European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

GRASP with strategic oscillation for the α-neighbor p-center problem

J. Sánchez-Oro[a], A. D. López-Sánchez[b], A. G. Hernández-Díaz[b], A. Duarte[a,*]

[a] *Universidad Rey Juan Carlos. C/ Tulipan s/n. Móstoles 28933, Madrid, Spain*
[b] *Pablo de Olavide University. Ctra. de Utrera km 1. Sevilla 41013, Spain*

ARTICLE INFO

Article history:
Received 2 October 2020
Accepted 16 February 2022
Available online xxx

Keywords:
Metaheuristics
GRASP
Tabu search
Strategic oscillation
p-Center Problem

ABSTRACT

This paper presents a competitive algorithm that combines the Greedy Randomized Adaptive Search Procedure including a Tabu Search instead of a traditional Local Search framework, with a Strategic Oscillation post-processing, to provide high-quality solutions for the α-neighbor p-center problem ($\alpha - p$CP). This problem seeks to locate p facilities to service or cover a set of n demand points with the objective of minimizing the maximum distance between each demand point and its αth nearest facility. The algorithm is compared to the best method found in the state of the art, which is an extremely efficient exact procedure for the continuous variant of the problem. An extensive comparison shows the relevance of the proposal, being able to provide competitive results independently of the α value.

1. Introduction

There exist many real-world situations in which a set of facilities (hospitals, malls, emergency centers, etc.) require to be located in order to service a set of demand points (patients, customers, people at risk, etc.). This family of problems are commonly known as Facility Location Problems (FLP) (Basu, Sharma, & Ghosh, 2015). Depending on the objective function to optimize and the considered constraints, we can identify a large variety of variants: the p-center problem (Ferone, Festa, Napoletano, & Resende, 2017), the p-median problem (Daskin & Maass, 2015), or the maximal coverage location problem (Murray, 2016), among others. A common characteristic in most of these FLP variants is that each demand point is always served by its closest facility (when not considering capacity constraints), with the aim of reducing the response time. However, there might exist some scenarios where facilities concerned are subject to failure and, therefore, the information about the closest one is not enough. In that situation it is not only relevant to have information about the closest facility, but also the second closest, or even more.

This issue has been approached in the related literature by considering whether the customer have access to *a priori* information about the failure of facilities or not. If this information is not available, customers need to reach the corresponding closest facility to certify its unavailability. Consequently, they will first go to the closest facility and then, they will go to the next facility, which is the closest to the previous one. In this scenario, we are dealing with the p-next center problem (pNCP) (Albareda-Sambola, Hinojosa, Marín, & Puerto, 2015; López-Sánchez, Sánchez-Oro, & Hernández-Díaz, 2019), whose goal is to minimize the maximum distance between demand points and their closest facility plus the distance from this facility and to its closest facility center. Notice that this definition can be trivially extended to more than two facilities.

If the customers know in advance that some facilities are not able to satisfy their demands, they do not need to reach their closest facility and, therefore, they can directly go to a different one. In this case, we are dealing with the α-neighbor p-center problem, $\alpha - p$CP (Chen & Chen, 2013) where, instead of assigning each demand point to a single facility (such as in the classical p-median problem Daskin & Maass, 2015), it is assigned to α facilities. The objective is then to minimize the maximum distance between each demand point and its αth closest facility, where α is a problem constraint. Note that $\alpha - p$CP is able to model other situations in which the customers may prefer to be served by another facility instead of the closest one, since there are αth facilities close to each customer.

The $\alpha - p$CP has been tackled from both, continuous and discrete perspectives. In the former, facilities may be located anywhere on the plane (Chen & Chen, 2013), while in the latter facilities must be placed in a finite set of potential service points (Chen & Chen, 2013). To the best of our knowledge, the research on the

* Corresponding author.
E-mail addresses: jesus.sanchezoro@urjc.es (J. Sánchez-Oro), adlopsan@upo.es (A.D. López-Sánchez), agarher@upo.es (A.G. Hernández-Díaz), abraham.duarte@urjc.es (A. Duarte).

https://doi.org/10.1016/j.ejor.2022.02.038

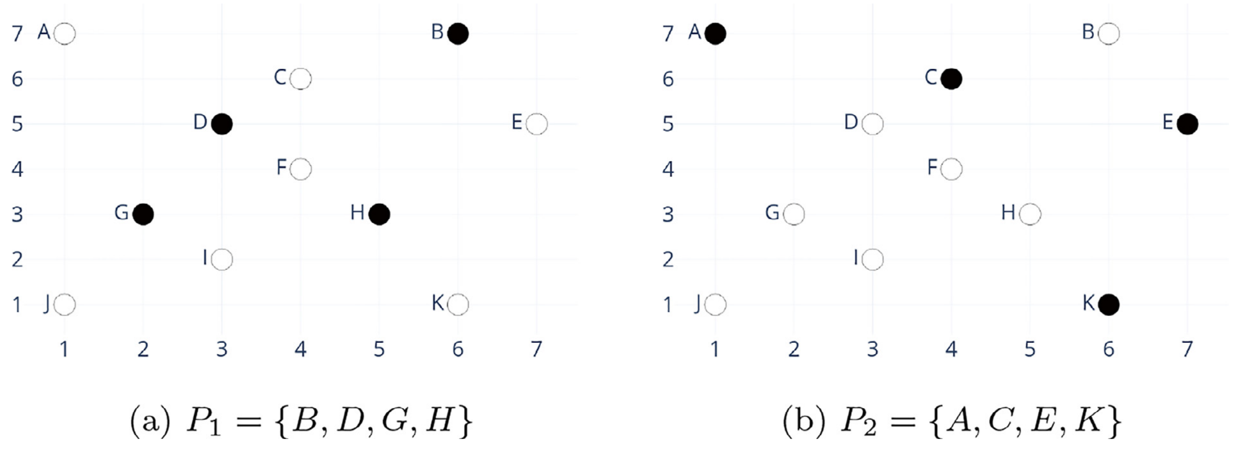

Fig. 1. Representation of solutions P_1 and P_2 for the same set of 11 points, with $p = 4$ and $\alpha = 2$.

$\alpha - p$CP has been mainly focused on the continuous version of the problem. In particular, quite effective and efficient exact methods have been recently proposed which are able to solve instances up to 1000 nodes. We refer the reader to (Callaghan, Salhi, & Brimberg, 2019) for the most recent work on the continuous $\alpha - p$CP which proposes an exact algorithm that leverages the structure of the problem to optimally solve a new set of large instances.

In this paper, we focus on the discrete variant, which can be formally defined as follows: let N be the set of points, with $|N| = n$; $d(i, j)$ be the distance between points i and j, where $i, j \in N$; and p a positive integer number, with $1 < p < |N|$. The aim of the $\alpha - p$CP is to select a set P with exactly p facilities in order to minimize the maximum distance between each demand point and its αth closest facility or, in other words, minimize the radius of the maximal circle (among the p allowed) so that each demand point is covered by at least α circles. Without loss of generality, it can be assumed that the set of candidate locations to host a facility is directly equal to N (i.e., every available point is able to host a facility). Therefore, a feasible solution for the $\alpha - p$CP can be simply represented as the set of selected facilities, P, where $P \subset N$ with $|P| = p$. It is worth mentioning that we do not explicitly indicate the set of demand points since it is determined by $N \setminus P$.

Given a set of facilities, let A be any subset of α elements taken from of P. The α-distance between any demand point $i \in N \setminus P$ and the set P, denoted as $d_\alpha(i, P)$, is defined as:

$$d_\alpha(i, P) = \min_{\substack{A \subset P \\ |A| = \alpha}} \{\max_{j \in A} d(i, j)\}.$$

The objective function of the $\alpha - p$CP, refereed to as $f_\alpha(P, N)$, is defined as:

$$f_\alpha(P, N) = \max_{i \in N \setminus P} d_\alpha(i, P)$$

Therefore, the $\alpha - p$CP consists in minimizing the aforementioned objective function. In mathematical terms:

$$\min_{\substack{P \subset N \\ |P| = p}} f_\alpha(P, N)$$

Figure 1 shows two feasible solutions for the same set of 11 points considering $p = 4$ and $\alpha = 2$, where facilities are located at points represented with a filled circle. Figure 1a represents solution $P_1 = \{B, D, G, H\}$, where $f_\alpha(P_1, N) = 4.47$ which is obtained at demand point J, whose closest facility is G and the second one is D, which is located at a distance of 4.47. In the case of solution $P_2 = \{A, C, E, K\}$, depicted at Fig. 1b, $f_\alpha(P_2, N) = 5.83$, since the second closest facility to the demand point J is C, which is at a distance of 5.83. Therefore, P_1 is better solution than P_2, since $f_\alpha(P_1, N) < f_\alpha(P_2, N)$.

The p-center problem (pCP) is a particular case of the $\alpha - p$CP when considering $\alpha = 1$. The pCP has been widely studied in the literature from both academic and real-life situations, using exact and heuristic approaches. To the best of our knowledge, the first work on the pCP was (Hakimi, 1964) where the problem was introduced, solving it by means of graphical methods. Then, an iterative set covering based exact method was proposed by Minieka (1970). The first mixed integer programming formulation was presented in Daskin (1995). Many heuristic approaches have been applied to solve the pCP, for instance, in Mladenović, Labbé, & Hansen (2003) whose authors proposed a Tabu Search and a Variable Neighborhood Search and in Yin, Zhou, Ding, Zhao, & Lv (2017), the authors propose a Greedy Randomized Adaptive Search Procedure with Path-Relinking to address the pCP.

The $\alpha - p$CP was first introduced in Krumke (1995), where the author presented this problem and showed that it is a $\mathcal{NP}$-hard problem. He also introduced an efficient algorithm for $\alpha \geq 2$ and gave a 4-approximation algorithm. Later on, the authors in Chaudhuri, Garg, & Ravi (1998) used the ideas of the previous paper to get a lower bound and proposed a 2-approximation algorithm. Other variant of the problem in which capacity constraints were considered was studied in Khuller, Pless, & Sussmann (2000) and they coined it as the α-fault-tolerant capacitated k-center. The authors proposed two polynomial-time algorithms when all demand points can be reassigned to other facilities, in the case of a failure in some facilities, and when only the demand points affected by a failure in its assigned facility can be reassigned to other facilities. Recently, in Brimberg, Maier, & Schäbel (2021) a new variant of the well-known p-median problem is presented, named distributed p-median problem, where it is assumed that the demand points interact with more than one facility simultaneously, so the closest facility is not necessarily the most relevant one.

The goal of this paper is to propose an algorithm able to solve the discrete version of the $\alpha - p$CP without the need of making particular adaptations for each α value. That is, the algorithm will provide high-quality solutions independently on which value is assigned to α. We have considered $\alpha = 1, 2, 3$ since larger values are not realistic. To this end, a Greedy Randomized Adaptive Search Procedure (GRASP) (Feo & Resende, 1989; Feo, Resende, & Smith, 1994) that includes a Tabu Search (TS) (Glover & Laguna, 1998) instead of a standard Local Search is presented and it is combined with post-processing method based on the Strategic Oscillation (SO) methodology (Glover & Laguna, 1998) to further improve the solutions found by the GRASP.

The main contributions of this work are described as follows. A GRASP algorithm is proposed for providing high-quality solutions for the $\alpha - p$CP. In order to implement a fast and reliable constructive procedure, the greedy function considered is based on the instance features instead of in the evaluation of the objective function, accelerating the construction process. Additionally to a traditional local search method, a TS metaheuristic is proposed for the improving stage of GRASP. The TS, based on a short-term memory structure, allows to escape from local optima by always performing the best move, even if it leads to a lower quality solution. Finally, a post-processing method based on Strategic Oscillation is able to leverage the exploration of the search space of unfeasible solutions being able to find better solutions for the $\alpha - p$CP. It is worth mentioning that, instead of considering the traditional SO, we propose four different alternatives that consider greedy and random perturbations and greedy and random repairing of solutions within the SO framework, to perform a deep analysis on the behavior of SO. The combination of diversification of GRASP with the intensification of TS and SO post-processing results in a very competitive algorithm for solving the discrete version of the $\alpha - p$CP.

The rest of this paper is organized as follows. Sections 2 and 3 describe the algorithms implemented to solve the problem under consideration. Section 5 presents the computational results performed to test the quality of the proposal. Section 6 analyzes the parameters selected for the proposed algorithm. Finally, Section 7 summarizes the paper and discusses future work.

2. Greedy adaptive search procedure

The algorithm proposed for solving the $\alpha - p$CP combines a Greedy Randomized Adaptive Search Procedure (GRASP) with a Strategic Oscillation (SO) based post-processing for further improving the solutions found. The GRASP metaheuristic framework was originally presented in Feo & Resende (1989) but it was not formally introduced until (Feo et al., 1994). We refer the reader to Festa & Resende (2008, 2009) for a complete analysis of this methodology.

GRASP is a multi-start procedure which can be divided into two different phases: construction and improvement. The former constructs a solution from scratch following a greedy, randomized and adaptive strategy, while the latter tries to find a local optimum departing from the corresponding constructed solution. These stages are iteratively applied until reaching a termination criterion (usually based on the number of iterations).

2.1. Constructive procedure

The constructive method within the GRASP methodology starts from an empty solution and iteratively adds elements to it until it becomes feasible. A greedy function is responsible for evaluating the quality of each candidate element, in order to select the next element to be included in the solution. Greedy functions used in the constructive phase for the GRASP framework are usually related with the objective function. However, the evaluation of the objective function for the $\alpha - p$CP is rather time consuming, since it is necessary to find the αth closest facility for each demand point. Therefore, in order to design a fast algorithm, an alternative greedy function is proposed.

Given a partial solution P, the greedy function is calculated as the distance among facilities, so that the quality of a solution is estimated when an element $i \in N \setminus P$ is inserted in a partial solution P. Formally, the greedy function $g(i, P)$ for a candidate $i \in N \setminus P$ is defined as:

$$g(i, P) = \min_{j \in P} d(i, j)$$

The greedy function then computes the minimum distance between candidate i and any element in the partial solution P.

Algorithm 1 *Construct*(N, β) .

```
1:  $i \leftarrow rand(N)$
2:  $P \leftarrow \{i\}$
3:  $CL \leftarrow N \setminus \{i\}$
4:  while $|P| < p$ do
5:      $g_{\min} \leftarrow \min_{i \in CL} g(i, P)$
6:      $g_{\max} \leftarrow \max_{i \in CL} g(i, P)$
7:      $\mu \leftarrow g_{\max} - \beta(g_{\max} - g_{\min})$
8:      $RCL \leftarrow \{i \in CL : g(i, P) \geq \mu\}$
9:      $j \leftarrow rand(RCL)$
10:     $P \leftarrow P \cup \{j\}$
11:     $CL \leftarrow CL \setminus \{j\}$
12: end while
13: return $P$
```

Algorithm 1 shows the pseudocode of the constructive procedure. The method starts by randomly selecting the first element to be included in the solution (step 1), initializing the solution with that point hosting a facility (step 2). This initial random selection is customary in GRASP as it increases the diversity of the search. After that, the candidate list, CL, is created with all the points except the one that has been firstly selected (step 3). The method iteratively adds a new element to the solution under construction (steps 4–12). In each iteration, the smallest ($g_{\min}$) and largest ($g_{\max}$) values of the greedy function over all the candidates are computed (steps 5–6). Instead of selecting the best element in each iteration, the randomized part of the constructive procedure requires a threshold μ to restrict the candidates that are considered promising (step 7). The threshold is calculated using the input $\beta \in [0, 1]$ parameter, which controls the greediness / randomness of the constructive procedure. On the one hand, if $\beta = 0$, then $\mu = g_{\max}$, being a totally greedy algorithm. On the other hand, if $\beta = 1$, then $\mu = g_{\min}$, resulting in a completely random selection. Therefore, it is important to find an appropriate value for the parameter β since it balances the grade of intensification and diversification of the constructed solutions. The restricted candidate list (RCL) contains all the candidates whose greedy function value is better (larger) than the given threshold μ (step 8). Then, the method randomly selects one of the most promising candidates included in the RCL (step 9), including it in the partial solution P (step 10), and updating the CL (step 11). The method ends by returning the constructed solution when it becomes feasible (step 13).

2.2. Local search

The improvement phase is devoted to find a local optimum with respect to a given neighborhood. In general, the neighborhood of a given solution is usually defined as the set of solutions that can be reached by performing a single movement over it. In the case of the $\alpha - p$CP, the movement consists in removing a point hosting a facility from the solution, and replacing it with another element that is not in the solution. This move is traditionally known as swap in the literature (Pérez-Peló, Sánchez-Oro, López-Sánchez, & Duarte, 2019). Given a solution P, a facility $i \in P$, and a demand point $j \in N \setminus P$, the swap movement is formally defined as:

$$Swap(P, i, j) \leftarrow P \cup \{j\} \setminus \{i\}$$

Having defined the movement, the neighborhood N_{swap} of a given solution P is conformed with all the solutions that can be generated by performing a swap move in P.

The computation of the objective function for the $\alpha - p$CP is quite time consuming, so we propose to limit, as much as possible, the number of solutions evaluated during the exploration of N_{swap}, with the aim of having an efficient local search procedure. Then, let $j^\star$ be the furthest α-neighbor facility of any demand point from those in P. Notice that one of these facilities determine the value of the objective function. This heuristic selection allows us to avoid exploring the complete 1-swap neighborhood. More formally, for any $i \in N \setminus P$, the facility $j^\star$ is determined as:

$$j^\star \leftarrow \arg\min_{\substack{A \subset P \\ |A| = \alpha}} \{\max_{j \in A} d(i, j)\}.$$

Therefore, the set of neighbor solutions of P is mathematically defined as:

$$N_S(P) \leftarrow \{Q : Q \leftarrow Swap(P, i, j^\star),\ \forall i \in N \setminus P\}$$

The second key element of a local search consists in defining the way in which the considered neighborhood is explored. The most extended strategies are Best Improvement (BI) and First Improvement (FI). On the one hand, BI selects, in each iteration, the best solution in the current neighborhood, thus requiring to explore the complete neighborhood, being rather time consuming. On the other hand, FI selects the first solution that presents a better objective function value that the incumbent one. As the evaluation of the objective function in the $\alpha - p$CP is very computationally demanding, a local search method based in the First Improvement strategy (LS-FI) is proposed in order to reduce the computing time of the proposed algorithm. Additionally, to increase diversification, the solutions in the neighborhood are explored at random in each iteration. A comparison of performance between these two local search strategies is shown in Section 5.

2.3. Tabu search

The main drawback of traditional local search procedures is that they strongly depend on the departing solution, getting trapped easily in a local optimum. In order to overcome this drawback, a second improvement method based on Tabu Search (TS) (Glover & Laguna, 1998) is proposed. This methodology helps traditional local search procedures to escape from local optima. The two key features of TS can be summed up in using an adaptive memory and accepting non-improving moves. The proposed TS method starts similarly to the previously described local search, selecting the first improving solution in the neighborhood under exploration. However, a short-term memory stores the last elements that have been removed from the solution in every *Swap* move performed. This memory follows a First-In First-Out (FIFO) scheme and has a predefined maximum size given by the *tenure* parameter. An element cannot be included in the solution again while it is stored in the memory, in order to restrict the available moves. Elements stored in the memory are labelled as tabu-active.

The TS method always performs the best available move even though it implies deteriorating the quality of the objective function. The aim of this strategy is to avoid getting trapped in local optima. The search stops when no improvement is found after a predefined number of iterations.

As it is well described in the related literature (Glover & Laguna, 1998), the use of memory may prohibit attractive moves. We consider the so-called aspiration criterion, where a move with the tabu status (since it involves tabu-active elements) is performed if it results in a solution with an objective value strictly better than that of the best solution found so far. Notice that there is not danger of cycling since the new solution has not been previously visited. Section 5 will discuss the influence of these parameters over the global performance of TS.

3. Strategic oscillation

As far as we know, all neighborhoods reported in the literature for the $\alpha - p$CP are based on swapping demand points that do not host a facility with those hosting facilities. This is the natural design to preserve feasibility during the search. In this paper, the exploration of non-feasible solutions by means of Strategic Oscillation (SO) is proposed. The idea of exploring unfeasible solutions for the continuous $\alpha - p$CP was recently explored in Elshaikh et al. (2016) where a perturbation method is presented that allows to increase the number of facilities to reach better, although unfeasible, solutions. Then, solutions are repaired by removing facilities until it becomes feasible again. This strategy is based on the original idea presented in Salhi (1997).

SO was originally introduced in the context of TS (Glover & Laguna, 1998) and has not been studied as thoroughly as other strategies. The boundary that we intend to cross in the current context is to explore the search space that includes solutions for which the size cardinality constraint may be violated. Our goal is to design a SO mechanism that is able to act as a post-processing method for the solutions generated with the GRASP algorithm.

The oscillation between feasibility and unfeasibility is determined by a parameter δ. Specifically, this method converts the incumbent solution into an unfeasible one, $\hat{P}$, by incorporating δ additional facilities to the solution P, where $0 \leq \delta \leq \delta_{\max}$ and $|\hat{P}| = p + \delta$. Notice that the addition of extra facilities always improve (or, at least, do not deteriorate) the value of the objective function, since all the demand points have a distance equal than or closer to the set of facilities in this situation. Finally, the solution is repaired by removing the extra δ facilities.

For the general case, two new movement operators are considered, called *Insert* and *Remove*. The former transforms a demand point into a facility, while the latter removes a facility from the solution. More formally, given a solution P and a demand point $i \in N \setminus P$, the insert movement is formally defined as:

$$Insert(P, i) \leftarrow P \cup \{i\}$$

while the *Remove* movement is defined as:

$$Remove(P, j) \leftarrow P \setminus \{j\}$$

Algorithm 2 depicts the pseudocode of the proposed Strategic

Algorithm 2 $SO(P, \delta_{\max})$.

```
1:  $\delta \leftarrow 1$
2:  while $\delta \leq \delta_{\max}$ do
3:      $\hat{P} \leftarrow P$
4:      while $|\hat{P}| < |P| + \delta$ do
5:          $j \leftarrow \begin{cases} rand(N \setminus \hat{P}) & \textit{Random Strategy} \\ \arg\max_{c \in N \setminus \hat{P}} d(c, \hat{P}) & \textit{Greedy Strategy} \end{cases}$
6:          $\hat{P} \leftarrow Insert(\hat{P}, j)$
7:      end while
8:      $\hat{P} \leftarrow Improve(\hat{P})$
9:      while $|\hat{P}| > |P|$ do
10:         $i \leftarrow \begin{cases} rand(\hat{P}) & \textit{Random Strategy} \\ \arg\min_{c \in \hat{P}} d(c, \hat{P} \setminus \{c\}) & \textit{Greedy Strategy} \end{cases}$
11:         $\hat{P} \leftarrow Remove(\hat{P}, i)$
12:     end while
13:     $\hat{P} \leftarrow Improve(\hat{P})$
14:     if $f_\alpha(\hat{P}, N) < f_\alpha(P, N)$ then
15:         $P \leftarrow \hat{P}$
16:         $\delta \leftarrow 1$
17:     else
18:         $\delta \leftarrow \delta + 1$
19:     end if
20: end while
21: return $P$
```

Oscillation procedure. It starts by initializing the number of additional elements to be included in the solution (step 1). Then, it

Table 1
Description of the different SO strategies presented.

Name	Insert	Remove
SO-RR	Random	Random
SO-RG	Random	Greedy
SO-GR	Greedy	Random
SO-GG	Greedy	Greedy

iterates until reaching the maximum number of elements to be included, $\delta_{\max}$, which is an input parameter (steps 2–20). At the beginning of each iteration, SO creates a copy $\hat{P}$ of the best solution found, which will be used for performing the oscillation (step 3). Then, the method starts with the insertion phase (steps 4–7). In this phase, δ demand points which do not host a facility are selected. We study either greedy or random strategies to incorporate new facilities into the solution (step 5). The former consists in adding the element that keeps the maximum distance between the elements selected to host a facility. For the sake of simplicity, let us define $d(i, P)$ as the minimum distance between i and any element of P. More formally,

$$d(i, P) \leftarrow \min_{j \in P} d(i, j).$$

Therefore, the greedy insertion strategy selects the element to be incorporated according to the following equation:

$$j \leftarrow \arg\max_{c \in N \setminus P} d(c, \hat{P})$$

The second criterion selects an element in $N \setminus \hat{P}$ at random. Once we have the corresponding facility, it is inserted in the solution $\hat{P}$, becoming unfeasible (step 6). After including δ elements, the improvement method is applied to $\hat{P}$ to find a local optimum which still unfeasible (step 8). In order to design an algorithm as much generic as possible either, the local search (Section 2.2) or the tabu search (Section 2.3) are applied to the unfeasible solution to improve $\hat{P}$ (step 8).

Once a local optimum has been found, the removal phase (steps 9–12) iteratively selects a facility to be removed (step 10). Again either greedy and random strategies are considered. The former consists in using the following greedy criterion:

$$i \leftarrow \arg\min_{c \in \hat{P}} d(c, \hat{P} \setminus \{c\})$$

The later selects a facility at random. Once we have the corresponding facility it is removed from the solution (step 11). This process is repeated until $\hat{P}$ becomes feasible. Then, the improvement method (either, the local search or the tabu search) is applied to $\hat{P}$ to find a local optimum (step 13). Finally, if an improvement is found (step 14), the best solution and the δ parameter are updated (steps 15 and 16). Otherwise, the algorithm increases the number of elements to be included in the next iteration (step 18). SO stops when no improvement is found after reaching the maximum size, returning the best solution found during the search (step 21).

With the aim of studying the effect of diversification and intensification in the context of Strategic Oscillation, we have proposed greedy and random strategies for both insertion and removal phases, following a similar scheme than the one presented in Sánchez-Oro, Pantrigo, & Duarte (2014). Therefore, four different SO strategies are finally analyzed, which are described in Table 1.

As can be derived from the results, *SO-RR* is totally focused on diversification, while *SO-GG* is designed for intensifying the search. Both *SO-RG* and *SO-GR* try to find a balance between intensification and diversification. Section 5 will deeply evaluate the influence of each variant in the results obtained.

4. Algorithmic approach

This section is designed for providing an overall view of the proposed algorithm, including all the stages that have been previously described. In particular, the proposed GRASP+SO algorithm is composed of two different stages. First of all, a solution is constructed and a local optimum with respect to the defined neighborhood is found. Then, the SO post-processing method is applied to further improve the generated solution. These two steps are repeated through a predefined number of iterations Δ, which will be tuned in the final algorithm.

Having defined all the proposed procedures, we now present the overall pseudocode in Algorithm 3. This algorithm requires

Algorithm 3 $GRASP + SO(N, \beta, \delta_{\max}, \Delta)$.

```
1: $P_b \leftarrow \emptyset$
2: for $i \in 1 \ldots \Delta$ do
3:     $P \leftarrow Construct(N, \beta)$
4:     $P' \leftarrow Improve(P)$
5:     $P'' \leftarrow SO(P', \delta_{\max})$
6:     if $f_\alpha(P'', N) < f_\alpha(P_b, N)$ then
7:         $P_b \leftarrow P''$
8:     end if
9: end for
```

4 input parameters. The first one, N, refers to the instance data, while the last three parameters are search parameters of the algorithm: β, responsible to balance the greediness/randomness of the constructive procedure, $\delta_{\max}$, to indicate how far from feasibility is the SO post-processing, and Δ, to determine the number of iterations that will be performed.

The method performs a given number of iterations Δ (steps 2-9). In each iteration, a solution P' is constructed using the constructive procedure presented in Section 2.1. Then, the solution is improved by using the local search or the tabu search presented in Section 4. Finally, with the aim of escaping from local optima, the SO procedure described in Section 5 is executed over the improved solution P', generating a new solution P''. Once a complete iteration is executed, the solution P'' resulting from the SO post-processing is compared to the best solution found so far P_b. If an improvement is found, then P_b is updated (step 6-8). The method ends when Δ iterations have been performed, returning the best solution found during the search, P_b.

5. Computational results

This section presents and discusses the results of the computational testing conducted with the proposed algorithm in this paper. In particular, an analysis of the contribution of SO with respect to GRASP is performed, highlighting the advantages and disadvantages of the proposal when considering the $\alpha - p$CP. Additionally, the best GRASP and SO configuration is compared to the best method found in the state of the art (Chen & Chen, 2013).

We have considered the same set of 37 instances used in the previous work, which are derived from the well-known TSP-Lib[1]. Additionally, we have also considered $\alpha = 1, 2, 3$, since larger values represent unrealistic situations. Therefore, the complete benchmark is composed of 111 instances since the set of 37 instances has been solved once per each α value. In order to consider larger instances, we have also included the instances proposed in Callaghan et al. (2019), which are more challenging for heuristic methods. All algorithms were implemented in Java 11 and executed over an AMD Ryzen 5 3600 (2.2 GHz) with 16GB RAM.

[1] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html.

Table 2
Comparison among the considered β parameter values.

β	Avg.	Time(s)	Dev(%)	#Best
RND	505.38	0.03	4.79	9
0.00	515.94	0.03	4.75	18
0.25	491.55	0.03	3.22	11
0.50	474.45	0.03	5.21	7
0.75	565.44	0.03	17.85	4
1.00	722.80	0.03	48.19	0

Table 3
Comparison among the considered β parameter values when coupled with local search.

Algorithm	Avg.	Time(s)	Dev(%)	#Best
GRASP(*RND*)	414.79	0.85	1.53	14
GRASP(0.00)	422.91	0.80	4.01	9
GRASP(0.25)	416.46	0.92	1.93	18
GRASP(0.50)	418.86	0.88	2.32	9
GRASP(0.75)	439.98	0.89	6.63	8
GRASP(1.00)	551.96	0.83	26.25	2

The experiments are divided into two different phases: preliminary and final experimentation. The former is intended to select the best configuration for the proposed algorithm, while the latter consists in a competitive testing among GRASP, GRASP+SO, and the state of the art for evaluating the quality of the proposal. The preliminary experiments are performed over a representative subset of instances (30 out of 111 instances) that were randomly selected to avoid overfitting.

All the experiments report the following metrics: Avg., the average objective function value; Time(s), the computing time required by the algorithm to finish in seconds; Dev(%), the average deviation with respect to the best solution found in the experiment; and #Best, the number of times that the algorithm reaches the best solution of the experiment.

5.1. Preliminary results

The first experiment is designed to evaluate the best value for the β parameter. In particular, we have tested $\beta = \{0.00, 0.25, 0.50, 0.75, 1.00, RND\}$, where *RND* indicates that the value is selected at random in each construction. We have included values 0.00 and 1.00 to see how a completely greedy and random (respectively) construction influences in the quality of the results. Table 2 shows the results obtained when considering these β values. In the context of GRASP, more than one solution must be constructed to leverage the diversification stage of the metaheuristic. For this problem, 100 constructions have been considered for each value of the β parameter.

As can be derived from the results, the best value for the β parameter is $\beta = 0.00$ in terms of number of best solutions found, while in deviation it is the third best option, after $\beta = 0.25$ and $\beta = RND$. This behavior can be partially explained since $\beta = 0.00$ results in a completely greedy algorithm and, for a construction without improvement, is the option focused on intensification. These results indicate that it is not recommendable to consider large β values since they include too much randomness in the search.

The next experiment (Table 3) aims to evaluate the behavior of the constructive procedures when coupled with the local search procedure (100 constructions followed by a local search for each construction). In the context of GRASP, the best constructive procedure is not necessarily the most adequate one, since including more diversification can eventually lead the local search procedure to find better solutions.

Table 4
Comparison between first and best improvement strategies in the proposed local search method.

Strategy	Avg.	Time (s)	Dev(%)	#Best
Best Improvement	414.87	12.50	0.91	20
First Improvement	414.79	1.42	0.40	24

Table 5
Comparison of different *tenure* values in the Tabu Search.

tenure	Avg.	Time(s)	Dev(%)	#Best
0.1	403.31	18.85	0.03	29
0.2	403.47	19.44	0.12	28
0.3	403.48	18.84	0.15	27
0.4	403.48	18.26	0.15	27
0.5	403.48	18.12	0.15	27

In this case, the GRASP(*RND*) is able to reduce the average deviation as well as the average objective function value, presenting the best results in these metrics. On the other hand, GRASP(0.25) obtains a larger number of best values. If we now analyze the results obtained by the completely greedy variant, $\beta = 0.00$, we can see how focusing just in intensification does not necessarily lead to better results. Even more, the greedy variant results in the fourth place, being better than the variants which are totally focused in diversification. These results experimentally confirm the hypothesis that including more diversification in the constructive procedure can eventually lead to better solutions. Therefore, the selected value for β is *RND*.

Since the proposed local search follows a first improvement strategy, it is interesting to compare it with a best improvement approach, with the aim of evaluating if the increase in intensification provided by a best improvement strategy is interesting to be used in the context of $\alpha - p$CP. To that end, the next experiment compares the results obtained by the proposed first improvement approach with the best improvement strategy, by repeating the last scheme: 100 constructions followed by a local improvement for each construction. Table 4 shows the results obtained in this experiment.

The first relevant result that can be extracted from the table is that considering best improvement instead of first improvement does not affect to the quality of the generated solutions. Indeed, although both results are rather similar, first improvement is able to reach 24 best solutions while best improvement reaches 20. However, the average deviation of both strategies is close to zero, indicating that the results are equivalent. Notwithstanding, the results in terms of computing time are determinant to select first improvement instead of best improvement. In particular, first improvement is almost 9 times faster than best improvement. Therefore, first improvement is the selected strategy for the final algorithm.

The next experiments are devoted to adjust the parameters for the TS procedure. The first one evaluates the influence of the *tenure* parameter, which indicates the length of the tabu list. The values compared are $tenure = \{0.1 \cdot p, 0.2 \cdot p, 0.3 \cdot p, 0.4 \cdot p, 0.5 \cdot p\}$. It is worth mentioning that the values indicate a percentage of elements in the solution to make the algorithm more scalable with different problem sizes. In order to obtain meaningful results, we set the maximum number of iterations without improvement, denoted as τ, to 20 (the influence of this parameter will be studied in the following experiment).

Table 5 shows the results obtained with these *tenure* values. Although the results are rather similar in quality, it can be easily observed that the smaller the tenure, the better the results. Therefore, we have selected $tenure = 0.1 \cdot p$ as the best value for the

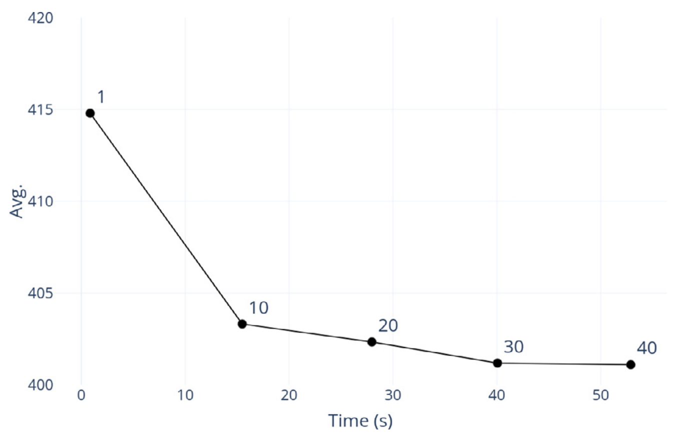

Fig. 2. Evolution of the objective function value (Y-axis) with respect to computing time (X-axis) when considering different values for the stopping criterion of TS.

Table 6
Comparison of local search and tabu search when considering similar computing times.

Algorithm	Avg.	Time(s)	Dev(%)	#Best
Local Search	407.25	39.68	1.69	15
Tabu Search	402.33	31.73	1.16	23

Table 7
Analysis of different $\delta_{\max}$ values for the SO algorithm.

$\delta_{\max}$	Avg.	Time(s)	Dev(%)	#Best
0.1	367.05	239.57	1.40	11
0.2	361.83	434.60	0.58	17
0.3	360.44	642.83	0.63	20

Table 8
Comparison among different SO strategies for the $\alpha-p$CP.

Strategy	Avg.	Time(s)	Dev(%)	#Best
SO-RR	374.38	318.63	2.55	11
SO-RG	383.30	262.48	2.90	14
SO-GR	361.83	434.60	1.11	17
SO-GG	382.18	357.98	2.47	12

Table 9
Comparison of the effect of including SO in the final algorithm or not.

Strategy	Avg.	Time(s)	Dev(%)	#Best
With SO	361.83	434.60	0.00	30
Without SO	402.33	31.73	34.26	0

TS parameter, since it presents a smaller average objective function value (403.31), smaller deviation (0.03%), and larger number of best solutions (29) in equivalent computing times (about 19 s). Attending to the results showed in the previous experiment, we can observe that GRASP is considerably faster than any TS variant.

The second parameter to be tuned in TS is the maximum number of iterations without improvement τ used as a stopping criterion. In this case, we have analyzed the values $\tau = \{1, 10, 20, 30, 40\}$. Figure 2 shows the evolution of the solution quality for these τ values.

As expected, the larger the value of τ, the better the solution. However, the computing time also drastically increases with the τ value. Therefore, in order to find a compromise between quality and computing time, we select $\tau = 20$ with the aim of limiting the time spent in the TS procedure to a value smaller than 30 s on average.

In order to evaluate whether longer CPU times for GRASP would result in a better algorithm than TS, we have executed both algorithms for similar CPU time. In particular, the number of constructions of GRASP when considering local search methods has been increased until reaching similar computing times than when considering Tabu Search. Table 6 shows the results obtained in this comparison.

The obtained results show the limitations of the local search method when compared to the proposed tabu search. In particular, TS maintains a smaller deviation (1.16% versus 1.69%) and a larger number of best solutions found (23 versus 15). This experiment additionally shows the influence of including TS in the final version of the algorithm.

The next experiment is devoted to set the value of the $\delta_{\max}$ parameter inside the SO framework, which limits the maximum number of elements to be added to the incumbent solution. In other words, it limits how far from feasibility a given solution is. Notice that selecting a large value will result in a completely different solution, which may be equivalent to construct a new solution from scratch. Therefore, we set $\delta_{\max} = \{0.1 \cdot p, 0.2 \cdot p, 0.3 \cdot p\}$. As we did with the *tenure* parameter, $\delta_{\max}$ is evaluated as a percentage of the number of selected elements to ease scalability.

Results show that, as expected, the larger the $\delta_{\max}$ value, the more computationally demanding SO becomes. However, it does not necessarily results in better solutions. Analyzing the deviation, we can see that $\delta_{\max} = 0.2$ is able to reach the smallest deviation, with a slightly smaller number of best solutions found (17 versus 20). Additionally the average objective function value is very similar, which suggests that the increment in approximately 200 s of computing time does not worth it. Therefore, we select $\delta_{\max} = 0.2$ for the final algorithm.

In the following preliminary experiment, the influence of diversification and intensification in the SO framework is evaluated. To this end, we test all SO variants: *SO-RR*, *SO-RG*, *SO-GR*, *SO-GG*. Table 8 shows this comparison.

As can be derived from the results, the best option is to consider a greedy oscillation and a random repair, since it is able to achieve the smallest deviation and the largest number of best solutions found. The main drawback is that a greedy oscillation always requires more computing time than the random one. Therefore, we have opted by SO-GR as the best strategy.

Finally, to evaluate the relevance of SO within the complete algorithm proposed, we have performed an additional experiment to evaluate the quality of the results obtained with the algorithm before and after considering the SO post-processing method (Table 9).

First of all, it is worth mentioning that the SO post-processing is the most time consuming part of the algorithm, which is reasonable considering that it implies the execution of two improvement stages as it can be seen in Algorithm 2. However, the impact on the quality of the solutions generated highlights the relevance of considering this post-processing stage. In particular, it is able to reach all the best solutions, resulting in an average deviation of 34.26% when not considering it. Therefore, SO has been included in the final version of the algorithm.

To summarize, the best configuration for the proposed algorithm consists of $\beta = RND$, $tenure = 0.1 \cdot p$, $\tau = 20$, $\delta_{\max} = 0.2 \cdot p$, and the *SO-GR* strategy. For the sake of brevity, we denote this algorithm as SO.

Table 10
Comparison of the results obtained by the adaptation of Chen & Chen (2013) coupled with an exhaustive local search, the multistart approach, and the proposed SO when considering $\alpha = 1$. The original results of the continuous version are also included as a lower bound for the discrete problem.

$\alpha = 1$	Lower Bound		Chen & Chen (2013) + LS			MultiStart + LS			SO		
Instance	O.F.	Time (s)	O.F.	Time (s)	Dev(%)	O.F.	Time (s)	Dev(%)	O.F.	Time (s)	Dev(%)
att48_48_10	836.34	0.09	1401.71	0.09	16.50%	1380.96	0.20	14.78	1203.18	1.85	0.00
att48_48_20	474.65	0.11	921.43	0.11	29.64	1203.18	0.09	69.28	710.77	0.65	0.00
att48_48_30	251.07	0.08	921.43	0.08	99.41	1203.18	0.06	160.38	462.08	0.22	0.00
att48_48_40	186.95	0.07	331.05	0.07	3.50	1203.18	0.03	276.18	319.85	0.06	0.00
ch150_150_10	122.55	8.59	177.64	8.59	25.51	162.18	3.25	14.59	141.53	82.96	0.00
ch150_150_100	18.91	1.78	45.25	1.78	35.19	86.70	0.36	159.05	33.47	1.50	0.00
ch150_150_110	16.48	1.83	48.22	1.83	59.78	86.70	0.28	187.28	30.18	0.95	0.00
ch150_150_120	13.77	1.73	40.72	1.73	48.86	86.70	0.21	216.94	27.36	0.55	0.00
ch150_150_130	11.91	1.76	33.40	1.76	48.75	86.70	0.15	286.15	22.45	0.26	0.00
ch150_150_140	8.80	1.46	36.68	1.46	108.71	86.70	0.09	393.31	17.58	0.11	0.00
ch150_150_20	76.80	14.33	133.37	14.33	37.31	124.60	1.86	28.28	97.13	39.97	0.00
ch150_150_30	55.72	9.55	118.75	9.55	49.26	103.02	1.35	29.49	79.56	18.01	0.00
ch150_150_40	47.41	6.14	93.49	6.14	37.02	87.68	1.09	28.51	68.23	12.22	0.00
ch150_150_50	38.01	3.94	79.66	3.94	30.72	86.70	0.93	42.27	60.94	7.74	0.00
ch150_150_60	31.98	3.95	68.02	3.95	37.02	86.70	0.78	74.65	49.64	6.29	0.00
ch150_150_70	27.06	2.11	68.02	2.11	46.35	86.70	0.65	86.55	46.48	4.38	0.00
ch150_150_80	24.99	1.72	49.64	1.72	19.72	86.70	0.54	109.10	41.46	3.28	0.00
ch150_150_90	20.73	1.83	45.95	1.83	19.75	86.70	0.47	125.92	38.38	2.18	0.00
eil101_101_10	12.14	2.72	18.68	2.72	30.48	17.46	0.98	21.95	14.32	26.04	0.00
eil101_101_100	0.71	0.18	1.41	0.18	0.00	12.72	0.02	799.44	1.41	0.05	0.00
eil101_101_20	7.53	5.78	12.73	5.78	23.62	13.00	0.59	26.27	10.30	8.50	0.00
eil101_101_30	5.76	2.06	12.04	2.06	46.03	12.72	0.42	54.25	8.25	4.75	0.00
eil101_101_40	4.61	1.16	11.18	1.16	53.57	12.72	0.33	74.72	7.28	2.89	0.00
eil101_101_50	3.64	1.00	10.82	1.00	52.97	12.72	0.26	79.89	7.07	1.80	0.00
eil101_101_60	3.35	0.65	7.62	0.65	20.42	12.72	0.19	101.12	6.32	1.08	0.00
eil101_101_70	2.69	0.35	7.21	0.35	44.22	12.72	0.14	154.40	5.00	0.55	0.00
eil101_101_80	2.24	0.28	6.71	0.28	62.70	12.72	0.10	208.51	4.12	0.25	0.00
eil101_101_90	1.58	0.20	6.32	0.20	100.00	12.72	0.06	302.24	3.16	0.08	0.00
pr439_439_10	1716.51	42.85	2580.94	42.85	30.89	2660.94	94.80	34.95	1971.83	1800.85	0.00
pr439_439_20	1029.71	120.01	2958.57	120.01	146.49	2289.65	51.90	90.76	1200.26	1566.51	0.00
pr439_439_30	739.19	279.80	1630.38	279.80	83.87	1746.42	36.95	96.96	886.71	761.30	0.00
pr439_439_40	580.01	319.76	1530.52	319.76	109.99	1530.52	28.84	109.99	728.87	490.00	0.00
pr439_439_50	468.54	368.89	1570.03	368.89	161.67	1510.38	23.49	151.73	600.00	294.52	0.00
pr439_439_60	400.20	370.46	1654.73	370.46	201.80	1386.09	20.04	152.80	548.29	230.09	0.00
pr439_439_70	357.95	271.50	1630.38	271.50	226.08	1364.73	17.47	172.95	500.00	175.29	0.00
pr439_439_80	312.50	264.24	1630.38	264.24	242.76	1364.73	15.47	186.91	475.66	155.71	0.00
pr439_439_90	280.90	93.01	1144.83	93.01	175.14	1364.73	13.82	227.99	416.08	131.18	0.00

5.2. Final results

The main objective of this section is to compare the results obtained with our best algorithm with the procedure introduced in Chen & Chen (2013). As aforementioned, the best method identified in the state of the art is an exact algorithm for solving the continuous version of the problem. With the aim of having a fair comparison with the discrete version tackled in this work, the exact algorithm for the continuous version has been adapted. In particular, each point given in the output of the continuous version is mapped to the closest point of the discrete variant. Since this adaptation can result in low quality solutions, we have additionally applied an exhaustive local search to the resulting solution. Therefore, the solution compared is a local optimum with respect to the considered $N_S(P)$ for the initial solution P given by the exact procedure for the continuous version. Additionally, we have included a complete multistart procedure which starts from a random solution and then applies the exhaustive local search method to find a local optimum. The objective of considering this multistart approach is to evaluate if the adaptation of the continuous version provides reasonable results in the discrete approach.

In this experiment, we consider the whole set of 111 instances. Tables 10, 11, and 12 show the associated results for $\alpha = 1, 2$, and 3, respectively, for the small and medium instances. We report, for each instance, the value of the objective function, O.F., the corresponding computing CPU time in seconds, Time(s), and the deviation with respect to the best result found, Dev (%), of the proposed Strategic Oscillation method (*SO*), the state-of-the-art method coupled with the exhaustive local search (Chen & Chen (2013) + LS), and the multistart approach, respectively. Additionally, the results obtained directly with the algorithm for the continuous version (Lower Bound) are included as a lower bound, since they correspond to unfeasible solution in the context of the discrete version. Notice that a time limit of 1800 s per instance has been set, stopping the algorithm when reaching that limit, returning the best solution found so far.

The proposed SO algorithm consistently finds the best results in all the instances, even considering the exhaustive local search in the adaptation for the continuous version and in the multistart approach, which highlights the relevance of using specific strategies for the discrete version of the tackled problem. Indeed, the differences between the two methods become more evident as the value of α increases, again supporting the same hypothesis. Additionally, the size of the instance is also a relevant factor in the analysis, since the results of SO are considerably better when the size of the instance increases. It is worth mentioning that the adaptation of the continuous version coupled with the exhaustive local search is competitive for the discrete version when considering instances with up to 100 nodes. The computing time required by the proposed SO is also affected by the instance size, being considerably higher for the largest instances. The comparison between the adaptation of the continuous version and the multistart approach shows how the former is able to reach better results in small instances, but the latter provides better quality solutions in the largest instances, highlighting that the quality of the adaptation of the continuous version deteriorates when the size of the instance increase.

Table 11
Comparison of the results obtained by the adaptation of Chen & Chen (2013) coupled with an exhaustive local search, the multistart approach, and the proposed SO when considering $\alpha = 2$. The original results of the continuous version are also included as a lower bound for the discrete problem.

$\alpha = 2$	Lower Bound		Chen & Chen (2013) + LS			MultiStart + LS			SO		
Instance	O.F.	Time (s)	O.F.	Time (s)	Dev(%)	O.F.	Time (s)	Dev(%)	O.F.	Time (s)	Dev(%)
att48_48_10	1377.53	0.20	2334.17	0.20	46.61	1931.36	0.32	21.31	1592.12	5.14	0.00
att48_48_20	820.45	0.22	1910.06	0.22	68.90	1590.90	0.14	40.68	1130.85	1.21	0.00
att48_48_30	601.59	0.22	1849.16	0.22	97.48	1590.90	0.07	69.90	936.38	0.42	0.00
att48_48_40	474.65	0.19	1267.89	0.19	138.29	1590.90	0.03	199.00	532.08	0.07	0.00
ch150_150_10	184.06	0.95	241.53	0.95	17.44	235.33	6.60	14.43	205.66	223.16	0.00
ch150_150_100	38.01	3.61	91.90	3.61	72.71	89.67	0.53	68.52	53.21	2.35	0.00
ch150_150_110	33.24	3.46	77.13	3.46	49.33	89.67	0.43	73.62	51.65	1.36	0.00
ch150_150_120	31.98	3.20	77.13	3.20	53.34	89.67	0.28	78.28	50.30	0.72	0.00
ch150_150_130	29.55	2.49	87.22	2.49	87.04	89.67	0.19	92.28	46.63	0.31	0.00
ch150_150_140	27.06	2.23	71.82	2.23	69.78	89.67	0.10	111.99	42.30	0.14	0.00
ch150_150_20	122.55	13.20	194.37	13.20	37.33	165.44	3.48	16.89	141.53	94.75	0.00
ch150_150_30	93.98	26.27	161.82	26.27	43.82	140.76	2.36	25.10	112.51	55.58	0.00
ch150_150_40	76.38	20.94	118.83	20.94	23.25	117.58	1.84	21.95	96.42	31.74	0.00
ch150_150_50	65.42	19.11	166.24	19.11	89.58	105.67	1.43	20.51	87.69	18.10	0.00
ch150_150_60	55.72	8.02	129.57	8.02	65.23	98.11	1.14	25.11	78.42	12.24	0.00
ch150_150_70	50.78	5.97	117.58	5.97	72.33	94.65	0.99	38.72	68.23	8.20	0.00
ch150_150_80	47.41	4.22	98.11	4.22	52.24	89.67	0.77	39.14	64.45	5.57	0.00
ch150_150_90	41.28	4.14	103.50	4.14	66.85	89.67	0.63	44.54	62.04	3.63	0.00
eil101_101_10	19.46	0.45	25.94	0.45	22.29	24.04	2.05	13.33	21.21	68.12	0.00
eil101_101_100	3.64	0.68	6.32	0.68	123.61	13.00	0.03	359.62	2.83	0.05	0.00
eil101_101_20	12.14	7.98	19.10	7.98	35.09	17.46	1.13	23.46	14.14	28.27	0.00
eil101_101_30	9.22	7.22	18.68	7.22	55.68	14.31	0.75	19.25	12.00	10.63	0.00
eil101_101_40	7.53	9.60	15.13	9.60	60.41	13.00	0.53	37.80	9.43	6.19	0.00
eil101_101_50	6.36	4.05	15.81	4.05	83.80	13.00	0.39	51.12	8.60	3.19	0.00
eil101_101_60	5.76	2.85	13.60	2.85	64.94	13.00	0.29	57.65	8.25	1.94	0.00
eil101_101_70	5.00	1.61	13.60	1.61	86.83	13.00	0.21	78.57	7.28	0.96	0.00
eil101_101_80	4.53	1.43	12.04	1.43	90.39	13.00	0.13	105.55	6.32	0.43	0.00
eil101_101_90	4.03	1.06	9.43	1.06	88.68	13.00	0.07	160.00	5.00	0.11	0.00
pr439_439_10	2752.64	0.37	3779.05	0.81	20.10	3530.22	217.34	12.19	3146.63	1800.97	0.00
pr439_439_20	1716.51	0.84	2440.54	26.27	9.63	2765.07	117.33	24.20	2226.26	1800.89	0.00
pr439_439_30	1271.83	0.94	3222.19	24.36	114.78	2575.12	78.47	71.65	1500.21	1800.18	0.00
pr439_439_40	1008.17	3.40	3300.00	76.85	163.16	2079.96	58.75	65.87	1253.99	1800.24	0.00
pr439_439_50	874.27	4.53	2432.33	183.53	127.75	2010.28	47.02	88.23	1068.00	1327.63	0.00
pr439_439_60	739.19	7.90	2037.31	487.18	108.95	1667.70	39.28	71.05	975.00	918.13	0.00
pr439_439_70	621.74	25.81	2575.12	1271.12	184.37	1541.50	33.65	70.23	905.54	639.50	0.00
pr439_439_80	580.01	24.80	2277.06	497.79	211.13	1453.44	29.05	98.59	731.86	509.87	0.00
pr439_439_90	530.48	49.18	2432.33	291.78	239.76	1453.44	25.42	103.03	715.89	405.64	0.00

However, the proposed SO remains as the best approach for solving the considered problem. Finally, it is important to remark that **SO** is able to reach results which are relatively close, compared to the other methods investigated, to the lower bound obtained with the optimal result for the continuous version.

Notice that for those instances with less than 200 nodes, the exact procedure is able to solve the continuous version in less than ten seconds. Symmetrically, it requires more than 1000 s in those instances with 439 nodes.

In order to test the limits of the exact procedure, we include a new set of large instances obtained from Callaghan et al. (2019), where the number of nodes ranges from 575 to 1323, being a more challenging set. Tables 13, 14, and 15 show the results obtained when considering this new set of large instances.

First of all, the complexity of these instances is high even for the continuous version, which it is able to optimally solve just one instance in less than 1800 s. Therefore, the value reported for the exact method is the best value found in that computing time limit. Again, the proposed SO is able to reach the best results in all the instances for the three considered values of α. Even more, it is able to provide better solutions than the ones reported with the exact method for the continuous version in 1800 s. It is worth mentioning that when considering this new set of large and more complex instances, the multistart approach performs considerably better than the adaptation of the continuous version, being competitive in some instances with the proposed method. These results support the appropriateness of proposing a specific algorithm design for the discrete variant of the tackled problem, such as SO.

5.3. What if the demand points are included in the objective function evaluation?

In the original work of the $\alpha - p$CP (Chen & Chen, 2013), when a demand point becomes a facility it is not considered that the closest facility is the demand point itself, i.e., the evaluation of the α-closest facility ignores that the demand point is already hosting a facility. However, considering the demand point itself in the evaluation if it hosts a facility is more realistic, since a customer or user will not travel to another location if the service can be satisfied in the point where it is currently located. With the aim of evaluating the impact of this new approach, we evaluate the performance of both, the multistart approach (which has been proven to be better than the adaptation of the continuous version) and the proposed SO algorithm.

After performing the same experiment as the one reported in Section 5.2, but including the modification in the evaluation of the objective function value, we have noticed that the algorithm reports the same results. The rationale behind this is that the modification of the objective function value benefits to those points hosting a facility, but it does not affect the other demand points. Since it is a problem where the objective function is computed as the maximum distance to the α-neighbor among all the points, we have experimentally checked that it always corresponds to a

Table 12
Comparison of the results obtained by the adaptation of Chen & Chen (2013) coupled with an exhaustive local search, the multistart approach, and the proposed SO when considering $\alpha = 3$. The original results of the continuous version are also included as a lower bound for the discrete problem.

$\alpha = 3$	Lower Bound		Chen & Chen (2013) + LS			MultiStart + LS			SO		
Instance	O.F.	Time (s)	O.F.	Time (s)	Dev(%)	O.F.	Time (s)	Dev(%)	O.F.	Time (s)	Dev(%)
att48_48_10	1965.63	0.01	2755.07	0.01	26.01	2373.56	0.50	8.56	2186.31	6.72	0.00
att48_48_20	1179.16	0.35	2011.66	0.33	46.36	1841.40	0.21	33.97	1374.48	1.61	0.00
att48_48_30	829.38	0.54	1628.44	0.50	60.97	1841.40	0.10	82.02	1011.66	0.54	0.00
att48_48_40	676.75	0.22	1374.48	0.22	103.63	1841.40	0.04	172.80	675.00	0.08	0.00
ch150_150_10	295.81	0.15	330.21	0.16	10.60	301.53	10.19	0.99	298.56	398.03	0.00
ch150_150_100	52.60	5.76	109.42	7.79	57.78	103.50	0.66	49.24	69.35	3.23	0.00
ch150_150_110	48.80	6.83	134.29	8.85	99.78	103.50	0.50	53.98	67.22	1.85	0.00
ch150_150_120	47.41	4.79	108.78	5.18	77.48	103.50	0.37	68.87	61.29	0.95	0.00
ch150_150_130	42.59	4.62	104.00	4.43	80.86	103.50	0.24	79.99	57.50	0.41	0.00
ch150_150_140	40.01	4.82	77.13	4.33	47.75	103.50	0.13	98.26	52.20	0.16	0.00
ch150_150_20	168.41	19.58	203.04	13.83	12.98	202.24	5.36	12.53	179.71	150.94	0.00
ch150_150_30	122.55	26.00	211.07	35.26	44.17	164.50	3.50	12.36	146.41	78.08	0.00
ch150_150_40	103.43	25.38	167.28	25.21	40.32	146.95	2.56	23.26	119.22	52.10	0.00
ch150_150_50	87.13	26.36	161.82	27.68	49.80	128.22	1.99	18.69	108.03	26.70	0.00
ch150_150_60	76.38	21.82	164.50	26.16	68.78	115.73	1.58	18.74	97.46	17.78	0.00
ch150_150_70	69.37	31.32	134.29	30.88	44.68	110.63	1.28	19.19	92.82	13.10	0.00
ch150_150_80	61.58	17.73	131.63	17.82	57.86	103.99	1.04	24.71	83.38	8.34	0.00
ch150_150_90	55.72	9.74	133.65	12.64	67.45	103.50	0.83	29.68	79.81	4.75	0.00
eil101_101_10	30.22	0.11	33.24	0.10	12.95	32.57	3.13	10.68	29.43	92.44	0.00
eil101_101_100	5.17	2.24	10.20	2.15	260.62	15.13	0.03	434.93	2.83	0.05	0.00
eil101_101_20	16.32	2.70	20.81	3.41	15.43	20.61	1.66	14.32	18.03	43.73	0.00
eil101_101_30	12.14	4.77	16.28	5.61	15.12	17.46	1.08	23.46	14.14	19.37	0.00
eil101_101_40	10.31	10.51	21.40	11.04	77.72	15.81	0.78	31.29	12.04	9.71	0.00
eil101_101_50	9.01	6.60	21.54	7.03	102.63	15.13	0.55	42.33	10.63	4.74	0.00
eil101_101_60	7.53	8.05	13.60	8.47	50.19	15.13	0.39	67.08	9.06	2.22	0.00
eil101_101_70	6.71	8.09	14.42	8.72	68.77	15.13	0.27	77.08	8.54	1.06	0.00
eil101_101_80	6.08	4.06	15.13	4.49	107.83	15.13	0.18	107.83	7.28	0.44	0.00
eil101_101_90	5.76	2.50	10.77	2.95	77.06	15.13	0.09	148.74	6.08	0.11	0.00
pr439_439_10	3989.30	0.49	4601.22	0.09	12.88	4257.34	352.85	4.44	4076.23	1800.47	0.00
pr439_439_20	2347.51	6.27	3220.64	29.76	18.14	3222.18	194.21	18.20	2726.03	1800.09	0.00
pr439_439_30	1716.51	2.01	3390.06	18.51	51.90	2986.84	135.48	33.84	2231.73	1800.48	0.00
pr439_439_40	1407.62	3.21	3222.19	67.48	95.89	2674.06	99.55	62.57	1644.88	1800.42	0.00
pr439_439_50	1226.02	3.23	3486.58	38.89	137.61	2620.23	78.16	78.57	1467.35	1800.05	0.00
pr439_439_60	1019.99	5.76	2622.26	105.76	95.69	2575.12	63.10	92.17	1340.01	1800.03	0.00
pr439_439_70	946.46	17.48	3222.19	102.37	161.73	2333.72	53.27	89.56	1231.11	1316.50	0.00
pr439_439_80	853.03	54.00	3390.06	402.48	178.43	2065.49	45.97	69.64	1217.58	955.74	0.00
pr439_439_90	739.19	21.18	2982.03	491.31	202.29	2065.49	39.70	109.38	986.47	723.38	0.00

demand point that is not hosting a facility, resulting in the same results as the one reported in the previous section.

6. Experimental analysis

This section is devoted to deeply analyze the parameters settings for the proposed algorithm. Specifically, we first test the robustness and reliability of the proposed algorithm by executing 30 times the algorithm over the 30 instances considered in the preliminary experimentation. Differences in the objective function are rather large, so we use the average deviation with respect to the best value found in the 30 independent executions, since it is a dimensionless metric. Figures 3, 4, and 5, depict the associated Box and Whisker plot for $\alpha = 1$, $\alpha = 2$, and $\alpha = 3$, respectively, reporting, for each instance, the minimum and the maximum shown at the far lower and upper of the chart, respectively; quartiles 1 and 3 in the far lower and upper of the box, respectively; the median represented with an horizontal line inside the box; and the mean represented with an horizontal dotted line. Note that the outliers have been excluded.

As can be derived from the results, the proposed algorithm presents a robust behavior when considering 30 independent executions in the instances considered for the preliminary experiments. As expected, our proposal is able to find robust solutions for the $\alpha - p$CP independently of the α value. Specifically, in all the instances the mean and median values are very close, with the median under the mean in most of the cases. This result indicates that the algorithm is able to diversify the search to explore a larger portion of the search space but the intensification phase is able to lead the algorithm to high-quality solutions.

We additionally conduct a sensitivity analysis. In particular, for each parameter of the proposed algorithm, namely β (the balance between greediness/ randomness of the constructive method), *tenure* (the length of the tabu list), τ (the number of iterations without improvement), and $\delta_{\max}$ (the percentage of the size increment inside Strategic Oscillation), we evaluate different possibilities while fixing the remaining parameters to the best values found in the preliminary experimentation. In particular, we test the same values than the ones used in the preliminary experimentation, executing for each instance and each parameter setting 30 independent iterations of each algorithm.

We use the Friedman test to evaluate whether the performance of the proposed algorithm varies significantly (in terms of the objective function value) when we vary a single parameter as mentioned above. To that end we use the p-value denoted as ρ-value since p indicates the number of facilities in our optimization problem and a significance level of 5%. The statistical results show us that β, τ, and $\delta_{\max}$ are statistically significant since its ρ-values are smaller than 0.05, specifically, 0.019, 0.000 and 0.010, respectively. This implies that these parameters are sensitive. See Table 16, 17, and 18 where the Wilcoxon tests are reported to check between which parameters are those differences. If we focus on Table 16 only $\beta = 0.75$ gets different results, and this is the only β value that gets worse values for objective function (see Tables 2 and 3 to check this affirmation). This statement is logical since very large β values find more greedy so-

Table 13
Comparison of the results obtained by the adaptation of Chen & Chen (2013) coupled with an exhaustive local search, the multistart approach, and the proposed SO when considering $\alpha = 1$ over the set of large instances. The original results of the continuous version are also included as a lower bound for the discrete problem.

$\alpha = 1$	Lower Bound		Chen & Chen (2013) + LS			MultiStart + LS			SO		
Instance	O.F.	Time (s)	O.F.	Time (s)	Dev(%)	O.F.	Time (s)	Dev(%)	O.F.	Time (s)	Dev(%)
pr1002_10.txt	2472.01	1886.09	3546.83	1.32	35.89	2934.70	937.09	12.44	2610.08	1800.04	0.00
pr1002_100.txt	715.13	1846.89	1600.78	0.48	111.57	1253.99	150.60	65.73	756.64	466.26	0.00
pr1002_20.txt	2010.67	1823.56	2755.45	1.23	53.50	2418.67	508.37	34.73	1795.13	1800.01	0.00
pr1002_30.txt	1640.51	1803.38	2352.13	0.93	63.39	2061.55	347.75	43.20	1439.62	1617.02	0.00
pr1002_40.txt	1308.32	1806.87	2170.83	1.11	73.11	1806.23	276.79	44.04	1253.99	1289.79	0.00
pr1002_50.txt	1170.07	1806.19	2280.35	0.60	107.95	1622.49	246.22	47.96	1096.59	993.45	0.00
pr1002_60.txt	1101.14	1820.00	1900.66	0.41	90.14	1457.73	202.70	45.83	999.64	986.67	0.00
pr1002_70.txt	1001.07	1804.61	1758.55	0.50	91.31	1430.03	179.13	55.57	919.24	895.47	0.00
pr1002_80.txt	891.64	1852.29	1346.29	0.44	58.11	1353.69	176.82	58.98	851.47	696.52	0.00
pr1002_90.txt	814.26	1810.06	1480.71	0.35	87.30	1253.99	168.97	58.62	790.57	561.37	0.00
rat575_10.txt	67.93	604.34	97.42	0.35	33.45	88.40	179.77	21.10	73.00	809.91	0.00
rat575_100.txt	16.42	8462.51	32.65	0.11	43.18	32.31	25.19	41.69	22.80	52.65	0.00
rat575_20.txt	51.10	1800.36	103.17	0.24	103.09	65.85	92.15	29.62	50.80	478.66	0.00
rat575_30.txt	38.72	1801.22	76.16	0.17	82.26	54.23	63.53	29.78	41.79	254.97	0.00
rat575_40.txt	30.06	27511.44	49.82	0.13	37.02	48.37	50.98	33.03	36.36	175.37	0.00
rat575_50.txt	25.83	3286.70	40.61	0.13	24.73	45.27	41.63	39.05	32.56	115.20	0.00
rat575_60.txt	23.22	3667.31	40.45	0.10	36.97	40.22	36.72	36.20	29.53	96.59	0.00
rat575_70.txt	20.86	27204.47	38.47	0.12	39.09	36.35	32.40	31.42	27.66	83.90	0.00
rat575_80.txt	19.03	4546.50	37.48	0.12	47.02	34.82	30.77	36.58	25.50	70.72	0.00
rat575_90.txt	17.46	2462.78	33.02	0.10	36.50	34.78	26.96	43.80	24.19	57.53	0.00
rat783_10.txt	86.35	1807.01	98.23	1.26	15.25	102.06	448.53	19.74	85.23	1800.00	0.00
rat783_100.txt	20.80	1949.20	44.18	0.29	63.52	38.89	66.59	43.94	27.02	219.71	0.00
rat783_20.txt	64.19	1831.76	138.92	0.55	132.41	73.34	235.48	22.69	59.77	1263.44	0.00
rat783_30.txt	53.50	1816.26	68.47	0.43	39.62	64.84	162.53	32.22	49.04	762.19	0.00
rat783_40.txt	44.92	1815.66	66.61	0.30	54.74	57.28	128.81	33.07	43.05	621.21	0.00
rat783_50.txt	40.26	1801.75	69.03	0.29	81.91	52.39	103.84	38.06	37.95	464.33	0.00
rat783_60.txt	32.42	1800.93	51.62	0.22	48.41	48.08	98.41	38.22	34.79	412.68	0.00
rat783_70.txt	27.67	1922.27	44.05	0.23	36.78	44.82	84.89	39.18	32.20	342.63	0.00
rat783_80.txt	25.62	1830.71	71.87	0.34	138.90	43.04	74.05	43.07	30.08	301.16	0.00
rat783_90.txt	22.93	1906.27	46.87	0.29	66.45	41.00	68.45	45.60	28.16	273.28	0.00
rl1323_10.txt	3234.12	1829.81	4213.25	4.99	34.58	3580.14	2285.40	14.36	3130.67	1800.05	0.00
rl1323_100.txt	973.52	1806.53	1325.68	1.13	50.65	1631.50	357.40	85.40	880.00	1329.81	0.00
rl1323_20.txt	2430.07	1827.34	3119.18	3.24	49.36	2931.06	1283.05	40.35	2088.39	1800.03	0.00
rl1323_30.txt	2023.46	1863.60	2948.00	2.83	68.87	2217.94	897.35	27.05	1745.76	1800.02	0.00
rl1323_40.txt	1720.12	1813.06	2579.84	1.91	77.70	2073.28	714.51	42.81	1451.77	1800.01	0.00
rl1323_50.txt	1518.40	1805.90	1925.14	1.70	49.20	2021.87	600.31	56.69	1290.32	1800.01	0.00
rl1323_60.txt	1374.07	1805.65	2021.87	1.41	69.69	1764.64	520.06	48.10	1191.50	1800.01	0.00
rl1323_70.txt	1186.25	1832.08	1911.26	1.66	77.65	1643.64	467.56	52.77	1075.86	1800.02	0.00
rl1323_80.txt	1120.79	1803.55	1735.40	1.08	75.74	1631.50	425.53	65.22	987.47	1788.07	0.00
rl1323_90.txt	1032.91	1825.12	1735.40	1.01	87.25	1631.50	388.57	76.04	926.77	1433.94	0.00

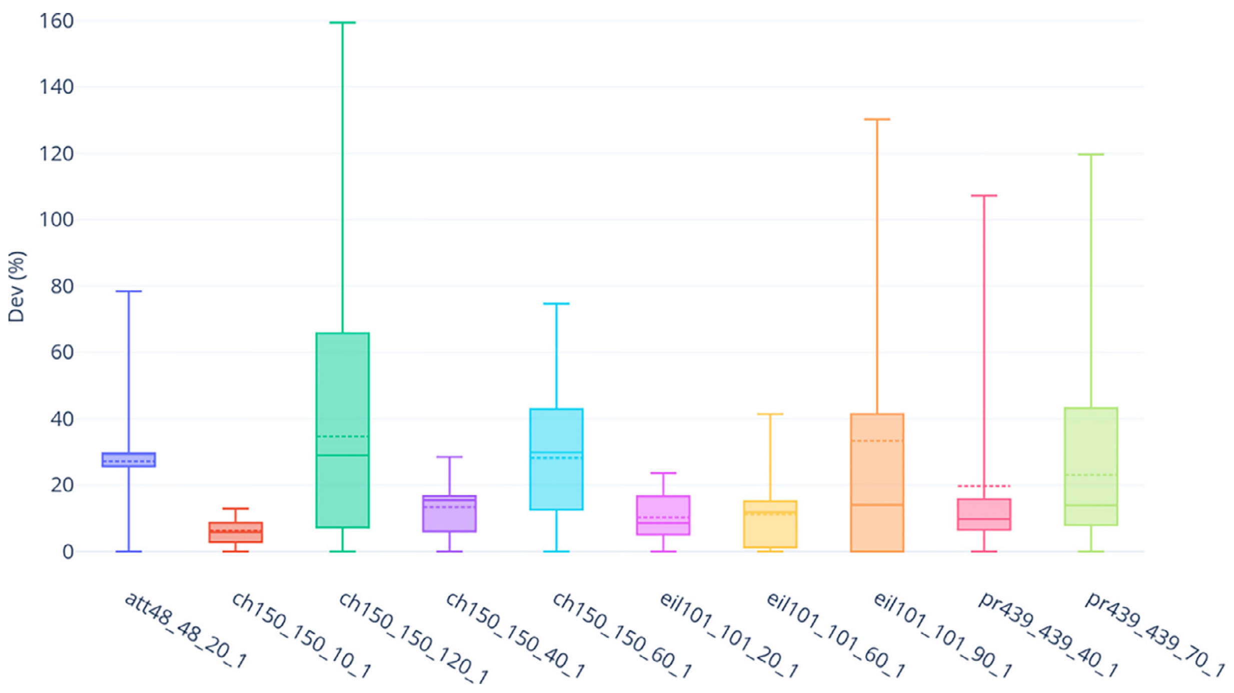

Fig. 3. Box and Whisker plot for 30 independent executions with $\alpha = 1$.

Table 14
Comparison of the results obtained by the adaptation of Chen & Chen (2013) coupled with an exhaustive local search, the multistart approach, and the proposed SO when considering $\alpha = 2$ over the set of large instances. The original results of the continuous version are also included as a lower bound for the discrete problem.

$\alpha = 2$	Lower Bound		Chen & Chen (2013) + LS			MultiStart + LS			SO		
Instance	O.F.	Time (s)	O.F.	Time (s)	Dev(%)	O.F.	Time (s)	Dev(%)	O.F.	Time (s)	Dev(%)
pr1002_10.txt	3641.56639	2.68	5521.0957	7.453	43.26	4373.20996	1991.881	13.48	3853.89404	1800.051	0.00
pr1002_100.txt	979.164806	7564.59	2280.3508	0.635	111.27	1662.06995	272.082	53.99	1079.35168	1337.796	0.00
pr1002_20.txt	2438.60823	7221.86	3628.0159	3.286	33.87	3070.01001	1058.489	13.28	2710.16602	1800.026	0.00
pr1002_30.txt	2048.99741	7202.64	3076.1177	1.899	43.04	2616.29004	726.696	21.66	2150.5813	1800.017	0.00
pr1002_40.txt	1781.5722	7335.57	2616.2952	1.563	44.41	2325.93994	553.193	28.38	1811.76709	1800.017	0.00
pr1002_50.txt	1565.87409	7351.33	3640.055	1.044	124.78	2040.21997	452.728	25.99	1619.41345	1800.008	0.00
pr1002_60.txt	1429.50268	7619.14	2350.5317	0.824	64.17	1949.93005	391.611	36.19	1431.7821	1800.006	0.00
pr1002_70.txt	1214.1438	7212.5	2651.8862	0.978	96.98	1835.75	347.732	36.36	1346.29126	1800.003	0.00
pr1002_80.txt	1115.86084	7633.17	2280.3508	0.913	81.99	1700.72998	312.153	35.73	1252.99646	1800.001	0.00
pr1002_90.txt	1028.65203	7385.37	2061.5527	0.659	76.13	1662.06995	280.157	42.00	1170.46997	1696.716	0.00
rat575_10.txt	114.063676	24.05	169.2956	0.582	44.86	119.099998	372.123	1.91	116.867447	1773.453	0.00
rat575_100.txt	26.915694	7856.19	54.147945	0.16	72.53	41.1800003	41.388	31.21	31.3847103	122.603	0.00
rat575_20.txt	71.344768	7896.24	96.17692	0.455	29.53	83.1800003	186.398	12.03	74.2495804	988.023	0.00
rat575_30.txt	58.453715	7385.02	85.61542	0.319	41.11	70.4499969	125.041	16.12	60.6712456	666.002	0.00
rat575_40.txt	49.561311	7730.75	72.00694	0.197	40.09	63.0600014	95.302	22.68	51.4003906	565.115	0.00
rat575_50.txt	42.13177	9105.17	114.28473	0.284	145.67	57.0699997	76.123	22.68	46.5188141	402.004	0.00
rat575_60.txt	38.059788	7226.7	58.821766	0.203	41.42	50.1500015	64.877	20.57	41.5932693	290.616	0.00
rat575_70.txt	34.465866	8398.71	66.287254	0.155	75.85	47.6699982	57.206	26.46	37.6961555	268.171	0.00
rat575_80.txt	31.717659	7261.39	56.0803	0.172	56.20	46.8400002	50.435	30.46	35.9026451	221.253	0.00
rat575_90.txt	29.487913	8186.07	53.814495	0.124	60.16	42.0099983	44.63	25.03	33.6005936	158.913	0.00
rat783_10.txt	131.846236	7.33	156.4257	3.133	12.86	144.199997	914.485	4.04	138.600143	1800.041	0.00
rat783_100.txt	33.836169	7369.35	60.00833	0.424	60.09	48	108.821	28.06	37.4833298	536.227	0.00
rat783_20.txt	79.312681	4819.7	123.9879	0.909	43.54	101.529999	480.244	17.54	86.3770828	1800.011	0.00
rat783_30.txt	70.362021	7204.76	138.92444	1.016	96.12	81.5999985	326.13	15.19	70.8378448	1717.016	0.00
rat783_40.txt	58.704094	7353.28	97.082436	0.712	61.42	71.5100021	250.268	18.90	60.1414986	1695.858	0.00
rat783_50.txt	51.672447	7500.51	96.0833	0.632	81.97	67.3499985	201.198	27.55	52.8015137	1212.406	0.00
rat783_60.txt	47.04916	7212.99	78.64477	0.484	61.31	59.2200012	170.918	21.47	48.7544861	1044.989	0.00
rat783_70.txt	42.138877	7338.42	93.00538	0.361	109.44	56.8800011	151.989	28.09	44.4072075	1038.246	0.00
rat783_80.txt	38.892918	7368.19	84.50444	0.457	99.18	53.0800018	132.217	25.11	42.4264069	748.929	0.00
rat783_90.txt	36.018601	7585.51	91.787796	0.522	134.13	51.6100006	113.502	31.64	39.2045898	722.807	0.00
rl1323_10.txt	4441.01278	22.77	5545.8267	16.349	18.14	4917.31006	4680.675	4.75	4694.15479	1800.123	0.00
rl1323_100.txt	1224.82566	7211.82	2948	2.227	127.89	2021.87	669.575	56.29	1293.63367	1800.017	0.00
rl1323_20.txt	3028.99135	7708	4466.1147	4.186	38.40	3472.12012	2490.425	7.60	3227.00171	1800.062	0.00
rl1323_30.txt	2496.03101	7458.43	4233.673	3.071	65.17	3084.1499	1770.238	20.32	2563.29785	1800.064	0.00
rl1323_40.txt	2098.58495	7254.52	3474.8467	5.179	60.36	2846.3999	1400.585	31.35	2166.95557	1800.039	0.00
rl1323_50.txt	1907.27506	7224.21	2886.6147	2.902	51.31	2525.71997	1156.037	32.40	1907.69385	1800.043	0.00
rl1323_60.txt	1701.80347	7248.55	4473.2134	3.424	157.76	2242.68994	1000.898	29.23	1735.39624	1800.024	0.00
rl1323_70.txt	1540.88384	7591.59	3149.8882	2.881	97.46	2021.87	877.173	26.75	1595.19775	1800.042	0.00
rl1323_80.txt	1439.20601	7258.54	4458.52	1.933	209.43	2021.87	793.622	40.32	1440.89417	1800.022	0.00
rl1323_90.txt	1332.22127	7246.26	4415.3623	2.146	221.18	2021.87	724.1	47.08	1374.72034	1800.024	0.00

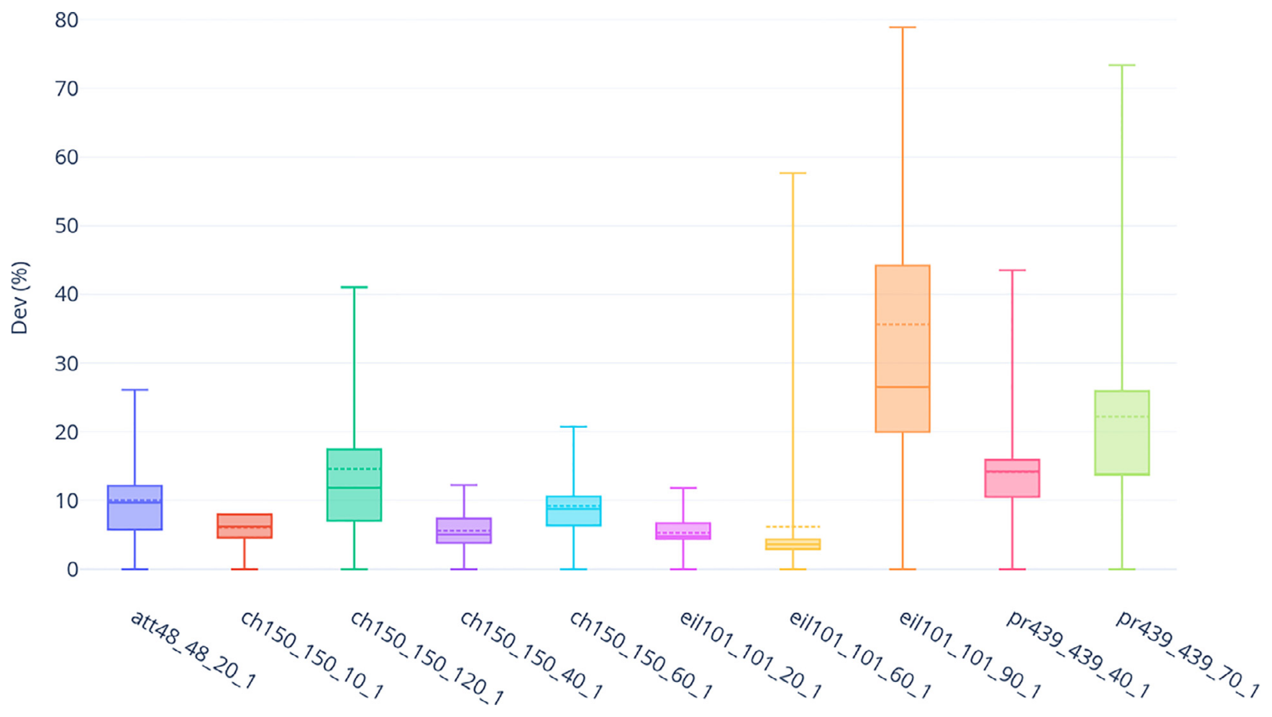

Fig. 4. Box and Whisker plot for 30 independent executions with $\alpha = 2$.

Table 15
Comparison of the results obtained by the adaptation of Chen & Chen (2013) coupled with an exhaustive local search, the multistart approach, and the proposed SO when considering $\alpha = 3$ over the set of large instances. The original results of the continuous version are also included as a lower bound for the discrete problem.

$\alpha = 3$	Lower Bound		Chen & Chen (2013) + LS			MultiStart + LS			SO		
Instance	O.F.	Time (s)	O.F.	Time (s)	Dev(%)	O.F.	Time (s)	Dev(%)	O.F.	Time (s)	Dev(%)
pr1002_10.txt	5268.06	0.53	6783.25	6.19	27.24	5510.89	2959.31	3.37	5331.28	1800.10	0.00
pr1002_100.txt	1288.67	7221.12	2960.57	1.25	118.70	1830.98	382.82	35.26	1353.70	1800.00	0.00
pr1002_20.txt	3107.09	35.85	5338.77	2.64	62.27	3734.30	1603.47	13.50	3290.14	1800.04	0.00
pr1002_30.txt	2522.88	7259.93	4301.16	1.41	62.66	3061.04	1110.37	15.76	2644.33	1800.03	0.00
pr1002_40.txt	2168.78	7900.26	3592.35	1.80	55.86	2728.09	841.52	18.36	2304.89	1800.03	0.00
pr1002_50.txt	1926.85	7436.76	3658.55	1.52	81.74	2452.04	688.20	21.81	2013.08	1800.02	0.00
pr1002_60.txt	1785.53	7220.27	3573.51	1.22	94.37	2304.88	592.94	25.37	1838.48	1800.03	0.00
pr1002_70.txt	1644.12	7511.96	3217.53	0.98	88.13	2220.36	526.73	29.83	1710.26	1800.02	0.00
pr1002_80.txt	1528.04	7246.86	3300.38	1.12	117.38	2050.60	465.60	35.07	1518.22	1800.01	0.00
pr1002_90.txt	1406.77	7252.89	2651.89	0.85	83.88	1950.00	431.13	35.21	1442.22	1800.01	0.00
rat575_10.txt	137.02	0.26	165.58	3.58	17.84	143.17	567.78	1.89	140.52	1800.01	0.00
rat575_100.txt	35.84	7371.34	70.71	0.15	83.06	47.51	58.86	23.00	38.63	247.96	0.00
rat575_20.txt	89.88	73.75	102.02	1.07	7.80	104.80	303.20	10.73	94.64	1800.00	0.00
rat575_30.txt	70.13	8953.88	91.44	0.68	22.71	86.40	196.56	15.94	74.52	1101.33	0.00
rat575_40.txt	61.99	7496.16	92.78	0.35	43.00	72.23	149.33	11.32	64.88	950.51	0.00
rat575_50.txt	55.29	7489.35	104.24	0.24	83.07	66.75	120.46	17.23	56.94	717.39	0.00
rat575_60.txt	49.30	7559.28	71.78	0.25	39.79	59.90	101.12	16.65	51.35	595.10	0.00
rat575_70.txt	43.76	7247.07	63.13	0.19	31.92	57.87	85.15	20.93	47.85	494.17	0.00
rat575_80.txt	41.21	7360.75	75.00	0.37	69.32	53.60	72.63	21.01	44.29	448.23	0.00
rat575_90.txt	37.90	7344.80	72.17	0.18	75.56	51.24	67.33	24.64	41.11	319.13	0.00
rat783_10.txt	160.72	0.44	195.49	5.08	17.60	172.59	1389.10	3.82	166.23	1800.05	0.00
rat783_100.txt	43.69	7428.60	81.02	0.64	76.60	56.85	155.61	23.91	45.88	1019.60	0.00
rat783_20.txt	106.30	261.79	132.20	2.05	17.30	125.60	746.83	11.45	112.70	1800.03	0.00
rat783_30.txt	82.23	7878.63	142.67	2.18	61.09	99.92	498.44	12.82	88.57	1800.01	0.00
rat783_40.txt	75.59	7229.26	139.98	1.55	84.12	86.21	369.54	13.39	76.03	1800.01	0.00
rat783_50.txt	67.11	7285.30	131.24	0.89	98.55	77.88	305.81	17.82	66.10	1800.00	0.00
rat783_60.txt	61.62	7309.73	134.63	1.16	124.32	70.93	258.90	18.18	60.02	1617.55	0.00
rat783_70.txt	55.38	7548.80	92.09	0.78	66.09	67.17	221.20	21.15	55.44	1642.05	0.00
rat783_80.txt	50.66	7498.65	131.24	0.50	154.03	64.62	193.52	25.08	51.66	1420.24	0.00
rat783_90.txt	46.30	7614.66	90.09	0.44	85.88	61.09	185.64	26.05	48.47	1211.55	0.00
rl1323_10.txt	6200.04	1.38	6922.07	44.67	9.63	6345.13	7899.18	0.50	6313.82	1800.23	0.00
rl1323_100.txt	1224.83	7211.82	3778.88	6.08	133.13	2869.67	986.21	77.04	1620.92	1800.01	0.00
rl1323_20.txt	3714.08	2832.87	5253.12	6.37	30.26	4442.68	3683.45	10.16	4032.83	1800.09	0.00
rl1323_30.txt	3062.71	7201.48	4072.01	5.40	27.09	3553.06	2597.18	10.89	3204.16	1800.07	0.00
rl1323_40.txt	2737.57	7219.34	4466.11	5.55	60.96	3243.26	2035.59	16.89	2774.72	1800.06	0.00
rl1323_50.txt	2324.20	7224.81	4072.01	9.89	67.55	2945.78	1690.00	21.21	2430.27	1800.04	0.00
rl1323_60.txt	2077.11	7607.58	4064.00	6.14	89.10	2869.67	1461.26	33.53	2149.14	1800.04	0.00
rl1323_70.txt	1957.03	7496.61	4364.49	5.79	118.53	2869.67	1301.09	43.68	1997.22	1800.04	0.00
rl1323_80.txt	1439.21	7258.54	4415.36	4.85	139.69	2869.67	1177.36	55.78	1842.10	1800.03	0.00
rl1323_90.txt	1332.22	7246.26	4154.19	2.60	137.98	2869.67	1079.63	64.40	1745.58	1800.02	0.00

Table 16
Sensitivity analysis for β parameter of the constructive procedure.

β	RND	0.25	0.5
RND			
0.25	0.05		
0.5	0.96	0.07	
0.75	0.12	0.01*	0.06

Table 18
Sensitivity analysis for $\delta_{\max}$ parameter of the Strategic Oscillation algorithm.

$\delta_{\max}$	0.1	0.2
0.1		
0.2	0.005*	
0.3	0.014*	0.445

Table 17
Sensitivity analysis for τ parameter of the Tabu Search parameter.

τ	1	10	20	30
1				
10	0.00003*			
20	0.00004*	0.168807		
30	0.00008*	0.460302	0.917632	
40	0.00008*	0.034001*	0.255989	0.20262

lutions and this means there would be less diversity in the initial population of solutions. On the other hand, if we analyze now Table 17, $\tau = 1$ confirms that there are significant differences with respect to the other τ values. Obviously, if we fix the maximum number of iterations without improvements as 1, we are limiting the stopping criterion too much. Consequently, we have repeated the Friedman test without this value and we obtain that the ρ-value is 0.274, that is, there are not significant differences among $\tau = \{10, 20, 30, 40\}$. To check this affirmation see Fig. 2 where is showed the evolution of the solution quality for the τ values. We can see that even if $\tau = 1$ is faster, it gets worse values for the objective function. However, larger τ values obtain practically the same objective function values. Finally, observing the Table 18, there are differences are between $\delta_{\max} = 0.1$ and the other values. There are no significant differences between $\delta_{\max} = 0.2$ and $\delta_{\max} = 0.3$ and we opted for $\delta_{\max} = 0.2$ because it is faster (see Table 7).

However, the Friedman test gets a ρ-value of 0.869 in the *tenure* paremeter which implies that this parameter does not ex-

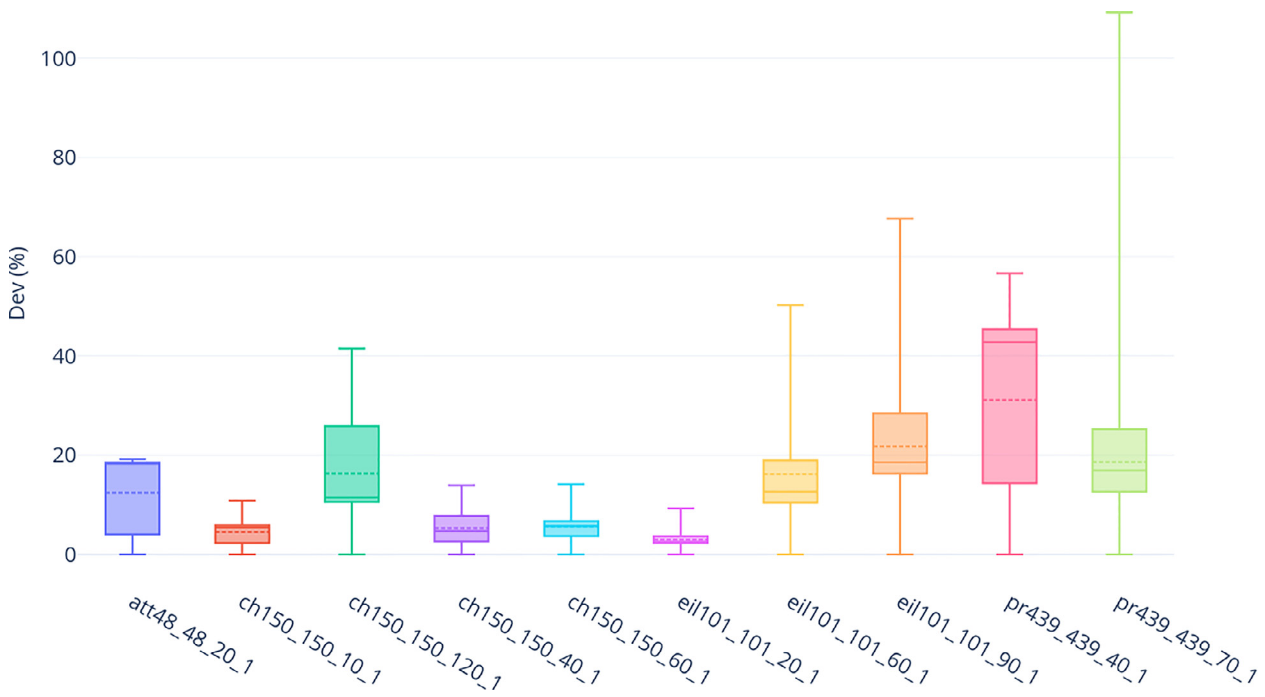

Fig. 5. Box and Whisker plot for 30 independent executions with $\alpha = 3$.

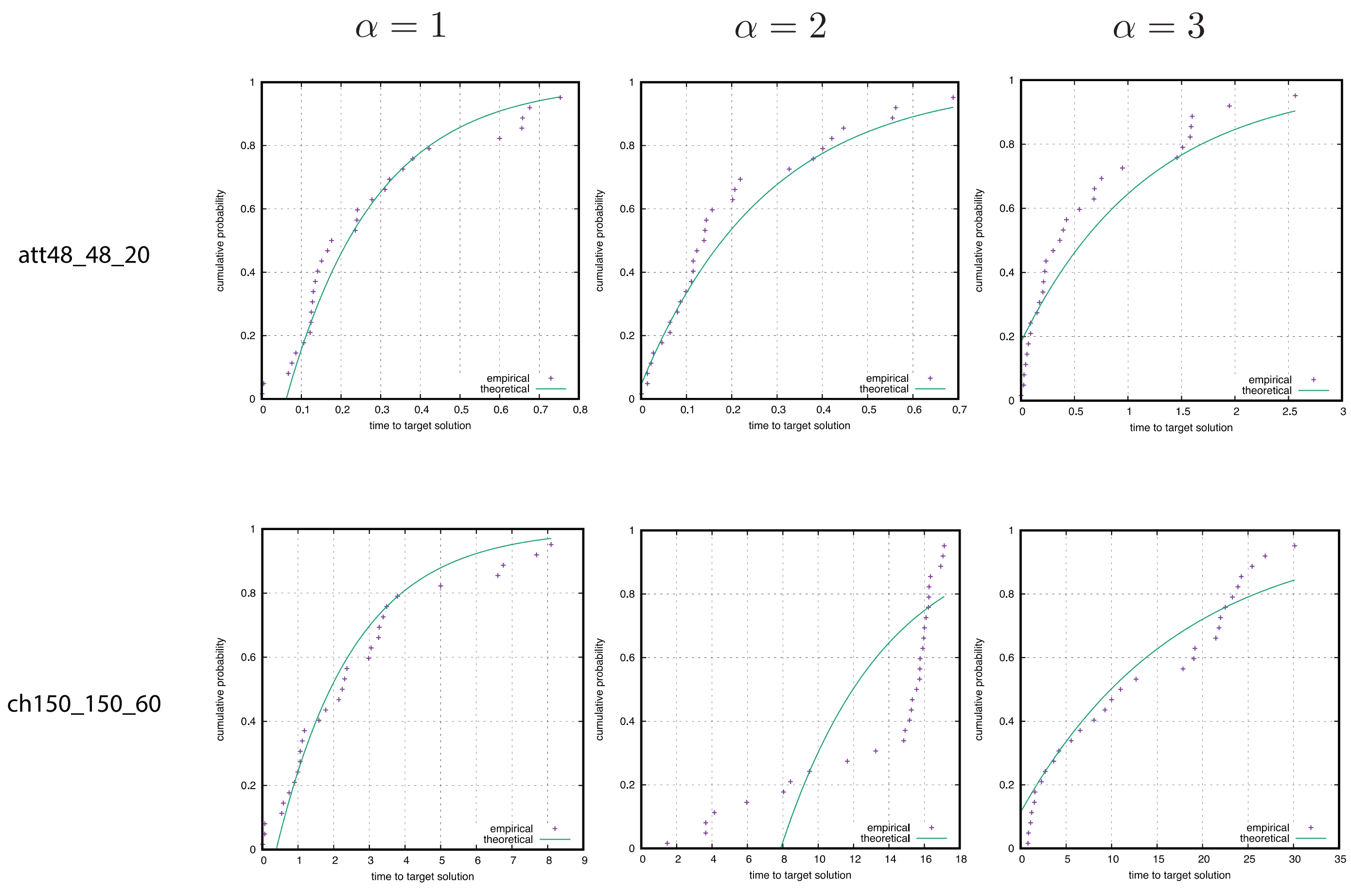

Fig. 6. Time to target plots for two representative instances of the preliminary set.

Table 19
Sensitivity analysis for *tenure* parameter of the Tabu Search improvement.

tenure	0.1	0.2	0.3	0.4
0.1				
0.2	0.55			
0.3	0.43	0.86		
0.4	0.91	0.32	0.28	
0.5	0.75	0.23	0.28	0.31

hibit a particular sensitivity. We recall that the choice of the *tenure* parameter was based on the deviation with respect to the mean and the number of best solutions found since all the values gets the same performance, see Table 5 for details.

To further investigate the performance of the proposed SO procedure, we conduct a convergence analysis by considering time-to-target plots (TTTPlot), which is essentially a run-time distribution (Aiex, Resende, & Ribeiro, 2007). The experimental hypothesis in TTTPlots is that running times fit a two parameter, or shifted, exponential distribution. Then, for a particular instance, the execution time needed to find an objective function value at least as good as a given target value is recorded. In the context of the heuristic optimization, the algorithm is determined a pre-established number of times on the selected instance and using the given target solution. For each run, the random number generator is initialized with a different seed and therefore the executions are assumed to be independent. To compare the empirical and the theoretical distributions, we follow a standard graphical methodology for data analysis (Chambers, 2017), executing our algorithm 30 times, and recording for each instance/target pair the corresponding running time. Figure 6 shows the TTTPlot for a small instance (at_48_20) and a large instance (ch_150_60) with $\alpha = \{1, 2, 3\}$. In these figures, each value in the abscissa axis represents a running time, while each value in the ordinate axis, reports the probability of obtaining the best-known value. This experiment confirms the expected exponential run-time distribution of our algorithm. If we analyze these instances, we can observe that the probability of SO to find a solution at least at good as the target value in half of the CPU time is close to 80%. Indeed, in a quarter of the total CPU time, this probability is near 50% (Table 19).

7. Conclusions

This paper addresses the α-neighbor p-center problem. This variant of the p-center problem considers real situations in which $\alpha - 1$ facilities may become unavailable so the goal is to locate more than one facility close to the demand points to overcome that issue. Therefore, the problem minimizes the maximum distance between each demand point and its $\alpha - th$ closest facility to tackle such type of contingency. Nevertheless, this problem could handle other real situations in which a customer/user (demand point) decides if he/she prefers to be served by another facility instead of the closest one, since the alternative facility still being close to him/her.

To address this problem a *GRASP* algorithm as well as a post-processing method based on Strategic Oscillation are proposed. The improvement for the GRASP algorithm consists in a Tabu Search that is able to find high quality solutions in small computing times. The SO post-processing achieves even better solutions but it is very computationally demanding, being suitable for those situations in which the time limit is not a hard constraint. The algorithms have been tested in a thorough experimentation that shows the superiority of the proposal, emerging *GRASP* combined with *SO* as the most competitive algorithm in the literature for the $\alpha - p$CP. It is worth mentioning that the proposed algorithm gets slower when the value of p increases. Although it is not a common situation for real-life problems, it would be interesting to provide new search strategies able to deal with this drawback efficiently. In order to do so, new fast local improvement methods that are applied before the proposed TS and SO procedures may help the algorithm to obtain high-quality results faster. Indeed, new adaptations of the strategies proposed for the continuous version may have a positive impact on the quality of the solutions provided. Finally, an experimental analysis of the algorithm has been performed to evaluate the impact of each parameter in the final design of the algorithm.

Future lines of research in this problem are focused on modifying some constraints with the aim of improving the solutions obtained. For instance, the set of demand points may not be reduced to $N \setminus P$ and, in this case, the maximum distance to the α-closest facility can be given by a selected demand point even if it is already hosting a facility. Additionally, this model can be adapted to open more than one facility in the same point, or even considering a distributed model, such as in Brimberg et al. (2021) where it is adapted for the p-median problem.

Declaration of Competing Interest

The authors declare that they have no conflict of interest.

Acknowledgments

This research was funded by the Spanish Ministry of “Ciencia, Innovación y Universidades” under grant ref. PGC2018-095322-B-C22, “Comunidad de Madrid” and “Fondos Estructurales” of European Union with grant refs. S2018/TCS-4566 and Y2018/EMT-5062, by the Spanish Ministry of “Economía, Industria y Competitividad through Project ECO2016-76567-C4-1-R”, and by “Junta de Andalucía”, FEDER-UPO Research and Development Call, reference number UPO-1263769.

References

Aiex, R. M., Resende, M. G., & Ribeiro, C. C. (2007). TTT plots: A perl program to create time-to-target plots. *Optimization Letters, 1*(4), 355–366.
Albareda-Sambola, M., Hinojosa, Y., Marín, A., & Puerto, J. (2015). When centers can fail: A close second opportunity. *Computers & Operations Research, 62*(Supplement C), 145–156.
Basu, S., Sharma, M., & Ghosh, P. S. (2015). Metaheuristic applications on discrete facility location problems: A survey. *Opsearch, 52*(3), 530–561.
Brimberg, J., Maier, A., & Schäbel, A. (2021). When closest is not always the best: The distributed p-median problem. *Journal of the Operational Research Society, 72*(1), 200–216.
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. *Journal of the Operational Research Society, 70*(2), 192–211.
Chambers, J. M. (2017). *Graphical methods for data analysis: 0*. Chapman and Hall/CRC.
Chaudhuri, S., Garg, N., & Ravi, R. (1998). The p-neighbor k-center problem. *Information Processing Letters, 65*(3), 131–134.
Chen, D., & Chen, R. (2013). Optimal algorithms for the alpha-neighbor p-center problem. *European Journal of Operational Research, 225*(1), 36–43.
Daskin, M. S. (1995). *Network and discrete location*. John Wiley & Sons, Inc..
Daskin, M. S., & Maass, K. L. (2015). The p-median problem. In *Location science* (pp. 21–45). Springer.
Elshaikh, A., Salhi, S., Brimberg, J., Mladenović, N., Callaghan, B., & Nagy, G. (2016). An adaptive perturbation-based heuristic: An application to the continuous p-centre problem. *Computers & Operations Research, 75*, 1–11.
Feo, T. A., & Resende, M. G. C. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters, 8*(2), 67–71.
Feo, T. A., Resende, M. G. C., & Smith, S. H. (1994). A greedy randomized adaptive search procedure for maximum independent set. *Operations Research, 42*(5), 860–878.
Ferone, D., Festa, P., Napoletano, A., & Resende, M. G. C. (2017). A new local search for the p-center problem based on the critical vertex concept. In R. Battiti, D. E. Kvasov, & Y. D. Sergeyev (Eds.), *Lion*. In *Lecture notes in computer science: vol. 10556* (pp. 79–92). Springer.
Festa, P., & Resende, M. G. C. (2008). An annotated bibliography of GRASP-Part I: Algorithms. *International Transactions in Operational Research, 16*(1), 1–24.
Festa, P., & Resende, M. G. C. (2009). An annotated bibliography of GRASP-Part II: Applications. *International Transactions in Operational Research, 16*(2), 131–172.

Glover, F., & Laguna, M. (1998). Tabu search. In *Handbook of combinatorial optimization* (pp. 2093–2229). Springer.

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

Khuller, S., Pless, R., & Sussmann, Y. J. (2000). Fault tolerant k-center problems. *Theoretical Computer Science, 242*(1), 237–245.

Krumke, S. (1995). On a generalization of the p-center problem. *Information Processing Letters, 56*(2), 67–71.

López-Sánchez, A., Sánchez-Oro, J., & Hernández-Díaz, A. (2019). GRASP and VNS for solving the p-next center problem. *Computers & Operations Research, 104*, 295–303.

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

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

Murray, A. T. (2016). Maximal coverage location problem: Impacts, significance, and evolution. *International Regional Science Review, 39*(1), 5–27.

Pérez-Peló, S., Sánchez-Oro, J., López-Sánchez, A. D., & Duarte, A. (2019). A multi-objective parallel iterated greedy for solving the p-center and p-dispersion problem. *Electronics, 8*(12), 1440.

Salhi, S. (1997). A perturbation heuristic for a class of location problems. *Journal of the Operational Research Society, 48*(12), 1233–1240.

Sánchez-Oro, J., Pantrigo, J. J., & Duarte, A. (2014). Combining intensification and diversification strategies in VNS. An application to the vertex separation problem. *Computers & Operations Research, 52*(Part B), 209–219. Recent advances in Variable neighborhood search

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. *Journal of Computer Science and Technology, 32*(6), 1319–1334.