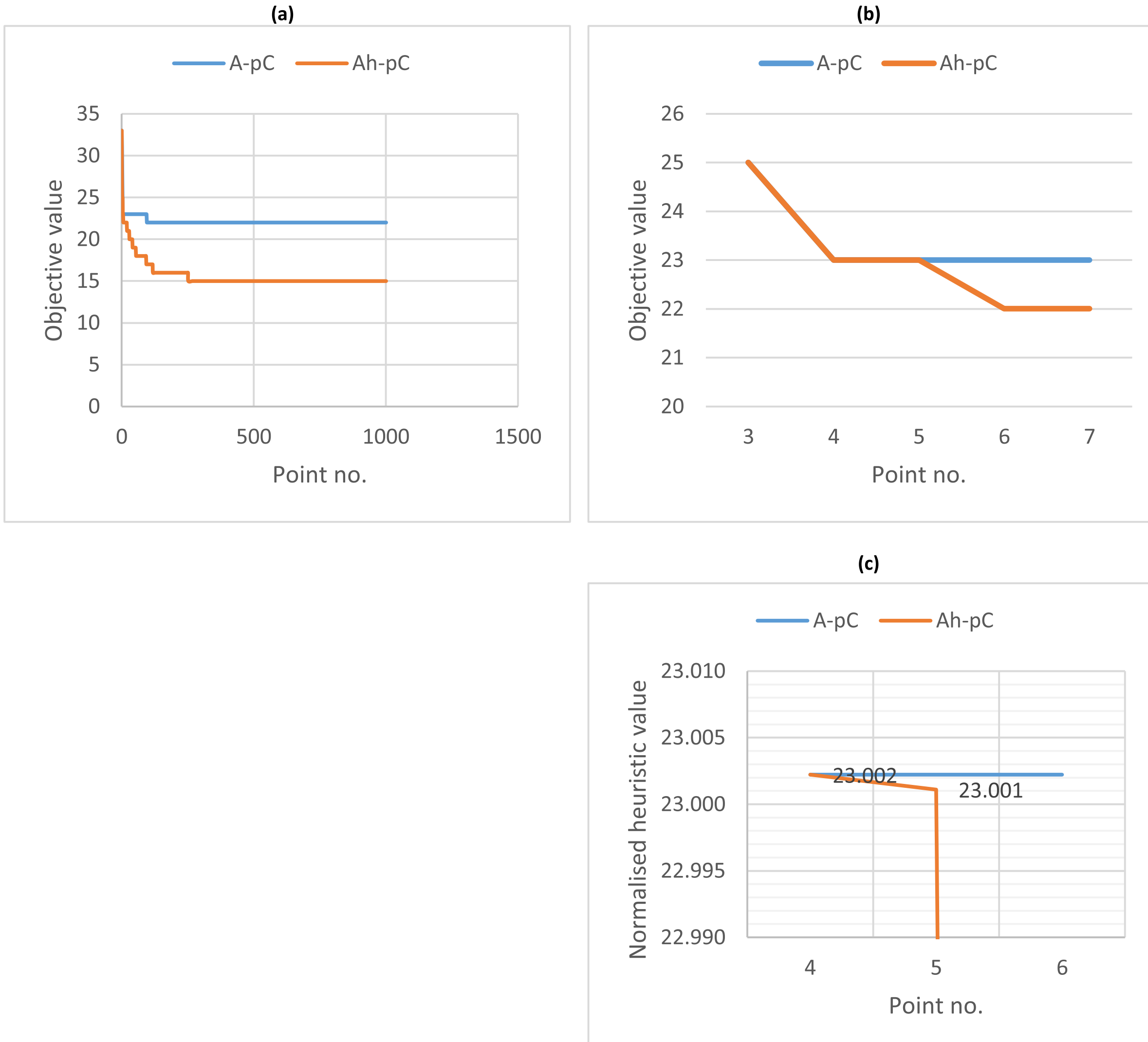


Fig. 5. (a)The objective values of the first 1000 points visited by the algorithm without the unflattening strategy (A-pC) and with it (Ah-pC) during their execution on pmed40 with $n = 900$ and $p = 90$. (b) The objective values for point no. 3–7. (c) The normalised heuristic values for point no. 4–5. The normalised heuristic value of point no. 6 is shown for A-pC but not for Ah-pC because of the small vertical axis' scale.

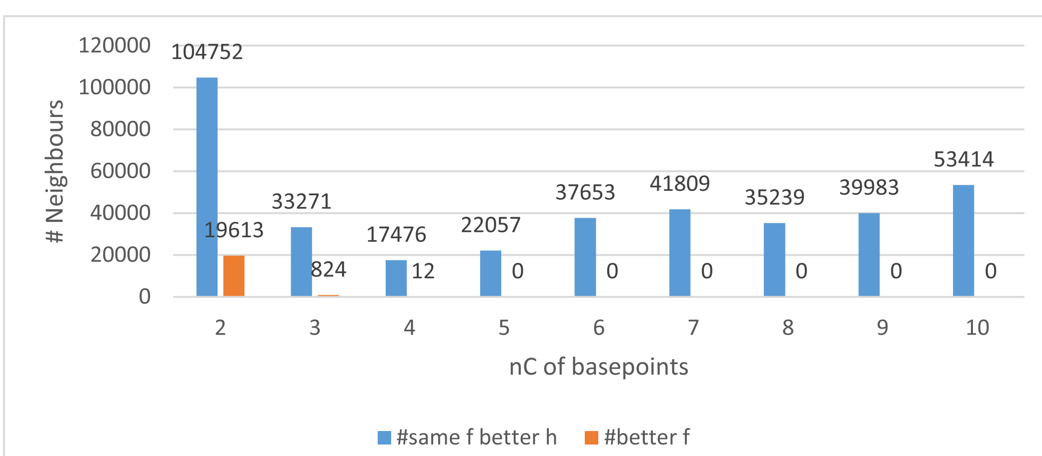


Fig. 6. The total numbers of the neighbours with improved objective values and the neighbours with the same objective but improved heuristic values shown by the number of critical vertices nC of 1000 basepoints randomly selected during the execution of *A-pC* on pmed40 with $n = 900$ and $p = 90$. The results are only shown for $2 \leqslant nC \leqslant 10$. Full results are presented in Table 8.

Table 8
The number of basepoints and three types of neighbours by the number of critical vertices of the basepoints.

nC	#Base	#Same f better h	#Better f	#Rest
1	706	0	615,763	5E + 07
2	98	104,752	19,613	7E + 06
3	23	33,271	824	2E + 06
4	8	17,476	12	565,712
5	8	22,057	0	561,143
6	22	37,653	0	2E + 06
7	19	41,809	0	1E + 06
8	17	35,239	0	1E + 06
9	8	39,983	0	543,217
10	18	53,414	0	1E + 06
11	18	74,508	0	1E + 06
12	6	14,136	0	423,264
13	7	31,488	0	478,812
14	1	3511	0	69,389
15	2	7382	0	138,418
16	5	28,873	0	335,627
17	2	4704	0	141,096
18	3	12,705	0	205,995
19	1	632	0	72,268
20	6	17,359	0	420,041
21	2	6318	0	139,482
22	1	141	0	72,759
23	3	6855	0	211,845
24	2	5764	0	140,036
25	5	8143	0	356,357
26	1	1278	0	71,622
27	2	506	0	145,294
28	1	119	0	72,781
29	0	0	0	0
30	0	0	0	0
31	0	0	0	0
32	1	197	0	72,703
33	1	149	0	72,751
34	0	0	0	0
35	0	0	0	0
36	1	2915	0	69,985
37	2	3147	0	142,653
>38	0	0	0	0

downward move. Then, it accepts a previously rejected (flat) move. Even for $2 \leqslant nC \leqslant 4$, the number of flat moves that reduce the number of critical vertices is significantly greater than the number of downward moves. Only when at a basepoint with $nC = 1$ (shown in Table 8), the number of downward moves is greater. In this case, no flat neighbour with an improved heuristic value exists because nC is at least 1 by definition.

5. Conclusions

This paper proposed new state-of-the-art metaheuristics for the *p*-Center, α-Neighbour *p*-Center and *p*-Next Center problems. The proposed algorithms share the same design, which is the integration of the first-improvement local search with two strategies to exploit flat subspaces in the search space. The local search used for these problems were different with respect to their design sophistication, from a basic local search for *p*-Next Center to a relatively sophisticated local search for *p*-Center.

The strategies integrated with the local search algorithms are the unflattening and the *IE* strategies. The unflattening strategy is to employ a heuristic function to predict which of the two flat neighbours is more promising based on properties not captured by the objective function. For the α-Neighbour *p*-Center and *p*-Next Center problems, while the objective function f is equal to $f^{(1)}$, the proposed heuristic function h_K uses the extra information $f^{(2)}$ to $f^{(K)}$ to distinguish between neighbours with the same $f^{(1)}$ value. It is still consistent with f in the sense that it prioritises solutions with better $f^{(1)}$ values. As a result, the search space that corresponds to h_K contains fewer flat neighbours than the original search space corresponding to f. In general, the number of flat neighbours decreases as K increases. Similarly, for the *p*-Center problem, extra information was used as the secondary criteria to compare flat neighbours. This extra information is the number of critical vertices. However, a method was presented to compare the heuristic values of the current candidate solution and its neighbour without calculating the heuristic value of the neighbour. The *IE* strategy is to accept flat moves (unless forbidden by tabu restriction). This simple strategy improves diversification without compromising the objective value. This and the unflattening strategies are combined by accepting flat moves in the search space that are downward or flat with respect to the heuristic function.

It is important to note that, although these strategies are promising for many cases, their integration with a local search algorithm for a given problem is not necessarily beneficial, and its impact may depend on the base local search algorithm, the way the integration is performed, the underlying problem and even the datasets used.

A second potential benefit of the unflattening strategy is to replace the objective function f with an unflattening heuristic function h which takes into account more information than that considered in the original objective function f. As a result, the final solution could be of more value to the end user. For example, for the *p*-Center problem, the final solution minimises not only the distance of the farthest client (to its nearest center) but also the number of clients with this maximum distance, which potentially adds further value to the solution. Similarly, for the α-Neighbour *p*-Center and *p*-Next Center problems, the final solution minimises not only $f = f^{(1)}$ but also $f^{(2)}$ to $f^{(K)}$ (while preserving their priorities) and is potentially more useful. The unflattening heuristic function may be further improved by considering more information (as long as its computational cost does not outweigh its benefit).

There are several potential avenues for future work. An interesting possibility is to study the case where the *f*-consistency condition of heuristic functions is relaxed. This could be achieved for α-Neighbour *p*-Center, for example, by using a constant c no more than (and possibly much less than) d_{max} in the heuristic function h_K-αNpC. The value of c may change dynamically during the search. Another possible future work is to extend the unflattening strategy from complete solutions in perturbative procedures such as local search to partial solutions in constructive procedures such as the construction phase of GRASP. Finally, an avenue for future work is the application of the unflattening and *IE* strategies to existing metaheuristics for other NP-hard optimisation problems. Such problems are countless and exist in variety of real-word domains. Indeed, the main motivation for this work was not to propose improved algorithms for the problems addressed here but to showcase and motivate further research on exploiting the potential of flat subspaces. There are numerous problems for which this potential has not yet been explored in the literature. This potential is higher for cases where a higher portion of the search space is flat. For example, it is more promising for the *p*-Center problem (as a max–min problem) whose objective function is the *maximum* of the f_i values, $i = 1, \cdots, n$, than the *p*-Median problem whose objective function is the *average* of these values. In general, it is more useful for problems where the number n_f of the distinct objective values is polynomial in input size. In such a problem, on average, there are $|S|/n_f$ points in the search space S with the same objective value, where the size $|S|$ of the search space is super polynomial because of the NP-hardness of the problem. Among such problems are the standard Maximum Satisfiability, Minimum Vertex Cover, Longest Common Subsequence, Minimum Cardinality Set Cover, Minimum Graph Colouring, Maximum Clique, and Minimum Dominating Set problems, to name a few. Each of these problems has various real-world applications and is a potential future work opportunity. Furthermore, the extension of the unflattening strategy to partial solutions, mentioned above as a potential future work, can further expand its potential applications.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research is a continuation of former research (on heuristics for NP-hard optimisation problems) supported in part by the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran. The author would also like to thank Dr. M. Albareda-Sambola and Dr. J. Sánchez-Oro for prompt responses to enquiries and Dr. M. England for valuable comments.

Appendix A. The pseudocode of the algorithm B_h-pC

Algorithm 4
Algorithm B_h-pC, with unflattening and *IE* strategies, proposed for pC.

Algorithm B_h-pC
Inputs: Matrix $D = [d_{ij}]_{n\times n}$ of shortest distances
Integer $p \in \{2, \cdots, n-1\}$

```
1   ls_while_itr_coeff = 1
2   while Termination_Condition not met do
3     Reset(P)
4     TB1 = TB2 = {} //initilaize the tabu lists
5     flg_moved = true
6     ls_best_h = h-pC(P)
7     ls_while_itr = 0
8     while flg_moved = true and ls_while_itr < ls_while_itr_coeff × (n − p) × p do
9       ls_while_itr = ls_while_itr + 1
10      SCL = [ ] //SCL holds at most the best 3 swap candidates
11      v_c = a random member of C
12      w = F^1_{v_c}
13      inx = N^{-1}_{v_c,w}
14      for each t ∈ {1, ···, inx} do //in a random order
15        v_j = N_{V_c,t}
16        if d_cj < D^1_{v_c} then
17          for v_i ∈ P do //in a random order
18            if promising(v_j, v_i, SCL) then //if it is not tabued and there is no better candidate in SCL
19              P_1 = P ∪ {v_j}\{v_i}
20              if h-pC(P_1) ≤ h-pC(P) then
21                if h-pC(P_1) = h-pC(P) then
22                    TB2 = TB2 ∪ (v_j, v_i)
23                end if
24                P = P_1 and update data structures //move
25                if h-pC(P) < ls_best_h then
26                  ls_best_h = h-pC(P)
27                  ls_while_itr = 0
28                  TB1 = TB2 = {}
29                end if
30                continue with next iteration of while loop (line 8)
31              else
32                update SCL with (v_j, v_i)
33              end if
34            end if
35          end for
36        end if
37      end for
38      if SCL ≠ [ ] then
39        (v_j, v_i) = select_move(SCL) //in a prioritised random fashion
40        P = P ∪ {v_j}\{v_i} and update data structures //move
41        TB2 = TB2 ∪ (v_j, v_i)
42        TB1 = TB1 ∪ (v_j)
43      else
44        flg_moved = false
45      end if
46    end while
47    ls_while_itr_coeff = ls_while_itr_coeff × 1.1
48  end while
49  return best solution found in all runs
```

Appendix B. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.cor.2022.106023.

References

Albareda-Sambola, M., Hinojosa, Y., Marín, A., Puerto, J., 2015. When centers can fail: A close second opportunity. Comput. Oper. Res. 62, 145–156.

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

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

Beasly JE., 1990b. OR-library. http://people.brunel.ac.uk/~mastjjb/jeb/orlib/pmedinfo.html, last access: 30 Sep. 20.

CPU Benchmarks, 2022. https://www.cpubenchmark.net, last access: 1 June 22.

Blum, C., Roli, A., 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys (CSUR) 35 (3), 268–308.

Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R., 2013. Hyper-heuristics: a survey of the state of the art. J. Operat. Res. Soc. 64 (12), 1695–1724.

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

Çalık, H., Labbé, M., Yaman, H., 2019. p-Center problems. In: Location Science. Springer, Cham, pp. 51–65.

Callaghan, R.J., 2016. An Investigation into exact methods for the continuous p-centre problem and its related problems. University of Kent. Doctoral dissertation.

Callaghan, B., Salhi, S., Brimberg, J., 2019. Optimal solutions for the continuous p-centre problem and related-neighbour and conditional problems: a relaxation-based algorithm. J. Operat. Res. Soc. 70 (2), 192–211.

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

Chaudhuri, S., Garg, N., Ravi, R., 1998. The p-neighbor k-center problem. Inform. Process. Lett. 65 (3), 131–134.

Chen, D., Chen, R., 2009. New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems. Comput. Oper. Res. 36 (5), 1646–1655.

Chen, D., Chen, R., 2013. Optimal algorithms for the α-neighbor p-center problem. Eur. J. Oper. Res. 225 (1), 36–43.

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., 1995. Network and discrete location: models, algorithms, and applications. Wiley, NewYork.

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

Davidović, T., Ramljak, D., Šelmić, M., Teodorović, D., 2011. Bee colony optimization for the p-center problem. Comput. Oper. Res. 38 (10), 1367–1376.

Drezner, T., Drezner, Z., 2014. The maximin gradual cover location problem. OR Spectrum 36 (4), 903–921.

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. In: June. A New LocAl SeArch for the P-center Problem BAsed on the CriticAl Vertex Concept. Springer, Cham, pp. 79–92.

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

Hakimi, S.L., 1965. Optimum distribution of switching centers in a communication network and some related graph theoretic problems. Oper. Res. 13 (3), 462–475.

Hassin, R., Levin, A., Morad, D., 2003. Lexicographic local search and the p-center problem. Eur. J. Oper. Res. 151 (2), 265–279.

Universität Heidelberg, 2018. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp, last access: 21 Apr. 21.

Jackson, W.G., Özcan, E., John, R.I., 2018. Move acceptance in local search metaheuristics for cross-domain search. Expert Syst. Appl. 109, 131–151.

Jayalakshmi, B., Singh, A., 2018. Two swarm intelligence-based approaches for the p-centre problem. Int. J. Swarm Intell. 3 (4), 290–308.

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.

Khuller, S., Pless, R., Sussmann, Y.J., 2000. Fault tolerant k-center problems. Theoret. Comput. Sci. 242 (1–2), 237–245.

Kramer, R., Iori, M., Vidal, T., 2020. Mathematical models and search algorithms for the capacitated-center problem. INFORMS J. Comput. 32 (2), 444–460.

Krumke, S.O., 1995. On a generalization of the p-center problem. Inform. Process. Lett. 56 (2), 67–71.

Liu, Y., 2022. Two lower-bounding algorithms for the p-center problem in an area. Comput. Urban Sci. 2 (1), 1–19.

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-Sánchez, A.D., Sánchez-Oro, J., Hernández-Díaz, A.G., 2019. GRASP and VNS for solving the p-next center problem. Comput. Oper. Res. 104, 295–303.

Lu, C.C., 2013. Robust weighted vertex p-center model considering uncertain data: an application to emergency management. Eur. J. Oper. Res. 230 (1), 113–121.

Lyu, J., Zeng, Y., Zhang, R., Lim, T.J., 2016. Placement optimization of UAV-mounted mobile base stations. IEEE Commun. Lett. 21 (3), 604–607.

Martínez-Merino, L.I., Albareda-Sambola, M., Rodríguez-Chía, A.M., 2017. The probabilistic p-center problem: planning service for potential customers. Eur. J. Oper. Res. 262 (2), 509–520.

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

Misir, M., Wauters, T., Verbeeck, K. and Vanden Berghe, G., 2009. A new learning hyper-heuristic for the traveling tournament problem. In *the 8th Metaheuristic International Conference (MIC'09), Date: 2009/07/13-2009/07/16, Location: Hamburg, Germany*.

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

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

Morales-Castañeda, B., Zaldivar, D., Cuevas, E., Fausto, F., Rodríguez, A., 2020. A better balance in metaheuristic algorithms: Does it exist? Swarm Evol. Comput. 54, 100671.

Morrison, R.W., De Jong, K.A., 2001. Measurement of population diversity. In: International Conference on Artificial Evolution. Springer, Berlin, Heidelberg, pp. 31–41.

Mousavi, S.R., Babaie, M., Montazerian, M., 2012. An improved heuristic for the far from most strings problem. J. Heuristics 18 (2), 239–262.

Mousavi, S.R., Esfahani, N.N., 2012. A GRASP algorithm for the Closest String Problem using a probability-based heuristic. Comput. Oper. Res. 39 (2), 238–248.

Murray, A.T., O'Kelly, M.E., Church, R.L., 2008. Regional service coverage modeling. Comput. Oper. Res. 35 (2), 339–355.

Pacheco, J.A., Casado, S., 2005. Solving two location models with few facilities by using a hybrid heuristic: a real health resources case. Comput. Oper. Res. 32 (12), 3075–3091.

Pinheiro, H.P., de Souza Pinheiro, A., Sen, P.K., 2005. Comparison of genomic sequences using the Hamming distance. J. Statist. Plann. Inference 130 (1–2), 325–339.

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.

Salleh, M.N.M., Hussain, K., Cheng, S., Shi, Y., Muhammad, A., Ullah, G., Naseem, R., 2018. Exploration and exploitation measurement in swarm-based metaheuristic algorithms: an empirical analysis. In: International Conference on Soft Computing and Data Mining. Springer, Cham, pp. 24–32.

Solmaz, G., Akkaya, K., Turgut, D., 2014. In: December. Communication-constrained P-center Problem for Event Coverage in Theme Parks. IEEE, pp. 486–491.

Suzuki, A., Drezner, Z., 1996. The p-center location problem in an area. Location Sci. 4 (1–2), 69–82.

Wang, W., Yang, K., Yang, L., Gao, Z., 2022. Tractable approximations for the distributionally robust conditional vertex p-center problem: application to the location of high-speed railway emergency rescue stations. J. Operat. Res. Soc. 73 (3), 525–539.

Yadav, M., Prakash, V.P., 2020. A comparison of the effectiveness of two novel clustering-based heuristics for the p-centre problem. In: Advances in Data and Information Sciences. Springer, Singapore, pp. 247–255.

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. Technol. 32 (6), 1319–1334.

Yurtkuran, A. and Emel, E., 2014. A modified artificial bee colony algorithm for p-center problems. *The Scientific World Journal*, *2014*, article ID 824196.