Expert Systems With Applications 175 (2021) 114728

Contents lists available at ScienceDirect

Expert Systems With Applications

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

An evolutionary approach for the p-next center problem

Mariana A. Londe [a], Carlos E. Andrade [b], Luciana S. Pessoa [a,*]

[a] Department of Industrial Engineering, PUC-Rio, Rua Marquês de São Vicente, 225, Gávea, 22453-900 Rio de Janeiro, RJ, Brazil
[b] AT&T Labs Research, 5312 SW 6th Pl, Ocala, FL 34474, USA

ARTICLE INFO

Keywords:
Facility location
Biased Random-Key Genetic Algorithm
Metaheuristics

ABSTRACT

The p-next center problem is an extension of the classical p-center problem, in which a backup center must be assigned to welcome users from a suddenly unavailable center. Usually, users tend to seek help in the closest facility they can find. However, during a significant event or crisis, one only realizes that the closest facility has been disrupted upon his/her arrival. Therefore, the user seeks help in the next closest center from the one that has failed to provide service. Therefore, the objective of the p-next center problem is to minimize the path of any user, which is made by the distance from this origin to its closest installed facility, plus the distance from this facility to its backup. We propose an evolutionary approach for the p-next center problem and an extension for the current benchmark instances. The proposed methods are built on the Multi-Parent Biased Random-Key Genetic Algorithm with Implicit Path-Relinking. Computational experiments carried out on 416 test instances show, experimentally, the outstanding performance of the developed algorithms and their flexibility to reach a good quality-speed trade-off.

1. Introduction

Given a set $A = \{1, \ldots, n\}$ of points which represent users and also the candidate locations to install a facility, let d_{ij}, for $i, j \in A$, be the travel distance from the users in i to the candidate location j. The number of centers to be installed is p. The *p-center problem* consists of choosing p among n candidate locations so that the maximum distance from any user in A to its closest installed facility is minimized (Kariv & Hakimi, 1979; Tansel, Francis, & Lowe, 1983).

The *p-next center problem* is an extension of the p-center problem in which there is the possibility of the users arrive in the facility assigned to them and find out a disruption in its operation. In this situation, the users must move to a backup point, which is the facility installed nearest to the one initially assigned. The objective of the *p-next center problem* is to minimize the path of any user in A, which is made by the distance of the point i to its closest installed facility, *plus* the distance from this facility to its backup. This problem, referred as pNCP, was introduced by Albareda-Sambola, Hinojosa, Marín, and Puerto (2015) in the context of humanitarian logistics, taking into account the substantial possibility of an emergency service facility become inactive. The problem definition was presented together with a variety of formulations and the proof of its NP-hardness. Later, López-Sánchez, Sánchez-Oro, and Hernández-Díaz (2019) approached the pNCP by developing heuristic methods based on GRASP (Greedy Randomized Adaptive Search Procedure) and VNS (Variable Neighborhood Search) metaheuristics (Gendreau & Potvin, 2010). To the best of our knowledge, no other study was devoted to the p-next center problem.

This study contributes to the literature by presenting an approach based on evolutionary programming to solve this problem efficiently. In this way, the Multi-Parent Biased Random-Key Genetic Algorithm (BRKGA-MP) is explored for solving the p-next center problem. Additionally, we propose an extension of the current benchmark instances to evaluate the algorithms' performance. Finally, upper and lower bounds are reported for all instances so that these values can be applied to conduct further studies.

The remainder of this article is organized as follows. Section 2 brings the problem formulation together with the description of related works. The proposed method is described in Section 3. Experimental results are reported in Section 4. Section 5 closes the paper with concluding remarks.

2. Related work and problem formulation

The p-center problem (pCP) is among the most studied facility location problem and it was proven to be NP-hard by Kariv and Hakimi (1979). In the services context, the pCP considers n locations which

* Corresponding author.
E-mail addresses: mlonde@aluno.puc-rio.br (M.A. Londe), cea@research.att.com (C.E. Andrade), lucianapessoa@puc-rio.br (L.S. Pessoa).

https://doi.org/10.1016/j.eswa.2021.114728
Received 9 July 2020; Received in revised form 11 December 2020; Accepted 11 February 2021
Available online 23 February 2021

represent users as well as potential locations where to install service center. The resolution of this problem aims of choosing *exactly p* among *n* locations so that to minimize the maximum distance from any user to its closest installed service center (Kariv & Hakimi, 1979; Tansel et al., 1983). Recently, the disaster responses have emerged with important practical applications for the p-center problem such as in medical services for large-scale emergencies (Huang, Kim, & Menezes, 2010; Jia, Ordóñez, & Dessouky, 2007), healthcare logistics (Ahmadi-Javid, Seyedi, & Syam, 2017), and humanitarian relief distribution (Shavarani, 2019).

It was, indeed, in the humanitarian logistics field that Albareda-Sambola et al. (2015) described the *p*-next center problem (pNCP) so that to deal with a probable disruption in the service center operation. The *p*-next center problem (pNCP) is, therefore, related to other facility location problems whose solutions must guarantee a certain level of service even in the event of disruptions or temporal unavailability of the facility.

Hogan and ReVelle (1986) describes the concept of backup coverage applied to the maximal covering location problem (Church & ReVelle, 1974) and the location set covering problem (Toregas, Swain, ReVelle, & Bergman, 1971). Besides, for reliability purposes, the set covering problem can be modeled as the set k-covering problem (Pessoa, Resende, & Ribeiro, 2011; Pessoa, Resende, & Ribeiro, 2013) or the set multi-covering problem (Hall & Hochbaum, 1992) by the addition of redundancy constraints, so that each user is covered by, at least, *k* or a variable number of facilities. Backup coverage is also approached by Karatas and Yakici (2019) and Karatas, Razi, and Tozan (2016) on the p-median and the maximal covering location problems where the so called "Q-coverage" guarantees that a user is primary covered by one facility plus $Q-1$ backup facilities. Yet, the *K*-center problem (Hochbaum & Shmoys, 1985) has its fault tolerance counterpart problem described by Khuller, Pless, and Sussmann (2000). The *K*-center problem slightly differs from the pCP by requiring that *at most K* (instead of *exactly K*, or *p*) facilities must be installed and each user is assigned to some facility. By applying the backup coverage concept on it, each user must be assigned to a subset of *K* facilities.

Expanding the literature and the applications on location problems with backup facilities, Albareda-Sambola et al. (2015) proposed the p-next center problem motivated by a real situation occurred in a town of Spain, in 2011, when two consecutive earthquakes led to the evacuation of an hospital. In such a situation the users must be immediately redirected to the closest service centers.*Problem formulation* According to Albareda-Sambola et al. (2015), the p-next center problem can be modeled by a two-indexed formulation considering the binary variables as below. Note that we use *i*,*j*, and *k* to refer either to centers or users.

$$y_j = \begin{cases} 1, & \text{if a facility is installed in } j \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{if the closest center to } i \text{ is } j \\ 0, & \text{otherwise.} \end{cases}$$

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

$$\text{s.t.} \sum_{j=1}^{n} y_j = p \tag{2}$$

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} x_{ij} = 1 \quad \forall i \in \{1,\ldots,n\}, \tag{3}$$

$$x_{ij} \leqslant y_j \quad \forall i, \quad j \in \{1,\ldots,n\}, \quad i \neq j, \tag{4}$$

$$y_j + \sum_{\substack{k=1 \\ d_{ik}>d_{ij}}}^{n} x_{ik} \leqslant 1 \quad \forall i,j \in \{1,\ldots,n\}, \quad i \neq j, \tag{5}$$

$$z \geqslant sum_{\substack{k=1 \\ k\neq j}}^{n} d_{jk} \cdot x_{jk} \quad \forall j \in \{1,\ldots,n\}, \tag{6}$$

$$z \geqslant d_{ij} \cdot (x_{ij} - y_i) + \sum_{\substack{k=1 \\ k\neq j}}^{n} d_{jk} \cdot x_{jk} \quad \forall i,j \in \{1,\ldots,n\}, \quad i \neq j, \tag{7}$$

$$y_j \in \{0,1\} \quad \forall j \in \{1,\ldots,n\}, \tag{8}$$

$$x_{ij} \in \{0,1\} \quad \forall i,j \in \{1,\ldots,n\}, \quad i \neq j. \tag{9}$$

Note that variable x_{ij} can have different meanings, either if *i* is a user or a center. In the former case, the variable indicates the assignment of a center to a user. In the latter, x_{ij} indicates the backup center to an existing center. The Objective Function (1) indicates that the model aims to minimize the highest total assignment cost amongst all nodes. If the node is an user, the assignment cost is the sum of the distances between user and reference center, and reference center and backup center. If the node is a center, the assignment cost is the distance between the reference center itself and its backup. Constraint (2) indicates that only *p* centers must exist. Constraints (3) and (4) guarantee that each node is assigned a reference center while preventing self-assignment for user nodes. Meanwhile, Constraint (5) imposes the minimal distance when allocating a user to his/her reference center, as well as to his/her the backup center. Constraint (6) ensures that z will always be higher or equal to the distance between a reference center and its backup. Meanwhile, Constraint (7) guarantees that z will, also, be higher or equal to the distance between a user and its backup center. Finally, Constraints (8) and (9) define the domain of the variables.

Albareda-Sambola et al. (2015) compared this formulation against other proposed models based on three-indexed variables as well as on covering variables. The computational experiments considered 132 instances, generated from the classical OR-Library benchmark `p-med` instances, defined by Beasley (1990), with *n* up to 200 while the maximum value of *p* is equal to 100. Indeed, the sets `p-med1`–`p-med4` and `p-med6`–`p-med8` originated the instances studied in their work. All mathematical models were solved using Fico Xpress-Optimizer with a time limit of two or four hours depending on the instances dimensions. The results showed the superiority of the above described two-indexed formulation, especially when solving instances with $n \geqslant 150$. Moreover, the three-indexed variable model was capable to consistently find optimal solutions for small instances ($n \leqslant 50$) in less than 20 min.

Recently, López-Sánchez et al. (2019) proposed a Greedy Randomized Adaptive Search Procedure (GRASP) to solve the pNCP. Additionally, a Variable Neighborhood Search (VNS) algorithm was proposed. GRASP is an iterative two-phase process: a construction phase, where a feasible solution is generated by a randomized greedy algorithm and, a local search phase that attempts to improve the incumbent solution provided by the first phase. VNS, on its turn, coordinates different local search procedures to explore the solution space, from an initial solution while generating a chain of improving solutions successively. The authors also proposed a hybrid method in order to mix the strengths of the GRASP and the VNS metaheuristics. Computational experiments were carried out on the same set of instances developed by Albareda-Sambola et al. (2015) (albeit maximum $p = 80$). The results demonstrated the

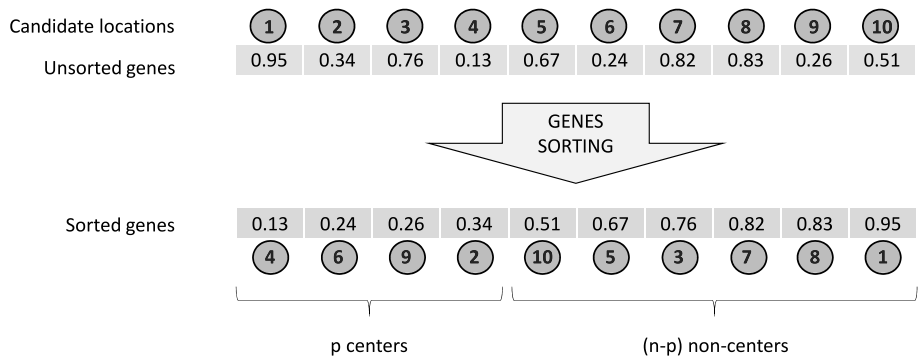


Fig. 1. Chromosome representation and decoder for the p-next center.

efficiency of the proposed hybrid heuristic, which was able to find optimal solutions for small instances ($n \leqslant 50$) in a few seconds. Despite showing the superiority of the hybrid method compared to the pure ones, no comparison was provided for larger instances concerning the optimal solutions.

To the best of our knowledge, Albareda-Sambola et al. (2015) and López-Sánchez et al. (2019) are the only authors to study the p-next center problem despite its importance in scenarios of uncertainty such as those found in humanitarian operations.

The research on the pNCP, therefore, enriches the literature on facility location (Brandeau & Chiu, 1989; Farahani & Hekmatfar, 2009) which can find application in healthcare management (Güneş, Melo, & Nickel, 2019; Ahmadi-Javid et al., 2017), in the deployment of services in urban areas (Farahani, Fallah, Ruiz, Hosseini, & Asgari, 2019), in supply chain management (Aikens, 1985; Melo, Nickel, & Saldanha-Da-Gama, 2009), as well as in humanitarian logistics planning (Boonmee, Arimura, & Asada, 2017), among other fields.

3. Solving the pNCP using a hybrid biased Random-Key Genetic Algorithm

Among several methods to solve facility location problems (such as GRASP and VNS mentioned in Section 2), evolutionary approaches have achieved outstanding performance considering solution quality and running time, especially on humanitarian relief applications (Jia, Ordóñez, & Dessouky, 2007). Recently, Mousavi, Bahreininejad, Musa, and Yusof (2017) presented a particle swarm algorithm for a complex two-echelon facility location problem. Krömer, Platoš, and Snášel (2016) presented a compendium of differential evolution algorithms applied to facility location problems. A multi-objective biogeography-based optimization was proposed in Jalali, Seifbarghy, Sadeghi, and Ahmadi (2016), and the authors compared their results with the classic non-dominated ranking genetic algorithm (NRGA) and multi-objective simulated annealing. Rabie (2020) proposed a grey wolf optimizer and a particle swarm optimization for the p-median location problem where the latter presents better performance than the former in most cases. Finally, Alcaraz, Landete, Monge, and Sainz-Pardo (2020) presents two evolutionary strategies for the reliability fixed-charge location problem.

Our approach to the pNCP is built on the Multi-Parent Biased Random-Key Genetic Algorithm with Implicit Path-Relinking (BRKGA-MP-IPR, Andrade, Toso, Gonçalves, & Resende (2021)), a variant of the standard BRKGA proposed by Gonçalves and Resende (2011). This method was chosen due to its suitability for location problems, as shown by Resende (2012), Lopes, de Andrade, de Queiroz, Resende, and Miyazawa (2016), Pessoa, Santos, and Resende (2017), Biajoli, Chaves, and Lorena (2019). BRKGA also has been used with success to other traditional combinatorial optimization problems such as packing (Gonçalves & Resende, 2011; Gonçalves & Resende, 2013), scheduling (Almeida, Correia, & Saldanha-da Gama, 2018; Andrade, Silva, & Pessoa, 2019; Pessoa & Andrade, 2018; Brandão, Noronha, Resende, & Ribeiro, 2017; Andrade et al., 2019; Andrade et al., 2019), routing (Martinez, Loiseau, Resende, & Rodriguez, 2011; Ruiz, Soto-Mendoza, Ruiz Barbosa, & Reyes, 2019), and network design (de Faria, Resende, & Ernst, 2017; Andrade et al., 2015). It has been hybridized with mixed-integer programming techniques (Andrade, Ahmed, Nemhauser, & Shao, 2017) and machine learning tools (Paliwal et al., 2019). Additionally, such *population-based* approach, dealing with multiple solutions at a time, contrasts with the *trajectory-based* method developed by López-Sánchez et al. (2019), in which only one solution is explored at each iteration. In this way, we explore another algorithmic perspective aiming to find better solutions for the p-next center problem.

3.1. Evolutionary process

The biased random-key genetic algorithm (BRKGA) represents a pool of solutions, called population $\mathscr{P}$, using a set of real-value vectors in the interval $[0,1]$, called chromosomes. Each gene of a chromosome is a real-value number in $[0, 1]$. Note that $\mathscr{P}$ is, in fact, a continuous unitary hypercube. Although the real-value gene is most common, some works such as Andrade, Resende, Karloff, and Miyazawa (2014), use alternative representations (such as integer numbers and bit-strings) to prune the hypercube, reducing the search space, and therefore, speeding up the search.

Initially, the BRKGA creates the first generation of p individuals, by attributing a random number uniformly drawn from $[0, 1]$ to each of its genes. Following, a *decoder* procedure maps these vectors in the space of feasible solutions of the pNCP, computing the objective function value of the solutions. Such values are known as *fitness of the individual*. At each generation, each individual is placed in the elite set $\mathscr{P}_e$, or in the non-elite set $\mathscr{P}_{ne}$, according to its classification after the whole population is sorted by the non-increased order of the individuals' fitness. The highest quality solutions compound the elite set $\mathscr{P}_e$, while the remaining are non-elite solutions $\mathscr{P}_{ne}$.

In the standard BRKGA, three steps are followed to create a new

population: *elitism*, *reproduction*, and *mutants generation*. The *elitism* step is accomplished by simply copying all $\mathscr{P}_e$ individuals to the population of the next generation, giving us the first biased aspect from the method. *Mutants generation* injects p_m new chromosomes in the new population by randomly chosen values for the genes, as occurred to the first population. The remain $|\mathscr{P}_e| - p_m$ individuals are offspring, generated from the mating between two parents randomly chosen: one from $\mathscr{P}_e$ and another from $\mathscr{P}_{ne}$. The second biased aspect from the method comes from the probability $\rho > 0.5$ that the *elite parent* gives its gene value to the offspring.

In the Multi-Parent Biased Random-Key Genetic Algorithm (BRKGA-MP), proposed by Lucena, Andrade, Resende, and Miyazawa (2014), the mating is done among several parents rather than two as in the standard BRKGA. For that, BRKGA-MP randomly selects π_e parents from the elite and $\pi_t - \pi_e$ parents from the non-elite set. Once ranked by their fitness, BRKGA-MP applies the bias function Φ, assigning probabilities for each parent. Then, each gene is taken from a parent according to its probability using the roulette method. Therefore, while in the standard BRKGA, the parameter ρ controls the elite parent's bias, in the BRKGA-MP, the total number of mating parents π_t, the number of mating parents from the elite π_e, and the bias function Φ plays that role. For more details, refer to Andrade et al. (2021).

After the accomplishment of the elitism, reproduction, and mutants generation steps, a new cycle of evaluation-selection-evolution takes place repeatedly until the algorithm hits a stopping criterion, i.e., the BRKGA runs while a pre-specified number of generations or a certain processing time is not reached.

3.2. Chromosome representation and decoder

A chromosome that represents a solution for the pNCP is a vector with n genes, where each gene in the chromosome is a real value number in the interval $[0, 1]$.

Fig. 1 illustrates this encoding for a pNCP instance containing $n = 10$ candidate locations among which $p = 4$ facilities are to be installed. Besides, the figure shows the operation of the decoder. Initially, all n genes are labeled $1, \ldots, n$, and sorted in ascending order of their respective genes/keys. Then, the first p genes are chosen as reference centers. The last $n - p$ locations (users) are assigned to its closest reference center, which has an associated backup center.

After decoding the random-key vector into a solution, the decoder computes the solution cost (or fitness) as follows: for each user, the algorithm evaluates the total distance from him/her to his/her reference center plus the distance from such a reference center to its backup center. Then, the maximum total distance verified among all users defines the solution cost.

3.3. Exploitation: local search and the implicit path-relinking

BRKGA can benefit from intensification strategies to explore the neighborhood of a solution aiming to reach better quality solutions.

A simple approach developed for the pNCP is a local search procedure that makes 1–1 exchanges between locations assigned to reference centers and the remaining potential locations of a solution. This method is applied in the solution space, i.e., after the random-keys decoding. Once a locally optimum solution is reached, its associated chromosome is adjusted to reflect the modifications performed. The search strategy can be based on first or best improvement, while the number of exchanges, is defined as a percentage of the size of the chromosome.

A more sophisticated intensification strategy refers to the Implicit Path-Relinking (IPR) (Andrade et al., 2021), which explores paths in the BRKGA representation space, connecting a base solution to a guide solution extracted from the population. In this way, intermediate solutions are discovered through the path by incorporating characteristics from the guiding random-key vector (v_g) to the base one (v_b).

The *permutation-based implicit path-relinking*, described by Andrade et al. (2021), is applied for the pNCP. In order to explain its implementation, consider $v_b = [0.1, 0.5, 0.9, 0.3, 0.7]$ and $v_g = [0.8, 0.4, 0.2, 0.6, 0.1]$ corresponding, respectively, to the solutions $S_b = [1, 4, 2, 5, 3]$ and $S_g = [5, 3, 2, 4, 1]$. We note that $(S_b[1] = 1) \neq (S_g[1] = 5)$. In order to make $S_b[1] = 5$, a move is performed on v_b swapping the random-key value in the position 1 of v_b (0.1) and the random-value of its 5th position, resulting in $v_b = [0.7, 0.5, 0.9, 0.3, 0.1]$ and, consequently, $S_{b1} = [5, 4, 2, 1, 3]$. A different situation is found in $S_b[3]$ which is equal to $S_g[3]$ so that no modification is carried out. All solutions generated in a step of the path have their costs evaluated and the best one replaces the original S_b. The algorithm then swaps S_g and S_b, and repeats the procedure. These iterations are repeated until the desired path size is reached.

It worth mentioning that S_b and S_g must be different enough to prevent short paths. Such difference can be evaluated by Kendall tau rank distance (Kendall, 1938), which computes the number of pairwise divergences between two ranking lists. In order to avoid unpromising path-relinking calls, a minimum distance parameter is set for the BRKGA execution. On the other hand, another parameter may also be useful to restrict the path exploration within a certain limit of steps.

3.4. Exploration: shake and reset

Generally, most genetic algorithms tend to quickly converge to local minima/maxima, limiting the exploration of the solution space. To mitigate such a situation, traditional genetic algorithms use mutation of the new individuals after mating, and BRKGAs inject mutants into the population of the next generation. However, even these mechanisms may not be sufficient to escape from such regions. To couple with convergence, Andrade et al. (2019) have introduced the shake mechanism. Such a feature applies controlled perturbations to the elite individuals, allowing the algorithm to escape from local minima/maxima and improving the exploration.

We choose to apply a simple shake mechanism described in Algorithm 1. For each elite individual ε in population $\mathscr{P}$ (line 1), we generate a solution $\mathscr{S}$ (line 2). The perturbation is applied as many times as the intensity ψ dictates (line 3), swapping a center location i by a non-center location j, both uniformly chosen from all locations (line 4). Once the perturbations are applied, we re-encode the elite solution back to the population (line 5). We then replace the remaining non-elite individuals by new random individuals (line 6).

Although the shake is very effective in escaping valleys or peaks, sometimes it is not enough. In such cases, we reset the population completely, replacing all individuals for new random solutions. We applied shake and reset in different situations, as described in Section 3.5.

Algorithm 1: Shake for p-next.

Input : Population $\mathcal{P}$ where $\mathcal{P}_e$ is the elite set and $\mathcal{P}_{ne}$ is the non-elite set; shaking intensity ψ.
Output : Changed population.

1 **foreach** *elite individual* $\varepsilon \in \mathcal{P}_e$ **do**
2 Let $\mathcal{S}$ be the solution induced by ε;
3 **for** $k \leftarrow 1$ **to** ψ **do**
4 Swap a center $i \in \mathcal{S}$ and a non-center $j \in \mathcal{S}$, uniformly random chosen;
5 Encode $\mathcal{S}$ back into $\mathcal{P}$;
6 Replace the members of $\mathcal{P}_{ne}$ with random samples;
7 **return** *changed population* $\mathcal{P}$.

3.5. Main loop

While the evolutionary mechanism of BRKGA has obtained excellent results for several problems, usually such a tool alone may not express its full power. Therefore, other intensification and diversification mechanisms have been used to great hybrid BRKGAs.

In this work, we use, as intensification mechanisms, the local search on the "exchange" neighborhood, and the implicit path-relink procedure, described on Section 3.3. For diversification, we use the concept of shake and the full population reset, described on Section 3.4.

Algorithm 2 shows the main loop of our algorithm. Table 1 serves as a companion listing Algorithm 2 attributes, variables, and counters. Without loss of generality, we have simplified some notations, mainly when we deal with solutions. We freely use $\mathscr{S}$ to represent a solution as much as its value. Therefore, when using relational operators, we compare the solution values, and we use the solution structure itself when using assignment operators. Note also that we consider that the BRKGA mechanism is evolving multiple independent populations in parallel, and we use both $\mathscr{P}$ indicating one population and $\mathscr{P}$s indicating several populations loosely, depending on the context.

In line 1, the algorithm invokes BRKGA evolutionary mechanism over the current population(s) $\mathscr{P}$ and extracts from this (these) population(s), the current best solution $\mathscr{S}_{\mathscr{P}}$ for that generation. In lines 2–9, the algorithm checks whether $\mathscr{S}_{\mathscr{P}}$ is better than the overall best solution $\mathscr{S}^{\star}$. If it is the case, a local search is performed on the solution if the local search is enable. If the local search finds an improved solution $\mathscr{S}_{\ell}$, the algorithm updates $\mathscr{S}_{\mathscr{P}}$ and injects $\mathscr{S}_{\ell}$ back to the population replacing the worst elite individual/solution $\mathscr{W}_e$ on the (first) population (lines 5–7). Note that we need to re-encode $\mathscr{S}_{\ell}$ to be conforming to the BRKGA population. We then update the best overall solution $\mathscr{S}^{\star}$ and holds the iteration when this occurs (lines 8–9). Since we use shake and reset mechanisms together, we also keep track of the best solution found between hard population resets ($\mathscr{S}^{\star}_{\mathscr{P}}$) and when it is updated (lines 10–12).

Every other I_{ipr} iterations without improvement on overall best solution $\mathscr{S}^{\star}$, we call the implicit path-relinking (IPR) for further intensification (lines 13–15). Such a procedure is a sophisticated and time-demanding algorithm with several control parameters, as described in Andrade et al. (2021). Section 4.1 details such parameter setting. After the application of IPR, if a better solution is found, we update $\mathscr{S}^{\star}$.

Table 1
Main algorithm attributes and control variables.

Parameter	Description
$\mathscr{P}$/ $\mathscr{P}$s	Current population(s)
$\mathscr{S}_{\mathscr{P}}$	Current generation best solution
$\mathscr{S}^{\star}_{\mathscr{P}}$	Current population best solution (between resets)
$\mathscr{S}_{\ell}$	Solution from local search
$\mathscr{B}_e$	Best individual of current elite set
$\mathscr{W}_e$	Worst individual of current elite set
$\mathscr{S}^{\star}$	Best overall solution
total_iter	Total number of iterations
pop_iter	Number of iterations on the current population
lui	Last update iteration of $\mathscr{S}^{\star}$
pop_lui	Last update iteration of $\mathscr{S}^{\star}_{\mathscr{P}}$
noimpr	Iterations without improvement on $\mathscr{S}^{\star}$
pop_noimpr	Iterations without improvement on $\mathscr{S}^{\star}_{\mathscr{P}}$
I_{ipr}	Number iterations for path-relink calling
I_{s}	Number iterations for shake calling
S_{m}	Shake iterations multiplier
R_{m}	Reset iteration multiplier

BRKGA is known for its very fast convergence. Therefore, other diversification mechanisms are necessary to improve the algorithm, namely, shake and reset. Our shake procedure is detailed on Section 3.4, and we apply it in three situations. In the first situation, we check for population diversity, comparing the values of the best and worst solution in the elite set of each population (lines 16–18). In case these values are the same, we have a homogeneous elite set, and we apply a low-intensity shake. The second situation happens when the best solution found between hard population resets ($\mathscr{S}^{\star}_{\mathscr{P}}$) has not been updated for I_{s} iterations. In such a scenario, a wide-variable intensity shake is applied (lines 19–22). Finally, when the best overall solution $\mathscr{S}^{\star}$ is stalled for $\mathrm{I}_{s} \times \mathrm{S}_{m}$ iterations, the algorithm uses a high-intensity shake (lines 23–24). Note that in all cases, the intensity is given as a function of the population size and a number uniformly drawn from an interval according to the situation. For that, we use the function `rng`. Although the intervals are hard-coded in the algorithm, we have tuned them carefully.

Algorithm 2: Algorithm main loop. Acronyms on Table 1.

Evolve all $\mathcal{P}$s for one generation. Let $\mathcal{S}_{\mathcal{P}}$ be the best solution among all $\mathcal{P}$s;

`// Update best overall solution.`
if $\mathcal{S}_{\mathcal{P}} < \mathcal{S}^{\star}$ **then**
  if *local search is active* **then**
    Perform local search on $\mathcal{S}_{\mathcal{P}}$. Let $\mathcal{S}_{\ell}$ be the generated solution;
    if $\mathcal{S}_{\ell} < \mathcal{S}_{\mathcal{P}}$ **then**
      $\mathcal{S}_{\mathcal{P}} \leftarrow \mathcal{S}_{\ell}$;
      Replace $\mathcal{W}_e(\mathcal{P})$ by $\mathcal{S}_{\ell}$;
  $\mathcal{S}^{\star} \leftarrow \mathcal{S}_{\mathcal{P}}$;
  $lui \leftarrow total\_iter$;

`// Update current population trackers.`
if $\mathcal{S}_{\mathcal{P}} < \mathcal{S}^{\star}_{\mathcal{P}}$ **then**
  $\mathcal{S}^{\star}_{\mathcal{P}} \leftarrow \mathcal{S}_{\mathcal{P}}$;
  $pop\_lui \leftarrow pop\_iter$;

`// Call path-relinking.`
$noimpr \leftarrow lui - total\_iter$;
if $noimpr > 0$ **and** $noimpr \bmod I_{ipr} = 0$ **then**
  Call implicit path-relinking, and update $\mathcal{S}^{\star}$ as necessary;

`// Diversity/weak shake when elite is too homogeneous.`
foreach *population* $\mathcal{P}$ **do**
  if $\mathcal{B}_e(\mathcal{P}) = \mathcal{W}_e(\mathcal{P})$ **then**
    Shake $\mathcal{P}$ with intensity $|\mathcal{P}| \cdot \texttt{rng}(0.05, 0.5)$;

`// Stalled population shake.`
$pop\_noimpr \leftarrow pop\_iter - pop\_lui$;
if $pop\_noimpr > 0$ **and** $pop\_noimpr \bmod I_s = 0$ **then**
  Shake all $\mathcal{P}$s with intensity $|\mathcal{P}| \cdot \texttt{rng}(0.0, 1.0)$;
  $pop\_lui \leftarrow 0$;

`// Stalled best solution shake.`
if $noimpr > 0$ **and** $noimpr \bmod (I_s \cdot S_m) = 0$ **then**
  Shake all $\mathcal{P}$s with intensity $|\mathcal{P}| \cdot \texttt{rng}(0.5, 1.0)$;

`// Hard reset.`
if $noimpr > 0$ **and** $noimpr \bmod (I_s \cdot S_m \cdot R_m) = 0$ **then**
  Reset all populations;
  $pop\_iter \leftarrow 0$;
  $pop\_lui \leftarrow 0$;

$pop\_iter \leftarrow pop\_iter + 1$;
$total\_iter \leftarrow total\_iter + 1$;

Table 2
Algorithms' parameters tuned by `irace`.

Algorithm	BRKGA					IPR			Shake/Reset			LS%
	$\lvert\mathscr{P}\rvert$	$\mathscr{P}_e$%	$\mathscr{P}_m$%	π_e, π_t	Φ	*md*	*ps*%	I_{ipr}	I_s	S_m	R_m	
`BRKGA-NLS`	3000	0.19	0.25	1,3	r^{-3}	0.13	0.40	270	100	1.79	1.88	0.50
`BRKGA-FI`	4500	0.19	0.12	7,10	r^{-2}	0.10	0.79	230	84	1.60	1.19	0.57
`BRKGA-BI`	4000	0.27	0.22	4,6	r^{-2}	0.20	0.92	120	85	1.76	1.87	0.45

Some times, even diversification methodologies such as shake are not enough to escape from local minima. Therefore, every other $I_s \times S_m \times R_m$ iterations without improvement in the best overall solution $\mathscr{S}^{\star}$, we fully reset all populations, restarting the evolutionary procedure again (lines 25–28).

Note that the shake and reset applications are interrelated based on the offset of iterations without a change in the tracked solutions. Although it is not necessary, we assume that $S_m \geqslant 1$ and $R_m \geqslant 1$, creating a "ladder" for different diversification processes. For example, if $I_s = 100$, $S_m = 1.50$, and $R_m = 1.80$, the stalled-population shake is called within 100 iterations (without improvements). The stalled-best-solution shake is called within 150 iterations, and finally, the population is reset within 270 iterations.

Finally, the algorithm updates the counters on lines 29 and 30. The algorithm repeats this main loop until some stopping criteria are met. In this work, we use only (wall-clock) maximum time as stopping criterion.

4. Experimental results

4.1. Computational environment

The computational experiments were performed in a cluster of identical machines with an Intel Xeon E5530 CPU at 2.40 GHz and 120 GB of RAM running CentOS Linux. Heuristics proposed in this paper were implemented in C++ language using the BRKGA-MP-IPR framework (Andrade et al., 2021). The formulation proposed by Albareda-Sambola et al. (2015) and described in Section 2 was solved with IBM ILOG CPLEX 12.10 solver. All algorithms use four threads, and all runs are limited to 30 wall-clock minutes.

In addition, we also compare the BRKGA results with an Iterated Local Search (`ILS`) algorithm (Lourenço, Martin, & Stützle, 2019). This is a two phase metaheuristic that initially builds a feasible solution which is given to a local search procedure aiming to improve the entry solution. Then, this locally optimum solution follows to an iterative process composed by perturbation-local search-acceptance criterion steps.

The `ILS` customized for the pNCP starts with the random generation of a feasible solution by applying the same representation of a BRKGA chromosome as well as the decoding procedure, as illustrated in Section 3.2. The incumbent solution undergoes a local search method which exchanges centers and users nodes, as explained in Section 3.3, following a best improvement strategy. Within the cycle of perturbation-local search-acceptance criterion, the perturbation step consists of a shaking procedure (Section 3.4), with the goal of escaping local minimum by randomly generating another feasible solution. The same local search procedure used on the initial solution is applied to the perturbed solution. The ILS makes use of a simulated annealing acceptance criteria so that better solutions are always accepted, while worse solutions may be accepted depending on a probability which is proportional to a temperature parameter. Initially, high temperatures ensure great probabilities of acceptance of worsening solutions. Then, as the temperature decreases, the referred probability is also reduced.

We run CPLEX with two different setups. In the first, CPLEX uses four threads and stops either when it finds an optimal integer solution, or it reaches the maximum time as defined for the heuristics (`CPLEX-30 min`). This setup is meant to achieve direct comparison with the other methods proposed in this paper. In the second setup, we run CPLEX for one week long, using 24 threads, and we set the parameter "MIP emphasis" to emphasize optimality over feasibility (`CPLEX-1w`). Such configuration looks to find optimal solutions or, at least, compute the best possible lower bounds. Since this configuration uses far more time and computer power per run than the other configurations, we use the results only for reporting.

We named the BRKGA variants as follows: `BRKGA-NLS` for the variant without local search (pure BRKGA evolution); `BRKGA-FI` for the variant with first-improvement local search; and `BRKGA-BI` for the variant with best-improvement local search. We performed 30 independent runs of each BRKGA variation for each instance.

We tuned the parameters for the BRKGA and `ILS` algorithms employing the iterated F-race and the `irace` package (López-Ibáñez, Dubois-Lacoste, Cáceres, Birattari, & Stützle, 2016). Table 2 presents the tuned parameters for BRKGA variants. The first column describes the BRKGA variation. Column $\lvert\mathscr{P}\rvert$ shows the population's size, followed by the percentage of elite individuals ($\mathscr{P}_e$%) and the percentage of mutants ($\mathscr{P}_m$%). Column "π_e, π_t" shows the number of elite individuals and the total number of individuals for mating, respectively. Column Φ depicts the bias function for mating. In all cases, `irace` recommended one population only (not shown in the table). The following three columns bring the IPR parameters: *md* is the minimum distance between two chromosomes for mating; *ps* indicates the percentage of the path to be explored; and I_{ipr} indicates the number of iterations without improvement in the best solution for calling the path-relinking. Note that `irace` suggested selecting two random individuals from the elite set for path-relinking (not shown in the table). We also fixed the number of tested pairs to one. The maximum IPR time is given by the remaining time when it is called. The next three columns show the shake and reset parameters: I_s indicates the interval without a change in the population's best solution (population stall), followed by the best solution stall multiplier (S_m) and reset multiplier (R_m). Recall that S_m and R_m give us the number of iterations based on the number of iterations given by I_s. Lastly, Column "LS%" indicates the percentage of neighbors the local search must explore.

For the `ILS` perturbation strength, `irace` suggested the value of 5. For the `ILS` simulating annealing acceptance criteria, `irace` suggested the initial temperature 43,000 with geometric decay of factor 2. For details in the parameters and configurations, refer to A.

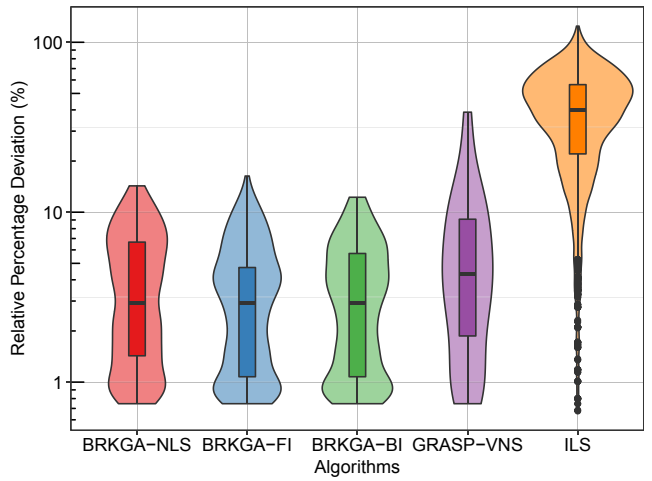


Fig. 2. Distribution of relative percentage deviations for each algorithm considering instances of Group 1. Note that, since the data is log-transformed before plot, the shown statistics (median, quartiles, and others) are from the transformed data rather than the actual data. Also, note that, for the same reason, zero deviations are not shown, although the algorithms have reached them.

4.2. Instances

We have used 416 instances split among two groups of instances, both derived from classic facility location instances from OR-Library (Beasley, 1990). The first group was proposed by Albareda-Sambola et al. (2015) generating it from the *p*-median instances `pmed1` to `pmed8` (except `pmed5`). This group contains 132 instances whose number of nodes varies from 10 to 200, and number of centers varies from 5 to 100. We named this group as "Group 1".

As we show in Section 4.3, the instances from Group 1 are easily solved by `CPLEX-30 min`. Therefore, we proposed 281 new instances using large facility location instances (`pmed5`, and `pmed9` to `pmed40`) using the same building strategy used by Albareda-Sambola et al. (2015). These instances, called "Group 2", contains from 10 to 500 nodes and from 5 to 170 centers. The reader can find details about Group 1 and Group 2 instances in B.

4.3. Instance optimality

Our first task resided in finding optimal solutions for the instances. Such optimal solutions are used as the baseline to compare the performance of the algorithms later. In the first step, we applied `CPLEX-30 min` for 30 min to all instances. In the second step, we apply `CPLEX-1w` for all instances for which an optimum solution was not found in 30 min. However, to improve `CPLEX-1w` capabilities, we inject the best solution found by the other algorithms, as a warmstart solution to `CPLEX-1w`. Such a strategy often helps solvers to improve bounds and prove the optimality via internal heuristics. We opted to provide the tables mentioned in this section in B since they are detailed and lengthy. For Group 1, we have found an optimal solution for all instances but `pmed6_200_30`. For this instance, the best found upper and lower bounds, before memory exhaustion, are 69 and 60, respectively, which give us a gap of 13.04%. `CPLEX-30 min` found optimal solutions in less than one minute for 76 instances (57%), and `CPLEX-1w` found optimal solutions in less than one hour for 48 of them (36%). The other seven instances revealed themselves much more difficult to solve, varying times from one hour to 21 h. Particularly, `CPLEX-1w` spent 7.76 days to find the optimal solution for `pmed6_150_20` (we have rerun `CPLEX-1w` without time limit for this case). The reader may refer for the full results of Table 10.

Table 3
Algorithm performance on instances of Group 1. Note that although we have no optimal solution for instance `pmed6_200_30`, we account it in this table.

Algorithm	Optima			Prop. diff.	
	# Opt	% Opt	% Run	%	σ
`BRKGA-NLS`	126	95.45	88.96	3.14	2.49
`BRKGA-FI`	126	95.45	88.31	2.66	2.04
`BRKGA-BI`	127	96.21	88.11	2.58	1.89
`GRASP-VNS`	96	72.73	74.42	5.84	6.76
`ILS`	27	20.45	6.77	40.43	22.52

For Group 2, `CPLEX-30 min` found optimal solutions for 217 (77%) instances in less than 30 min. For 17 instances (6%), `CPLEX-1w` found an optimal solution, with times varying from 30 min to seven days, with an average of 1.88 ± 2.38 days. Instances for which an optimal solution was found are described on Table 11. For 11 instances (4%), we could not prove the optimality, but we produce upper and lower bounds. For these instances which are described on Table 12, the average gap between the best solution and the lower bound is $10.55 \pm 7.86\%$, with a minimum of 4.45% and a maximum of 30.42%. For the remaining 37 instances (13%), either `CPLEX-30 min` or `CPLEX-1w` had issues with memory exhaustion and a lower bound could not be found. The detailed results are shown in Table 13.

4.4. Result analysis

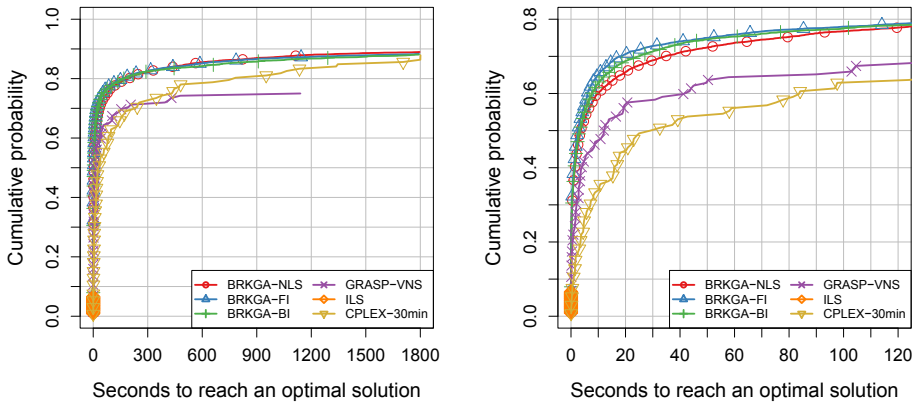
We compare our proposed algorithms with the results provided by López-Sánchez et al. (2019). In that work, the authors proposed three algorithms and applied them to Group 1 instances. Here, we only compare the results for the hybrid `GRASP-VNS` algorithm, since this version has obtained the state-of-the-art results for the p-next center problem.

To compare the algorithms, we analyze the results regarding the solution quality and computational effort. For solution quality, we compute the classical Relative Percentage Deviation (RPD) and associated averages as defined in Andrade et al. (2019), which we reproduce here. Let $\mathscr{I}$ be a set of instances. Let $\mathscr{A}$ be the set of algorithms, and assume that set $\mathscr{R}_A$ enumerates the independent runs for algorithm $A \in \mathscr{A}$ (as defined in Section 4.1, 30 runs for the heuristics and one run for `CPLEX-30 min`). We defined C_{ir}^{A} as the total cost obtained by algorithm A in instance i on run r, and C_i^{best} as the best total cost found across all algorithms for instance i. The Relative Percentage Deviation (RPD) from the best solution of instance i is defined as

$$RPD_{ir}^{A} = \frac{C_{ir}^{A} - C_i^{best}}{C_i^{best}} \times 100, \quad \forall A \in \mathscr{A}, \quad i \in \mathscr{I}, \quad \text{and} \quad r \in \mathscr{R}_A. \tag{10}$$

Since Section 4.3 revealed that Group 1 instances are substantially easier to solve than Group 2 instances, and we have no experimental information of `GRASP-VNS` for Group 2 instances, we split our analysis for each instance group.

Fig. 2 depicts a boxplot with the RDP for each algorithm for results of Group 1 instances. Note that the y-axis is plotted on a log scale to enhance the visualization. The reason is that most of the algorithms



(a) Full profile. (b) Clipping of two first minutes.

Fig. 3. Running time empirical distributions to the optimal solution values for Group 1 instances. The identification marks correspond to 2% of the points plotted for each algorithm.

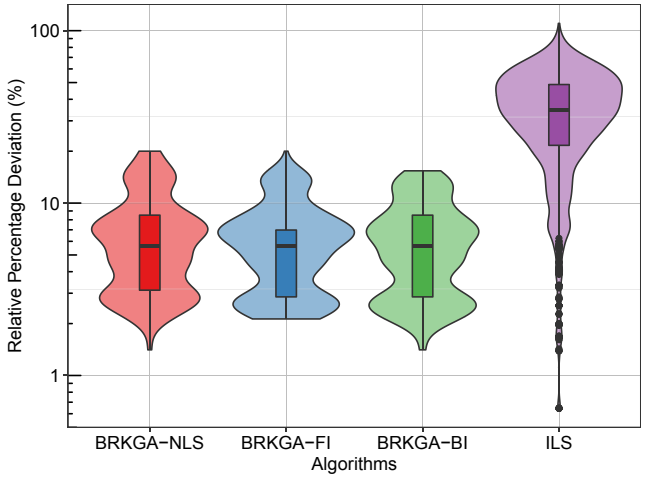


Fig. 4. Distribution of relative percentage deviations for each algorithm considering instances of Group 2. Note that, since the data is log-transformed before plot, the shown statistics (median, quartiles, and others) are from the transformed data rather than the actual data. Also, note that, for the same reason, zero deviations are not shown, although the algorithms have reached them.

found optimal solutions frequently, skewing the display on a linear scale. Indeed, the median deviation for all algorithms (except ILS) is zero indicating that, at least, half of the results found an optimal solution. On average, BRKGA variations presented a smaller deviation than GRASP-VNS. BRKGA-NLS resulted in $0.51 \pm 1.80\%$, with a maximum RDP of 14.29%; BRKGA-FI produced $0.48 \pm 1.62\%$, with maximum of 16.33%; and BRKGA-FI produced $0.47 \pm 1.60\%$, with maximum of 12.25%. GRASP-VNS results are slightly worse, with an average of $1.64 \pm 4.82\%$, with a maximum of 38.78%. ILS could not achieve good results as the other algorithms with average of $37.95 \pm 24.29\%$, with a maximum of 124.49%.

Since these results are to close to call, we applied the pairwise Wilcoxon rank-sum test with Hommel's *p*-value adjusting, among all algorithms, including CPLEX-30 min. Within a confidence interval of 95%, we cannot affirm a significant difference among the BRKGA variations.

Table 4
p-values from pairwise Wilcoxon rank-sum test (with Bonferroni *p*-value correction) of RPDs for Group 2 instances.

	BRKGA-BI	BRKGA-FI	BRKGA-NLS	CPLEX-30 min
BRKGA-FI	0.9311	—	—	—
BRKGA-NLS	0.0032	0.5449	—	—
CPLEX-30 min	0.0912	0.0583	0.0343	—
ILS	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$

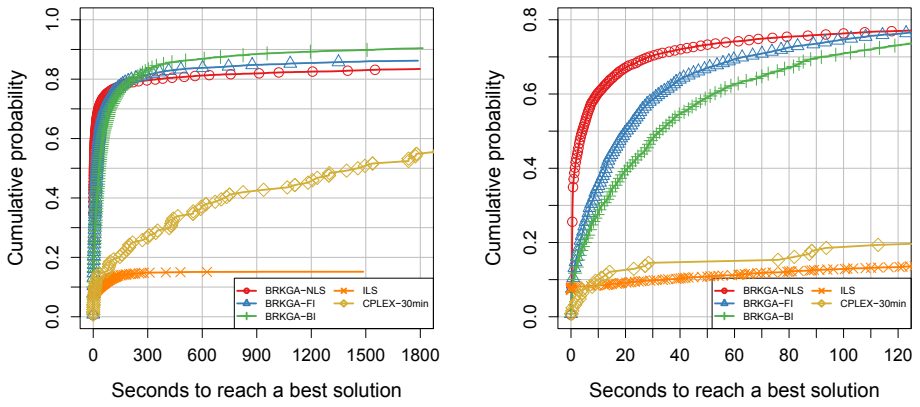
However, all BRKGA variations presented significantly better results than GRASP-VNS and ILS. Nevertheless, CPLEX-30 min presented significantly better results than all heuristics. These results were expected, though, since CPLEX-30 min found an optimal solution for all instances (but one).

Table 3 summarizes the results for each algorithm with relation to the optimal solutions for Group 1. The first column describes the algorithm. Column "# Opt" describes the number of instances for which the algorithm found an optimal solution following the absolute percentage in Column "% Opt." Column "% Run" represents the percentage of runs in which an optimal solution is found. The last two columns deal with instances from which the algorithms did not find an optimal solution. In this case, we depict the average proportional difference (%), and its standard deviation (σ), of the best-found solution to the optimal solution for each instance. Since CPLEX-30 min obtained optimal solutions for all instances but pmed6_200_30, we did not include its results in the table. Note that BRKGA variations have systematically obtained optimal solutions (around 95% of the instances), in almost all independent runs (approximately 88%). These results represent an improvement in the previous state-of-the-art GRASP-VNS, which reached about 72% of optimal solutions. BRKGA variants also outperformed GRASP-VNS on instances where they did not find an optimal solution, presenting a smaller proportional difference (and deviations) compared to GRASP-VNS. Although presented poor performance, ILS still found an optimal solution for a considerable number of instances.

Fig. 3 shows performance profiles for all algorithms for instances of Group 1. The X-axis shows the time needed to reach a target solution value while the Y-axis shows the empirical cumulative probability to reach a target solution value for the given time in the X-axis. We used all optimal solution values as target values. First, since GRASP-VNS

Table 5
Algorithm performance on instances of Group 2.

Algorithm	Known Optima (233 instances)					Unknown Optima (48 instances)				
	Optima			Prop. diff.		Best			Prop. diff.	
	# Opt	% Opt	% Run	%	σ	# Best	% Best	% Run	%	σ
BRKGA-NLS	219	93.99	85.01	5.25	2.70	47	97.92	84.31	4.86	2.02
BRKGA-FI	221	94.85	87.44	4.91	2.19	44	91.67	80.28	6.59	3.63
BRKGA-BI	232	99.57	91.69	4.13	1.50	41	85.42	75.76	7.61	3.52
CPLEX-30 min	84	36.05	36.05	$\gg 10^4$	$\gg 10^4$	0	0.00	0.00	$\gg 10^4$	—
ILS	69	29.61	11.03	33.77	18.24	25	52.08	35.21	30.25	20.71



(a) Full profile. (b) Clipping of two first minutes.

Fig. 5. Running time empirical distributions to the best solution values for Group 2 instances. The identification marks correspond to 2% of the points plotted for each algorithm.

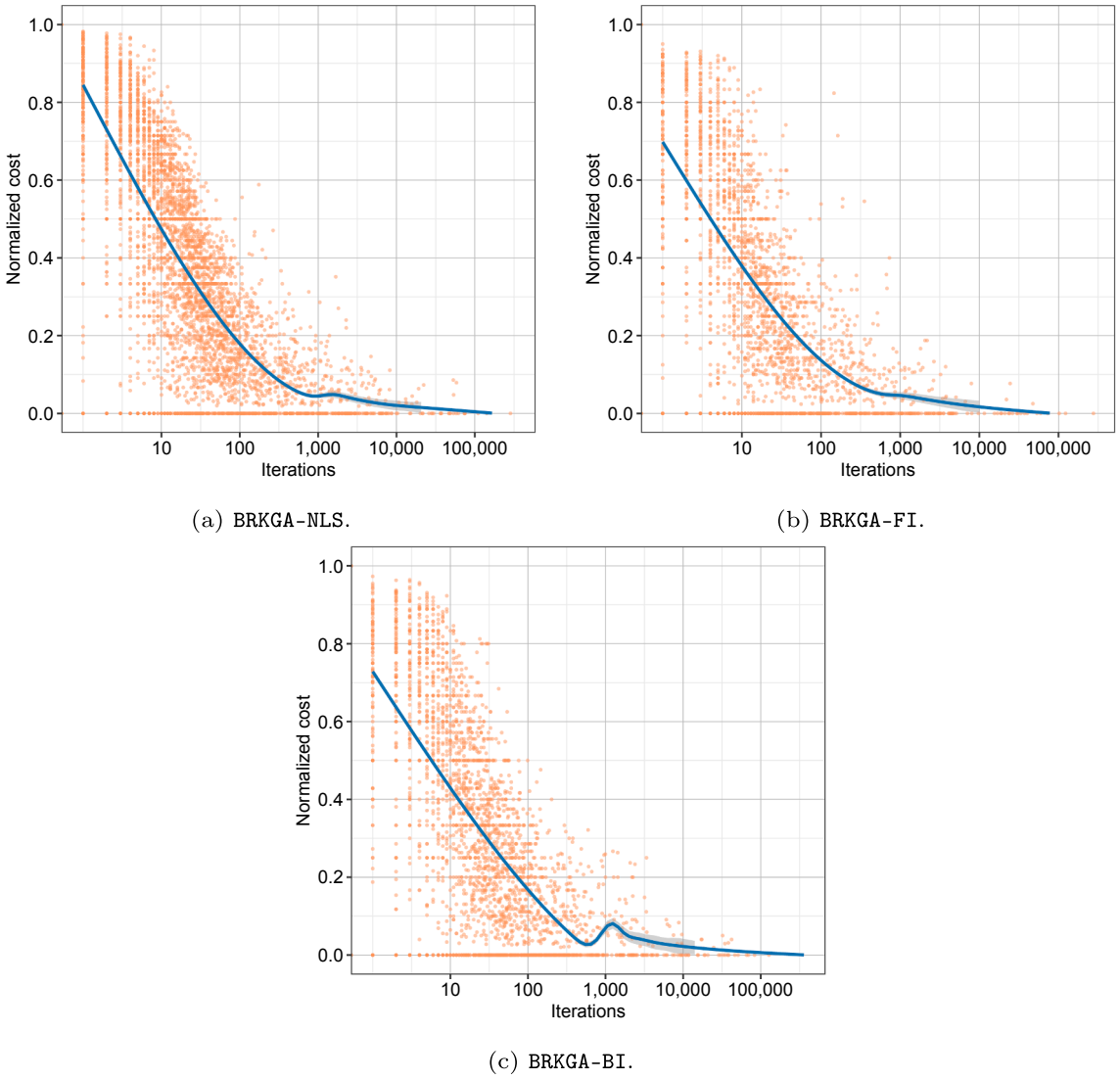
experiments were conducted in a different setup, straight time comparison can be misled for differences of the experimental test-beds. Therefore, we refrain from a direct comparison between GRASP-VNS and BRKGA variants in this matter. Note that GRASP-VNS has a reasonable probability of finding an optimal solution in the first two minutes (around 70%), and indeed, it presents a better probability than CPLEX-30 min in the first 3 min. BRKGA variants have similar behavior again, although BRKGA-FI presents slightly better probability in the first minute. ILS could not achieve good probabilities, getting good solutions only in the first few seconds.

Since the instances of Group 1 can be easily solved by CPLEX-30 min, and we could not detect a substantial difference among our proposed strategies using such instances, we turn to analyze the results for the harder instances of Group 2. Note that we have no results of GRASP-VNS for such instances, and therefore, we do not discuss this algorithm here.

Fig. 4 shows the boxplot for the RDP of Group 2 results for each algorithm. Again, note that the y-axis is depicted on a log scale to enhance the visualization. Note that the deviations are very similar among the algorithms. BRKGA-NLS have an average RDP of $0.44 \pm 2.00\%$ with a maximum of 20%, while BRKGA-FI presents an average deviation of $0.36 \pm 1.67\%$ with a maximum of 20%, and BRKGA-BI presents $0.32 \pm 1.58\%$ with a maximum of 15.38%. ILS delivered worse results with average deviation of $14.71 \pm 21.37\%$ with a maximum of 110.26%. However, all algorithm, including ILS, have their median of 0.00, indicating that at least half of the results achieve the best solution.

To confirm such differences, we once again apply the pairwise Wilcoxon rank-sum test among the algorithms, including CPLEX-30 min. Table 4 shows the p-value matrix for these tests, in which a value of less than 0.05 indicates that the column algorithm has significantly better deviations than the row algorithm. For an interval of confidence of 95%, BRKGA-BI is significantly better than BRKGA-NLS, but we cannot affirm the same when compared to either BRKGA-FI or CPLEX-30 min. CPLEX-30 min presented significantly better results than BRKGA-NLS and BRKGA-FI, but not when compared to BRKGA-BI. ILS presented significantly worse results than all other algorithms.

Table 5 summarizes the results for each algorithm with relation to the optimal solutions for Group 2. The first section of the table depicts results compared to optimal solutions, as described in Table 3. The second section of the table shows the results for instances with unknown optimum solutions. The description is similar to the first section. However, the results are compared to the best solution found by the algorithms. Note that BRKGA-BI found an optimal solution for all optimal instances but pmed23_500_130, in more than 90% of its runs. This is an extraordinary result compared to the other variants and CPLEX-30 min. It is interesting to note that, for instances with unknown optima, BRKGA-BI obtained the worst results among BRKGA variants. The reason is that such instances are enormous, and the full-search mechanism of the best improvement strategy wastes time polishing the solution. Indeed, for these cases, BRKGA-NLS presented the best results since



(a) BRKGA-NLS. (b) BRKGA-FI.

(c) BRKGA-BI.

Fig. 6. BRKGA variants evolution profiles, given the number of iterations and the normalized cost for each instance.

it does not spend time on local search and allows more time for evolution and convergence. `CPLEX-30 min` had issues finding good solutions for instances of Group 2, and in some cases, it presented huge differences in cost. For the instances with known optima, `ILS` performed worse than all other algorithms, although it found optimal solutions for about 30% of the instances. For the large instances with unknown optima, `ILS` performed better than `CPLEX-30 min` and found the best solutions for about half of the instances. These results are interesting and show that `ILS` is still usable in some cases.

Fig. 5 shows performance profiles for all algorithms for instances of Group 2. The axis descriptions follow Fig. 3. However, we use both optimal solution values as much as the best-known solution values as targets. In the long run, `BRKGA-BI` outpaces the other variants reaching a 90% probability of finding an optimal or a best-known solution. Such behavior starts around three minutes. Before that, `BRKGA-NLS` performs better than other variants reaching a cumulative probability of 70% in just 50 s. Again, we attribute this behavior to the lack of local search of `BRKGA-NLS`, allowing faster convergence of the population. Contrasting the results for Group 1, `ILS` obtained a cumulative probability similar to `CPLEX-30 min` in the first two minutes for Group 2. After that, `CPLEX-30 min` outpaces `ILS`.

Fig. 6 depicts the evolutionary profile of BRKGA variants. We normalize each instance's cost and each run, making the initial solution proportional to 1.0 and the best solution proportional to 0.0 on the Y-axis. The X-axis shows the number of iterations on the log scale. One can note that both variants converge around 1,000 iterations, with a small blip after that. Such behavior is due to the large and hard instances, which require far more iterations to obtain reasonable solutions. Indeed, more than 64 instances took more than 1,000 iterations to converge on average for both variants. Considering an average of 10,000 iterations, we have 24 instances. Moreover, instance `pmed1_50_10` uses more than 100,000 iterations on `BRKGA-NLS` and `BRKGA-FI` on average.

Therefore, `BRKGA-BI` presents both excellent quality solutions as finding them more frequently than the other variants, and it should be chosen by a practitioner to design and deploy its relief centers. However, if time is crucial, and the engineer cannot wait to make a decision and deploy the relief units, `BRKGA-NLS` may serve as an excellent and fast algorithm to propose the deployment.

5. Conclusions

In this work, we investigated an evolutionary algorithm to solve the p-next center problem (pNCP). pNCP is a location problem, derived from the classical p-center problem, that considers the possibility of operation

disruption in the facility installed in a reference center. In this way, a backup center must be assigned to the reference one.

The evolutionary approach implements a Biased Random-Key Genetic Algorithm (BRKGA), which incorporates a multi-parent (MP) strategy to generate offspring and includes a path-relinking (PR) as an intensification search procedure. We presented three variations of this BRKGA-MP-IPR and compared them with results from a commercial solver using an integer programming formulation, a hybrid GRASP-VNS heuristic described in the literature on the pNCP, and an Iterated Local Search algorithm .

Computational experiments established optimal solution values for 401 out of 416 benchmark instances and provided upper and lower bounds for the remaining ones. We also showed the superior performance of the proposed algorithms concerning the GRASP-VNS as well as the `ILS`, since they were able to find a larger number of optimal solutions as well as a smaller percentage deviations to the optimal values. Besides, all BRKGA variations showed their ability to find excellent quality solutions. Worth noting that the proper choice of the local search strategy provides a good trade-off between running time and quality, which makes the proposed methods a practical approach for different contexts.

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.

CRediT authorship contribution statement

Mariana A. Londe: Conceptualization, Methodology, Software, Validation, Investigation, Writing - original draft, Writing - review & editing. **Carlos E. Andrade:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing, Visualization, Supervision. **Luciana S. Pessoa:** Conceptualization, Methodology, Software, Validation, Investigation, Writing - original draft, Writing - review & editing, Supervision.

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.

Acknowledgement

The authors wish to thank Yolanda Hinojosa for having kindly provided the results obtained by the mathematical formulations as described by Albareda-Sambola et al. (2015). This study was financed in part by CNPq, PUC-Rio, FAPERJ (Project Numbers E-26/211.086/2019 and E-26/010.002576/2019), and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

Appendix A. Parameter settings

BRKGA-MP-IPR has a larger number of control parameters which makes difficult to perform a full factorial design. Therefore, we used the *iterated racing* (Birattari, Yuan, Balaprakash, & Stützle, 2010) to tune the parameters. This method consists in sampling configurations from a particular distribution, evaluating them using a statistical, and refining the sampling distribution with repeated applications of F-Race. We used the `irace` package (López-Ibáñez et al., 2016), implemented in R, for such task.

We used a budget of 3,000 experiments in the tuning procedure over the 30 instances selected randomly. Each experiment was limited to 10 min. To tune the BRKGA-MP-IPR parameters (the notation follows Andrade et al., 2021), we used the following ranges or sets:

- population size $|\mathscr{P}| \in [100, 5000]$;
- percentage of elite individuals $\mathscr{P}_e\% \in [0.1, 0.5]$;
- percentage of mutants introduced at each generation $\mathscr{P}_m\% \in [0.1, 0.5]$;
- number of elite individuals and the total number of individuals for mating, pairs $(\pi_e, \pi_t) \in \{(1,3),(2,3),(2,6),(3,6),(4,6),(3,10),(5,10),(7,10)\}$;
- bias function for mating $\Phi(r) \in \{1/\log(r+1), r^{-1}, r^{-2}, r^{-3}, e^{-r}, 1/\pi_t\}$;
- number of independent populations $p \in [1, 3]$;
- minimum distance between chromosomes for path-relink $md \in [0.0, 0.3]$;
- individual selection $sel \in \{\text{RE}, \text{BS}\}$, where RE indicates random elite individuals, and BS indicates best solutions from different populations;
- path percentage/size $ps\% \in [0.01, 1.00]$;
- shake interval without a change in the population's best solution (population stall) $\text{I}_s \in [20, 100]$;
- best solution stall multiplier $\text{S}_m \in [1.1, 2.0]$;
- reset multiplier $\text{R}_m \in [1.1, 2.0]$;
- percentage of neighbors the local search LS% $\in [0.1, 1.0]$.

Since we use a permutation decoder, the path relink type (*typ*) was set to "permutation", and the block size (*bs*) to zero (since *bs* does not matter for permutation). The maximum IPR time is given by the remaining time when it is called.

Tables 6–8 show the suggestions made by `irace`. We picked the values of the first line of each table, rounded up to two digits for real values, and round to the next multiple of 10, in the case of integer values.

We also used `irace` to tune `ILS` parameters using the same methodology (and worth to stress, the same instances) applied to BRKGA variants. The following ranges were used:

- perturbation strength in $PS \in [1, 10]$;
- initial temperature $T_{init} \in [1024, 65536]$;
- temperature decay $\delta \in [2, 10]$;
- local search (LS) either best improvement or first improvement.

Table 9 shows the suggestions made by `irace`.

Table 6
Best parameter configurations suggested by `irace` for `BRKGA-NLS`.

BRKGA						IPR				Shaking			
	$\lvert\mathscr{P}\rvert$	$\mathscr{P}_e$%	$\mathscr{P}_m$%	π_e, π_t	Φ	p	md	sel	ps%	I_{ipr}	I_s	S_m	R_m
3000	0.19	0.25	1,3	r^{-3}	1	0.13	RE	0.40	270	100	1.79	1.88	1.88
3810	0.24	0.23	5,10	e^{-r}	2	0.15	RE	0.27	139	91	1.48	1.69	1.69
3031	0.14	0.24	1,3	r^{-3}	3	0.27	RE	0.66	146	90	1.77	1.81	1.81
3104	0.17	0.17	1,3	r^{-3}	1	0.16	BS	0.33	291	89	1.76	1.92	1.92
2418	0.11	0.18	1,3	r^{-3}	2	0.21	RE	0.80	134	100	1.82	1.78	1.78

Table 7
Best parameter configurations suggested by `irace` for `BRKGA-FI`.

BRKGA						IPR				Shaking			LS%
$\lvert\mathscr{P}\rvert$	$\mathscr{P}_e$%	$\mathscr{P}_m$%	π_e, π_t	Φ	p	md	sel	ps%	I_{ipr}	I_s	S_m	R_m	
4500	0.19	0.12	7,10	r^{-2}	1	0.10	RE	0.79	230	84	1.60	1.19	0.57
4942	0.21	0.18	7,10	r^{-2}	1	0.12	BS	0.97	300	86	1.99	1.88	0.49
3088	0.30	0.28	7,10	r^{-2}	2	0.21	BS	0.92	199	78	1.95	1.80	0.36
4015	0.20	0.32	7,10	r^{-2}	1	0.17	BS	0.80	269	82	1.83	1.90	0.43
2614	0.26	0.12	7,10	r^{-2}	2	0.28	BS	0.83	293	64	1.82	1.44	0.60

Table 8
Best parameter configurations suggested by `irace` for `BRKGA-BI`.

BRKGA						IPR				Shaking			LS%
$\lvert\mathscr{P}\rvert$	$\mathscr{P}_e$%	$\mathscr{P}_m$%	π_e, π_t	Φ	p	md	sel	ps%	I_{ipr}	I_s	S_m	R_m	
4000	0.27	0.22	4,6	r^{-2}	1	0.20	RE	0.92	120	85	1.76	1.87	0.45
2890	0.25	0.19	4,6	r^{-2}	1	0.21	RE	0.71	87	78	1.81	1.64	0.30
1959	0.20	0.15	1,3	r^{-3}	3	0.15	RE	0.02	442	88	1.46	1.82	0.98
4227	0.25	0.34	5,10	r^{-2}	1	0.26	RE	0.17	95	56	1.85	1.21	0.66
2793	0.26	0.22	4,6	r^{-2}	1	0.12	RE	0.74	70	73	1.84	1.73	0.24

Table 9
Best parameter configurations suggested by `irace` for `ILS`. BI stands for best improvement.

PS	T_{init}	δ	LS
5	43000	2	BI
7	46612	2	BI
1	1677	2	BI
8	56078	2	BI

Appendix B. Instance details and best results

Tables 10–13.

Table 10
Description of the instances from Group 1 with the respective optima, except for `pmed6_200_30`. The first column describes the name of the instances, followed by the number of nodes and centers on columns 2 and 3. The fourth column shows the optima value. The last column shows the total time for finding an optimal solution, where "s," "m," "h," and "d" stand for seconds, minutes, hours, and days, respectively.

Instance	Nodes	Centers	Cost	Time
pmed1_10_5	10	5	84	0.04 s
pmed1_20_5	20	5	120	0.54 s
pmed1_20_10	20	10	95	0.13 s
pmed1_30_5	30	5	126	2.23 s
pmed1_30_10	30	10	95	0.93 s
pmed1_40_5	40	5	144	8.02 s
pmed1_40_10	40	10	111	3.49 s
pmed1_40_20	40	20	89	1.30 s
pmed1_50_10	50	10	110	12.47 s
pmed1_50_20	50	20	89	2.79 s
pmed1_60_10	60	10	112	1.24 m
pmed1_60_20	60	20	89	5.35 s
pmed1_60_30	60	30	89	4.79 s
pmed1_70_10	70	10	119	6.49 m
pmed1_70_20	70	20	99	18.48 s
pmed1_70_30	70	30	73	7.39 s
pmed1_80_10	80	10	129	39.19 m
pmed1_80_20	80	20	102	1.65 m
pmed1_80_30	80	30	85	45.37 s
pmed1_90_10	90	10	133	41.52 m
pmed1_90_20	90	20	107	15.80 m
pmed1_90_30	90	30	87	38.43 s
pmed1_90_50	90	50	70	15.74 s
pmed1_100_10	100	10	133	1.85 h
pmed1_100_20	100	20	108	1.53 h
pmed1_100_30	100	30	93	3.80 m
pmed1_100_50	100	50	70	22.85 s
pmed2_10_5	10	5	121	0.19 s
pmed2_20_5	20	5	147	0.53 s
pmed2_20_10	20	10	99	0.38 s
pmed2_30_5	30	5	169	2.07 s
pmed2_30_10	30	10	110	1.01 s
pmed2_40_5	40	5	164	4.18 s
pmed2_40_10	40	10	112	4.23 s
pmed2_40_20	40	20	96	1.14 s
pmed2_50_10	50	10	140	12.29 s
pmed2_50_20	50	20	99	2.55 s
pmed2_60_10	60	10	140	1.43 m
pmed2_60_20	60	20	99	8.95 s
pmed2_60_30	60	30	96	4.85 s
pmed2_70_10	70	10	138	2.03 m
pmed2_70_20	70	20	102	19.38 s
pmed2_70_30	70	30	96	8.43 s
pmed2_80_10	80	10	138	40.10 m
pmed2_80_20	80	20	109	57.72 s
pmed2_80_30	80	30	97	25.40 s
pmed2_90_10	90	10	138	35.10 m
pmed2_90_20	90	20	108	1.91 m
pmed2_90_30	90	30	96	16.44 s
pmed2_90_50	90	50	96	23.06 s
pmed2_100_10	100	10	135	47.24 m
pmed2_100_20	100	20	107	5.62 m
pmed2_100_30	100	30	96	26.75 s
pmed2_100_50	100	50	96	25.59 s
pmed3_10_5	10	5	77	0.10 s
pmed3_20_5	20	5	145	0.47 s
pmed3_20_10	20	10	77	0.20 s
pmed3_30_5	30	5	157	2.60 s
pmed3_30_10	30	10	122	1.31 s
pmed3_40_5	40	5	157	7.03 s
pmed3_40_10	40	10	105	3.72 s
pmed3_40_20	40	20	77	1.68 s
pmed3_50_10	50	10	125	10.62 s
pmed3_50_20	50	20	87	2.68 s
pmed3_60_10	60	10	124	59.36 s
pmed3_60_20	60	20	97	25.37 s
pmed3_60_30	60	30	73	5.60 s
pmed3_70_10	70	10	121	5.75 m
pmed3_70_20	70	20	97	33.94 s
pmed3_70_30	70	30	82	14.16 s
pmed3_80_10	80	10	121	8.72 m
pmed3_80_20	80	20	93	2.04 m
pmed3_80_30	80	30	84	21.56 s
pmed3_90_10	90	10	148	13.03 m
pmed3_90_20	90	20	105	2.79 m
pmed3_90_30	90	30	93	21.80 s
pmed3_90_50	90	50	93	13.19 s
pmed3_100_10	100	10	151	58.59 m
pmed3_100_20	100	20	109	11.98 m
pmed3_100_30	100	30	93	33.98 s
pmed3_100_50	100	50	93	20.84 s
pmed4_10_5	10	5	126	0.08 s
pmed4_20_5	20	5	139	0.33 s
pmed4_20_10	20	10	125	0.26 s
pmed4_30_5	30	5	173	1.66 s
pmed4_30_10	30	10	122	1.28 s
pmed4_40_5	40	5	175	6.31 s
pmed4_40_10	40	10	122	4.03 s
pmed4_40_20	40	20	85	1.44 s
pmed4_50_10	50	10	126	12.16 s
pmed4_50_20	50	20	91	3.23 s
pmed4_60_10	60	10	134	1.22 m
pmed4_60_20	60	20	93	21.26 s
pmed4_60_30	60	30	79	4.01 s
pmed4_70_10	70	10	146	5.28 m
pmed4_70_20	70	20	102	18.65 s
pmed4_70_30	70	30	85	9.69 s
pmed4_80_10	80	10	146	22.09 m
pmed4_80_20	80	20	114	2.91 m
pmed4_80_30	80	30	90	18.98 s
pmed4_90_10	90	10	147	55.71 m
pmed4_90_20	90	20	112	2.34 m
pmed4_90_30	90	30	92	25.83 s
pmed4_90_50	90	50	82	15.71 s
pmed4_100_10	100	10	147	1.13 h
pmed4_100_20	100	20	118	44.30 m
pmed4_100_30	100	30	96	1.27 m
pmed4_100_50	100	50	82	25.63 s
pmed6_150_20	150	20	77	7.78 d
pmed6_150_30	150	30	64	5.17 h
pmed6_150_50	150	50	56	2.42 m
pmed6_150_80	150	80	56	1.30 m
pmed6_200_30	200	30	69	6.05 d
pmed6_200_50	200	50	49	48.73 m
pmed6_200_80	200	80	49	5.99 m
pmed6_200_100	200	100	49	2.93 m
pmed7_150_20	150	20	67	10.13 h
pmed7_150_30	150	30	59	3.92 m
pmed7_150_50	150	50	59	1.69 m
pmed7_150_80	150	80	59	1.78 m
pmed7_200_30	200	30	60	22.00 h
pmed7_200_50	200	50	50	55.14 m
pmed7_200_80	200	80	46	5.70 m
pmed7_200_100	200	100	46	3.78 m
pmed8_150_20	150	20	70	11.21 h
pmed8_150_30	150	30	58	4.38 m
pmed8_150_50	150	50	58	1.61 m
pmed8_150_80	150	80	58	1.52 m
pmed8_200_30	200	30	69	1.39 h
pmed8_200_50	200	50	68	5.82 m
pmed8_200_80	200	80	68	2.43 m
pmed8_200_100	200	100	68	4.64 m

Table 11
Description of the instances from Group 2 with the respective optima. Column 1 describes the name of the instances, followed by the number of nodes and centers on columns 2 and 3. Column 5 shows the best value found. Column 6 shows the total time for finding an optimal solution, where "s," "m," "h," and "d" stand for seconds, minutes, hours, and days, respectively.

Instance	Nodes	Centers	Cost	Time
pmed5_10_5	10	5	125	0.00 s
pmed5_20_5	20	5	139	0.00 s
pmed5_20_10	20	10	91	0.00 s
pmed5_30_5	30	5	155	0.00 s
pmed5_30_10	30	10	120	0.00 s
pmed5_40_5	40	5	164	0.00 s
pmed5_40_10	40	10	127	1.00 s
pmed5_40_20	40	20	91	1.00 s
pmed5_50_10	50	10	121	1.00 s
pmed5_50_20	50	20	89	1.00 s
pmed5_60_10	60	10	119	1.00 s
pmed5_60_20	60	20	90	1.00 s
pmed5_60_30	60	30	82	1.00 s
pmed5_70_10	70	10	119	1.00 s
pmed5_70_20	70	20	86	1.00 s
pmed5_70_30	70	30	85	1.00 s
pmed5_80_10	80	10	118	1.00 s
pmed5_80_20	80	20	90	1.00 s
pmed5_80_30	80	30	85	1.00 s
pmed5_90_10	90	10	119	1.00 s
pmed5_90_20	90	20	92	1.00 s
pmed5_90_30	90	30	85	1.00 s
pmed5_100_10	100	10	119	1.00 s
pmed5_100_20	100	20	94	3.00 s
pmed5_100_30	100	30	85	1.00 s
pmed9_150_20	150	20	71	1.78 m
pmed9_150_30	150	30	71	1.00 s
pmed9_150_50	150	50	71	1.00 s
pmed9_150_80	150	80	71	1.00 s
pmed9_200_30	200	30	71	1.13 m
pmed9_200_50	200	50	71	1.00 s
pmed9_200_80	200	80	71	1.00 s
pmed9_200_100	200	100	71	1.00 s
pmed10_150_20	150	20	62	1.00 s
pmed10_150_30	150	30	62	1.00 s
pmed10_150_50	150	50	62	1.00 s
pmed10_150_80	150	80	62	1.00 s
pmed10_200_30	200	30	70	1.00 s
pmed10_200_50	200	50	70	1.00 s
pmed10_200_80	200	80	70	1.00 s
pmed10_200_100	200	100	70	1.00 s
pmed11_250_30	250	30	44	6.40 d
pmed11_250_50	250	50	42	14.00 s
pmed11_250_70	250	70	42	6.00 s
pmed11_250_90	250	90	42	3.00 s
pmed11_300_60	300	60	51	4.00 s
pmed11_300_80	300	80	51	3.00 s
pmed11_300_100	300	100	51	1.00 s
pmed11_300_150	300	150	51	1.00 s
pmed12_250_30	250	30	47	2.55 d
pmed12_250_50	250	50	43	21.08 m
pmed12_250_70	250	70	43	14.00 s
pmed12_250_90	250	90	43	5.00 s
pmed12_300_60	300	60	72	1.00 s
pmed12_300_80	300	80	72	1.00 s
pmed12_300_100	300	100	72	1.00 s
pmed12_300_150	300	150	72	1.00 s
pmed13_250_30	250	30	47	1.58 h
pmed13_250_50	250	50	44	3.10 m
pmed13_250_70	250	70	44	11.00 s
pmed13_250_90	250	90	44	5.00 s
pmed13_300_60	300	60	39	7.20 h
pmed13_300_80	300	80	39	6.04 h
pmed13_300_100	300	100	39	46.00 s
pmed13_300_150	300	150	39	6.00 s
pmed14_250_30	250	30	60	6.00 s
pmed14_250_50	250	50	60	2.00 s
pmed14_250_70	250	70	60	1.00 s
pmed14_250_90	250	90	60	1.00 s
pmed14_300_60	300	60	60	3.00 s
pmed14_300_80	300	80	60	1.00 s
pmed14_300_100	300	100	60	1.00 s
pmed14_300_150	300	150	60	1.00 s
pmed15_250_30	250	30	49	16.00 s
pmed15_250_50	250	50	49	4.00 s
pmed15_250_70	250	70	49	2.00 s
pmed15_250_90	250	90	49	2.00 s
pmed15_300_60	300	60	44	14.00 s
pmed15_300_80	300	80	44	9.00 s
pmed15_300_100	300	100	44	4.00 s
pmed15_300_150	300	150	44	1.00 s
pmed16_350_70	350	70	33	32.00 s
pmed16_350_90	350	90	33	18.00 s
pmed16_350_120	350	120	33	8.00 s
pmed16_400_80	400	80	33	50.00 s
pmed16_400_100	400	100	33	34.00 s
pmed16_400_140	400	140	33	10.00 s
pmed16_400_200	400	200	33	2.00 s
pmed17_350_40	350	40	37	29.00 s
pmed17_350_70	350	70	37	9.00 s
pmed17_350_90	350	90	37	7.00 s
pmed17_350_120	350	120	37	4.00 s
pmed17_400_80	400	80	37	12.00 s
pmed17_400_100	400	100	37	8.00 s
pmed17_400_140	400	140	37	1.00 s
pmed17_400_200	400	200	37	1.00 s
pmed18_350_40	350	40	50	3.00 s
pmed18_350_70	350	70	50	1.00 s
pmed18_350_90	350	90	50	1.00 s
pmed18_350_120	350	120	50	1.00 s
pmed18_400_80	400	80	50	1.00 s
pmed18_400_100	400	100	50	1.00 s
pmed18_400_140	400	140	50	1.00 s
pmed18_400_200	400	200	50	1.00 s
pmed19_350_40	350	40	35	1.83 d
pmed19_350_70	350	70	35	57.00 s
pmed19_350_90	350	90	35	27.00 s
pmed19_350_120	350	120	35	10.00 s
pmed19_400_80	400	80	32	1.30 m
pmed19_400_100	400	100	32	52.00 s
pmed19_400_140	400	140	32	10.00 s
pmed19_400_200	400	200	32	3.00 s
pmed20_350_40	350	40	40	4.40 m
pmed20_350_70	350	70	40	15.00 s
pmed20_350_90	350	90	40	7.00 s
pmed20_350_120	350	120	40	4.00 s
pmed20_400_80	400	80	40	16.00 s
pmed20_400_100	400	100	40	11.00 s
pmed20_400_140	400	140	40	3.00 s
pmed20_400_200	400	200	40	1.00 s
pmed21_450_90	450	90	25	19.61 h
pmed21_450_120	450	120	25	2.15 h
pmed21_450_150	450	150	25	1.88 m
pmed21_500_100	500	100	28	1.40 m
pmed21_500_130	500	130	28	41.00 s
pmed21_500_170	500	170	28	17.00 s
pmed21_500_250	500	250	28	2.00 s
pmed22_450_50	450	50	36	22.00 s
pmed22_450_90	450	90	36	9.00 s
pmed22_450_120	450	120	36	3.00 s
pmed22_450_150	450	150	36	1.00 s
pmed22_500_100	500	100	44	1.00 s
pmed22_500_130	500	130	44	1.00 s
pmed22_500_170	500	170	44	1.00 s
pmed22_500_250	500	250	44	1.00 s
pmed23_450_90	450	90	29	4.18 h
pmed23_450_120	450	120	29	30.08 m
pmed23_450_150	450	150	29	52.00 s
pmed23_500_100	500	100	29	4.70 h
pmed23_500_130	500	130	29	2.44 d
pmed23_500_170	500	170	29	32.00 s
pmed23_500_250	500	250	29	5.00 s
pmed24_450_50	450	50	33	30.00 s
pmed24_450_90	450	90	33	11.00 s
pmed24_450_120	450	120	33	5.00 s
pmed24_450_150	450	150	33	2.00 s

(*continued on next page*)

Table 11 (*continued*)

Instance	Nodes	Centers	Cost	Time
pmed24_500_100	500	100	33	15.00 s
pmed24_500_130	500	130	33	7.00 s
pmed24_500_170	500	170	33	1.00 s
pmed24_500_250	500	250	33	1.00 s
pmed25_450_50	450	50	44	2.00 s
pmed25_450_90	450	90	44	1.00 s
pmed25_450_120	450	120	44	1.00 s
pmed25_450_150	450	150	44	1.00 s
pmed25_500_100	500	100	44	1.00 s
pmed25_500_130	500	130	44	1.00 s
pmed25_500_170	500	170	44	1.00 s
pmed25_500_250	500	250	44	1.00 s
pmed26_550_60	550	60	35	11.00 s
pmed26_550_110	550	110	35	3.00 s
pmed26_550_150	550	150	35	1.00 s
pmed26_550_190	550	190	35	1.00 s
pmed26_600_120	600	120	32	8.00 s
pmed26_600_150	600	150	32	5.00 s
pmed26_600_200	600	200	32	1.00 s
pmed26_600_300	600	300	32	1.00 s
pmed27_550_60	550	60	27	7.00 d
pmed27_550_110	550	110	27	56.00 s
pmed27_550_150	550	150	27	19.00 s
pmed27_550_190	550	190	27	3.00 s
pmed27_600_120	600	120	33	4.00 s
pmed27_600_150	600	150	33	1.00 s
pmed27_600_200	600	200	33	1.00 s
pmed27_600_300	600	300	33	1.00 s
pmed28_550_60	550	60	27	1.60 m
pmed28_550_110	550	110	27	22.00 s
pmed28_550_150	550	150	27	10.00 s
pmed28_550_190	550	190	27	4.00 s
pmed28_600_120	600	120	57	1.00 s
pmed28_600_150	600	150	57	1.00 s
pmed28_600_200	600	200	57	1.00 s
pmed28_600_300	600	300	57	1.00 s
pmed29_550_110	550	110	23	2.54 d
pmed29_550_150	550	150	23	2.52 m
pmed29_550_190	550	190	23	59.00 s
pmed29_600_120	600	120	36	1.00 s
pmed29_600_150	600	150	36	1.00 s
pmed29_600_200	600	200	36	1.00 s
pmed29_600_300	600	300	36	1.00 s
pmed30_550_60	550	60	43	1.00 s
pmed30_550_110	550	110	43	1.00 s
pmed30_550_150	550	150	43	1.00 s
pmed30_550_190	550	190	43	1.00 s
pmed30_600_120	600	120	40	1.00 s
pmed30_600_150	600	150	40	1.00 s
pmed30_600_200	600	200	40	1.00 s
pmed30_600_300	600	300	40	1.00 s
pmed31_650_130	650	130	22	1.47 m
pmed31_650_170	650	170	22	1.27 m
pmed31_650_220	650	220	22	25.00 s
pmed31_700_180	700	180	21	1.93 h
pmed31_700_240	700	240	21	39.00 s
pmed31_700_350	700	350	21	5.00 s
pmed32_650_20	650	220	22	2.55 m
pmed32_700_140	700	140	72	1.00 s
pmed32_700_180	700	180	72	1.00 s
pmed32_700_240	700	240	72	1.00 s
pmed32_700_350	700	350	72	1.00 s
pmed33_650_70	650	70	25	1.70 m
pmed33_650_130	650	130	25	37.00 s
pmed33_650_170	650	170	25	16.00 s
pmed33_650_220	650	220	25	4.00 s
pmed33_700_180	700	180	22	3.02 m
pmed33_700_240	700	240	22	18.00 s
pmed33_700_350	700	350	22	2.00 s
pmed34_650_70	650	70	26	5.34 d
pmed34_650_130	650	130	26	26.00 s
pmed34_650_170	650	170	26	14.00 s
pmed34_650_220	650	220	26	5.00 s
pmed34_700_140	700	140	41	1.00 s
pmed34_700_180	700	180	41	1.00 s
pmed34_700_240	700	240	41	1.00 s

Table 11 (*continued*)

Instance	Nodes	Centers	Cost	Time
pmed34_700_350	700	350	41	1.00 s
pmed35_750_150	750	150	20	7.00 d
pmed35_750_190	750	190	20	5.08 m
pmed35_750_250	750	250	20	22.00 s
pmed36_750_80	750	80	24	5.17 m
pmed36_750_150	750	150	24	57.00 s
pmed36_750_190	750	190	24	26.00 s
pmed36_750_250	750	250	24	6.00 s
pmed37_750_80	750	80	33	3.00 s
pmed37_750_150	750	150	33	1.00 s
pmed37_750_190	750	190	33	1.00 s
pmed37_750_250	750	250	33	1.00 s

Table 12
Description of the instances from Group 2 with the respective best solution values and lower bounds within seven days lasting experiments. Columns 1–6 follow the same description of those of Table 11. Column "LB" gives us the best lower bound found for that instance followed by the "Gap" the best solution value and the lower bound, given in percentage.

Instance	Nodes	Centers	Cost	LB	Gap%
pmed16_350_40	350	40	35	33	6.06
pmed21_450_50	450	50	29	26	11.54
pmed23_450_50	450	50	34	29	17.24
pmed29_550_60	550	60	30	23	30.43
pmed31_650_70	650	70	24	22	9.09
pmed31_700_140	700	140	22	21	4.76
pmed32_650_70	650	70	25	22	13.63
pmed32_650_130	650	130	23	22	4.45
pmed32_650_170	650	170	23	22	4.45
pmed33_700_140	700	140	23	22	4.45
pmed35_750_80	750	80	22	20	10.00

Table 13
Description of the instances from Group 2 for which no lower was found due to lack of memory issues. The columns follow the same description of those of Table 11.

Instance	Nodes	Centers	Cost
pmed35_800_160	800	160	29
pmed35_800_200	800	200	29
pmed35_800_270	800	270	29
pmed35_800_400	800	400	29
pmed36_800_160	800	160	42
pmed36_800_200	800	200	42
pmed36_800_270	800	270	42
pmed36_800_400	800	400	42
pmed37_800_160	800	160	33
pmed37_800_200	800	200	33
pmed37_800_270	800	270	33
pmed37_800_400	800	400	33
pmed38_900_90	850	90	28
pmed38_900_170	850	170	28
pmed38_900_180	900	180	40
pmed38_900_220	850	220	28
pmed38_900_230	900	230	40
pmed38_900_290	850	290	28
pmed38_900_300	900	300	40
pmed38_900_450	900	450	40
pmed39_850_90	850	90	22
pmed39_850_170	850	170	22
pmed39_850_220	850	220	22
pmed39_850_290	850	290	22
pmed39_900_180	900	180	74
pmed39_900_230	900	230	74
pmed39_900_300	900	300	74
pmed39_900_450	900	450	74
pmed40_850_90	850	90	27
pmed40_850_170	850	170	27
pmed40_850_220	850	220	27
pmed40_850_290	850	290	27
pmed40_900_180	900	180	23
pmed40_900_230	900	230	23
pmed40_900_300	900	300	23
pmed40_900_450	900	450	23

References

Ahmadi-Javid, A., Seyedi, P., & Syam, S. S. (2017). A survey of healthcare facility location. *Computers & Operations Research, 79*, 223–263.

Aikens, C. (1985). Facility location models for distribution planning. *European Journal of Operational Research, 22*, 263–279. https://doi.org/10.1016/0377-2217(85)90246-2. URL:http://www.sciencedirect.com/science/article/pii/0377221785902462.

Albareda-Sambola, M., Hinojosa, Y., Marín, A., & Puerto, J. (2015). When centers can fail: A close second opportunity. *Computers & Operations Research, 62*, 145–156. https://doi.org/10.1016/j.cor.2015.01.002

Alcaraz, J., Landete, M., Monge, J. F., & Sainz-Pardo, J. L. (2020). Multi-objective evolutionary algorithms for a reliability location problem. *European Journal of Operational Research, 283*, 83–93. https://doi.org/10.1016/j.ejor.2019.10.043

Almeida, B., Correia, I., & Saldanha-da Gama, F. (2018). A biased random-key genetic algorithm for the project scheduling problem with flexible resources. *TOP, 26*, 283–308. https://doi.org/10.1007/s11750-018-0472-9. URL:https://www.scopus.com/inward/record.uri?eid=2-s2.0-85045456587&doi=10.1007%2fs11750-018-0472-9&partnerID=40&md5=7a4b50d26dc68606eee0d34d08f05dd8 Cited By 1.

Andrade, C. E., Ahmed, S., Nemhauser, G. L., & Shao, Y. (2017). A hybrid primal heuristic for finding feasible solutions to mixed integer programs. *European Journal of Operational Research, 263*, 62–71. https://doi.org/10.1016/j.ejor.2017.05.003

Andrade, C. E., Byers, S. D., Gopalakrishnan, V., Halepovic, E., Poole, D. J., Tran, L. K., & Volinsky, C. T. (2019). Scheduling software updates for connected cars with limited availability. *Applied Soft Computing, 82*, Article 105575. https://doi.org/10.1016/j.asoc.2019.105575

Andrade, C. E., Resende, M. G. C., Karloff, H. J., & Miyazawa, F. K. (2014). Evolutionary algorithms for overlapping correlation clustering. In *Proceedings of the 16th conference on genetic and evolutionary computation GECCO'14* (pp. 405–412). New York, NY, USA: ACM. https://doi.org/10.1145/2576768.2598284.

Andrade, C. E., Resende, M. G. C., Zhang, W., Sinha, R. K., Reichmann, K. C., Doverspike, R. D., & Miyazawa, F. K. (2015). A biased random-key genetic algorithm for wireless backhaul network design. *Applied Soft Computing, 33*, 150–169. https://doi.org/10.1016/j.asoc.2015.04.016

Andrade, C. E., Silva, T., & Pessoa, L. S. (2019). Minimizing flowtime in a flowshop scheduling problem with a biased random-key genetic algorithm. *Expert Systems with Applications, 128*, 67–80. https://doi.org/10.1016/j.eswa.2019.03.007

Andrade, C. E., Toso, R. F., Gonçalves, J. F., & Resende, M. G. C. (2021). The multi-parent biased random-key genetic algorithm with implicit path-relinking and its real-world applications. *European Journal of Operational Research, 289*, 17–30. https://doi.org/10.1016/j.ejor.2019.11.037

Beasley, J. E. (1990). OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society, 41*, 1069–1072. https://doi.org/10.1057/jors.1990.166

Biajoli, F. L., Chaves, A. A., & Lorena, L. A. N. (2019). A biased random-key genetic algorithm for the two-stage capacitated facility location problem. *Expert Systems with Applications, 115*, 418–426. https://doi.org/10.1016/j.eswa.2018.08.024

Birattari, M., Yuan, Z., Balaprakash, P., & Stützle, T. (2010). F-Race and iterated F-Race: an overview. In *Experimental methods for the analysis of optimization algorithms* (pp. 311–336). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-02538-9_13.

Boonmee, C., Arimura, M., & Asada, T. (2017). Facility location optimization model for emergency humanitarian logistics. *International Journal of Disaster Risk Reduction, 24*, 485–498. https://doi.org/10.1016/j.ijdrr.2017.01.017

Brandão, J., Noronha, T., Resende, M., & Ribeiro, C. (2017). A biased random-key genetic algorithm for scheduling heterogeneous multi-round systems. *International Transactions in Operational Research, 24*, 1061–1077. https://doi.org/10.1111/itor.12429. URL:https://www.scopus.com/inward/record.uri?eid=2-s2.0-85019882030&doi=10.1111%2fitor.12429&partnerID=40&md5=fd80f9b96e15a229ca3dcb22318d3df5 Cited By 7.

Brandeau, M. L., & Chiu, S. S. (1989). An overview of representative problems in location research. *Management Science, 35*, 645–674. https://doi.org/10.1287/mnsc.35.6.645

Church, R. & ReVelle, C. (1974). The maximal covering location problem. In Papers of the regional science association (Vol. 32, pp. 101–118). Springer-Verlag.

Farahani, R. Z., Fallah, S., Ruiz, R., Hosseini, S., & Asgari, N. (2019). Or models in urban service facility location: A critical review of applications and future developments. *European Journal of Operational Research, 276*, 1–27. https://doi.org/10.1016/j.ejor.2018.07.036

Farahani, R. Z., & Hekmatfar, M. (2009). Facility location: Concepts, models, algorithms and case studies. *Springer*. https://doi.org/10.1007/978-3-7908-2151-2

de Faria, J. H., Resende, M. & Ernst, D. (2017). A biased random key genetic algorithm applied to the electric distribution network reconfiguration problem. Journal of Heuristics, 23, 533–550. URL:https://www.scopus.com/inward/record.uri?eid=2-s2.0-85027528177&doi=10.1007%2fs10732-017-9355-8&partnerID=40&md5=f90f1b4f70a1ce25aaa366ccb8879882. doi: 10.1007/s10732-017-9355-8. Cited By 5.

Gendreau, M. & Potvin, J. -Y. (2010). Handbook of metaheuristics, Vol. 2. Springer. doi: 10.1007/978-1-4419-1665-5.

Gonçalves, J. F., & Resende, M. G. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics, 17*, 487–525. https://doi.org/10.1007/s10732-010-9143-1

Gonçalves, J. F., & Resende, M. G. (2011). A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem. *Journal of Combinatorial Optimization, 22*, 180–201. https://doi.org/10.1007/s10878-009-9282-1

Gonçalves, J. F., & Resende, M. G. (2013). A biased random key genetic algorithm for 2d and 3d bin packing problems. *International Journal of Production Economics, 145*, 500–510. https://doi.org/10.1016/j.ijpe.2013.04.019. URL:http://www.sciencedirect.com/science/article/pii/S0925527313001837.

Güneş, E. D., Melo, T. & Nickel, S. (2019). Location problems in healthcare. In Location science (pp. 657–686). Springer. doi: 10.1007/978-3-030-32177-2_23.

Hall, N., & Hochbaum, D. (1992). The multicovering problem. *European Journal of Operational Research, 62*, 323–339. https://doi.org/10.1016/0377-2217(92)90122-P

Hochbaum, D. S., & Shmoys, D. B. (1985). A best possible heuristic for the k-center problem. *Mathematics of Operations Research, 10*, 180–184.

Hogan, K., & ReVelle, C. (1986). Concepts and applications of backup coverage. *Management Science, 32*, 1434–1444.

Huang, R., Kim, S., & Menezes, M. (2010). Facility location for large-scale emergencies. *Annals of Operations Research, 181*, 271–286. https://doi.org/10.1007/s10479-010-0736-8. URL:https://www.scopus.com/inward/record.uri?eid=2-s2.0-78649777583&doi=10.1007%2fs10479-010-0736-8&partnerID=40&md5=c35d028c1b5ca57931bc8c78facf4686 Cited By 71.

Jalali, S., Seifbarghy, M., Sadeghi, J., & Ahmadi, S. (2016). Optimizing a bi-objective reliable facility location problem with adapted stochastic measures using tuned-parameter multi-objective algorithms. *Knowledge-Based Systems, 95*, 45–57. https://doi.org/10.1016/j.knosys.2015.12.001. URL:http://www.sciencedirect.com/science/article/pii/S0950705115004633.

Jia, H., Ordóñez, F., & Dessouky, M. (2007). A modeling framework for facility location of medical services for large-scale emergencies. *IIE Transactions, 39*, 41–55. https://doi.org/10.1080/07408170500539113. URL:https://doi.org/10.1080/07408170500539113 arXiv:https://doi.org/10.1080/07408170500539113.

Jia, H., Ordóñez, F., & Dessouky, M. (2007). Solution approaches for facility location of medical supplies for large-scale emergencies. *Computers & Industrial Engineering, 52*, 257–276. https://doi.org/10.1016/j.cie.2006.12.007. URL:http://www.sciencedirect.com/science/article/pii/S0360835206002129.

Karatas, M., Razi, N., & Tozan, H. (2016). A comparison of p-median and maximal coverage location models with q-coverage requirement. Procedia Engineering, 149, 169–176. URL:http://www.sciencedirect.com/science/article/pii/S1877705816311602. doi: https://doi.org/10.1016/j.proeng.2016.06.652. International Conference on Manufacturing Engineering and Materials, ICMEM 2016, 6-10 June 2016, Nový Smokovec, Slovakia.

Karatas, M., & Yakici, E. (2019). An analysis of p-median location problem: Effects of backup service level and demand assignment policy. *European Journal of Operational Research, 272*, 207–218. https://doi.org/10.1016/j.ejor.2018.06.017. URL:http://www.sciencedirect.com/science/article/pii/S037722171830540X.

Kariv, O., & Hakimi, S. L. (1979). An algorithmic approach to network location problems. I: The p-centers. *SIAM Journal on Applied Mathematics, 37*, 513–538. https://doi.org/10.1137/0137040

Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika, 30*, 81–93. https://doi.org/10.2307/2332226

Khuller, S., Pless, R., & Sussmann, Y. (2000). Fault tolerant k-center problems. *Theoretical Computer Science, 242*, 237–245. https://doi.org/10.1016/S0304-3975(98)00222-9. URL:https://www.scopus.com/inward/record.uri?eid=2-s2.0-0041953557&doi=10.1016%2fS0304-3975%2898%2900222-9&partnerID=40&md5=1c68ed5a2646374584f0c14315f426f7.

Krömer, P., Platoš, J., & Snášel, V. (2016). Three types of differential evolution applied to the facility location problem. In A. Abraham, S. Kovalev, V. Tarassov, & V. Snášel (Eds.), Proceedings of the first international scientific conference "Intelligent Information Technologies for Industry" (IITI'16) (pp. 487–499). Cham: Springer International Publishing.

Lopes, M. C., de Andrade, C. E., de Queiroz, T. A., Resende, M. G., & Miyazawa, F. K. (2016). Heuristics for a hub location-routing problem. *Networks, 68*, 54–90. https://doi.org/10.1002/net.21685

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives, 3*, 43–58. https://doi.org/10.1016/j.orp.2016.09.002

López-Sánchez, A. D., Sánchez-Oro, J., & Hernández-Díaz, A. G. (2019). GRASP and VNS for solving the p-next center problem. *Computers & Operations Research, 104*, 295–303. https://doi.org/10.1016/j.cor.2018.12.017

Lourenço, H. R., Martin, O. C., & Stützle, T. (2019). Iterated local search: Framework and applications. In M. Gendreau, & J.-Y. Potvin (Eds.), *Handbook of metaheuristics* (pp. 129–168). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-91086-4_5. URL:https://doi.org/10.1007/978-3-319-91086-4_5.

Lucena, M. L., Andrade, C. E., Resende, M. G. C., & Miyazawa, F. K. (2014). Some extensions of biased random-key genetic algorithms. In *Proceedings of the 46th Brazilian symposium of operational research XLVI SBPO* (pp. 2469–2480). URL:http://www.din.uem.br/sbpo/sbpo2014/pdf/arq0357.pdf.

Martinez, C., Loiseau, I., Resende, M. & Rodriguez, S. (2011). Brkga algorithm for the capacitated arc routing problem. Electronic Notes in Theoretical Computer Science, 281, 69–83. URL:http://www.sciencedirect.com/science/article/pii/S1571066111001757. doi: 10.1016/j.entcs.2011.11.026. Proceedings of the 2011 Latin American Conference in Informatics (CLEI).

Melo, M. T., Nickel, S., & Saldanha-Da-Gama, F. (2009). Facility location and supply chain management–a review. *European Journal of Operational Research, 196*, 401–412. https://doi.org/10.1016/j.ejor.2008.05.007

Mousavi, S. M., Bahreininejad, A., Musa, S. N., & Yusof, F. (2017). A modified particle swarm optimization for solving the integrated location and inventory control problems in a two-echelon supply chain network. *Journal of Intelligent Manufacturing, 28*, 191–206. https://doi.org/10.1007/s10845-014-0970-z

Paliwal, A., Gimeno, F., Nair, V., Li, Y., Lubin, M., Kohli, P. & Vinyals, O. (2019). REGAL: Transfer learning for fast optimization of computation graphs. CoRR, abs/1905.02494. URL:http://arxiv.org/abs/1905.02494. arXiv:1905.02494.

Pessoa, L., & Andrade, C. (2018). Heuristics for a flowshop scheduling problem with stepwise job objective function. *European Journal of Operational Research, 266*, 950–962. https://doi.org/10.1016/j.ejor.2017.10.045. URL:https://www.scopus.com/inward/record.uri?eid=2-s2.0-85035039758&doi=10.1016%2fj.ejor.2017.10.045&partnerID=40&md5=33468c9171975dac2596a2455ea8b9b6 Cited By 2.

Pessoa, L., Resende, M., & Ribeiro, C. (2011). Experiments with lagrasp heuristic for set k-covering. *Optimization Letters, 5*, 407–419. https://doi.org/10.1007/s11590-011-0312-4. URL:https://www.scopus.com/inward/record.uri?eid=2-s2.0-79960122563&doi=10.1007%2fs11590-011-0312-4&partnerID=40&md5=22ca9b6e45fb56fc9eabbf5c32507aee.

Pessoa, L., Resende, M., & Ribeiro, C. (2013). A hybrid lagrangean heuristic with grasp and path-relinking for set k-covering. *Computers & Operations Research, 40*, 3132–3146. https://doi.org/10.1016/j.cor.2011.11.018. URL:https://www.scopus.com/inward/record.uri?eid=2-s2.0-84885956452&doi=10.1016%2fj.cor.2011.11.018&partnerID=40&md5=b069ef2d901a4e6e75da29f103662c4f.

Pessoa, L., Santos, A., & Resende, M. (2017). A biased random-key genetic algorithm for the tree of hubs location problem. *Optimization Letters, 11*, 1371–1384. https://doi.org/10.1007/s11590-016-1082-9. URL:https://www.scopus.com/inward/record.uri?eid=2-s2.0-84988664264&doi=10.1007%2fs11590-016-1082-9&partnerID=40&md5=c0fe921c11b41b90ee5323ea5be6de7b Cited By 3.

Rabie, H. M. (2020). Particle swarm optimization and grey wolf optimizer to solve continuous p-median location problems. In A.E. Hassanien, K. Shaalan, & M.F. Tolba (Eds.), Proceedings of the international conference on advanced intelligent systems and informatics 2019 (pp. 136–146). Cham: Springer International Publishing.

Resende, M. G. C. (2012). Biased random-key genetic algorithms with applications in telecommunications. *TOP, 20*, 130–153. https://doi.org/10.1007/s11750-011-0176-x. URL:10.1007/s11750-011-0176-x.

Ruiz, E., Soto-Mendoza, V., Ruiz Barbosa, A., & Reyes, R. (2019). Solving the open vehicle routing problem with capacity and distance constraints with a biased random key genetic algorithm. *Computers & Industrial Engineering, 133*, 207–219. https://doi.org/10.1016/j.cie.2019.05.002. URL:https://www.scopus.com/inward/record.uri?eid=2-s2.0-85065704445&doi=10.1016%2fj.cie.2019.05.002&partnerID=40&md5=c65c53f5d61a89569c46bf3c8d057724 Cited By 0.

Shavarani, S. M. (2019). Multi-level facility location-allocation problem for post-disaster humanitarian relief distribution. *Journal of Humanitarian Logistics and Supply Chain Management*.

Tansel, B. C., Francis, R. L., & Lowe, T. J. (1983). State of the art – location on networks: A survey. Part I: The p-center and p-median problems. *Management Science, 29*, 482–497. https://doi.org/10.1287/mnsc.29.4.482

Toregas, C., Swain, R., ReVelle, C., & Bergman, L. (1971). The location of emergency service facilities. *Operations Research, 19*, 1363–1373.