The p-next center problem with capacity and coverage radius constraints: model and heuristics

Mariana A. Londe[1], Luciana S. Pessoa[1], and Carlos E. Andrade[2]

[1] Department of Industrial Engineering, PUC-Rio
Rua Marquês de São Vicente, 225, Gávea - 22453-900 Rio de Janeiro, RJ, Brazil
mlonde@aluno.puc-rio.br
lucianapessoa@puc-rio.br

[2] AT&T Labs Research
200 South Laurel Avenue, Middletown, NJ 07748 USA
cea@research.att.com

Abstract. This paper introduces a novel problem of facility location, called the p-next center problem with capacity and coverage radius constraints. We formulate a mixed integer programming model for this problem, and compare the results found by CPLEX with three Biased Random-Key Genetic Algorithms variants. We also propose several instances for this problem, based on existing ones for the p-next center problem. Additionally, we analyze the effect of the radius and demand on instance difficulty. We also observe the performance gains with a relaxed capacity and demand constraint, i.e., permitting demand to be unmet by the model. Results point that the BRKGA variants had significantly better performance than CPLEX, and similar performances among themselves. Of those, `BRKGA-FI` was shown to have slightly better results than the other variants.

Keywords: Facility location · Biased Random-Key Genetic Algorithm · Metaheuristics

1 Introduction

The *p-center problem* is a classical location problem that consists of choosing p centers among n nodes in a network in order to minimize the maximum distance from any node to its closest facility [24].

The *p-next center problem* (pNCP) is an extension of the previous one that considers the possibility of a user arriving at a facility and discovering a disruption in its operation. In this scenario, the user must move to a backup point, defined as its nearest facility. The objective of the pNCP is to minimize the maximal distance a user must travel, which is composed by the distance from a point to its closest existing facility plus the distance from this facility to its backup. This problem was introduced in the context of humanitarian logistics by [1], since during an emergency there is a possibility of disruption in facilities such as shelters and hospitals. The authors also present several formulations, instances and the proof of its NP-Hardness. The first heuristic method to solve this problem was described by [19], based on the Greedy Randomized Adaptive Search Procedure (GRASP) [10] and Variable Neighborhood Search (VNS) [14]. Later, [17] presented a Biased Random Key Genetic Algorithm (BRKGA) [13] for the pNCP, alongside several new benchmarks.

The pNCP does not consider that a user may be unable to travel extensive distances during an emergency, nor the possible consequences of lack of capacity on the centers. For example, consider the case of a snake attack victim. The patient must quickly reach the closest treatment center, which may be located on the same neighborhood as the attack or on a close one. The speed of transport to the treatment facility is crucial for the patient‘s prospects, as the elapsed time between injury and care is a factor in severeness and lethality of the wound [6]. In fact, venom of some species of snake demand an elapsed time below a specific threshold. If there is enough anti-venom at the facility, then the patient will be efficiently treated and the emergency is resolved. However, the center may lack the medicine. There are two possible routes for treatment if this happens: (1) the patient is transferred to a supplied facility, and (2) the medicine is transported to them. On both cases the elapsed time between injury and therapy increases, and may lead to avoidable after-effects and/or death if higher then the time threshold.

Thus, the network of treatment centers must consider the maximum time elapsed between a possible attack and the closest care facility alongside the possible transference/transport time, which deal directly with the victim‘s perspective. The elapsed time must be below a specific threshold, otherwise the patient may have severe side-effects or death. It must also deal with a limited anti-venom capacity among all facilities, so that a center may be able to treat the highest possible number of patients without needing to bring medicine from other centers in the network. An unmet demand on a center, after all, is indicative of a probable death. Lastly, this network must deal with a demand for anti-venom that may be considered static, but distributed among many cities and/or neighborhoods.

This scenario has motivated us to present an extension of the pNCP, the *p-next center problem with capacity and coverage radius constraints*, referred as pNCPCR. We assume that the user does not know whether there is enough capacity in a center before arriving, which forces the user to go to a backup center. We also consider that there is a maximum distance between the beginning of the user’s journey and the arrival at the backup center, to simulate the elapsed time between injury and treatment. Thus, this problem unifies the pNCP with a single-source capacitated facility location problem [11] and a maximum coverage location problem [22]. The pNCPCR, thus, is an attempt at increasing the effectiveness of locating and allocating facilities in emergency situations, such as the snake attack mentioned previously.

The pNCPCR focus on minimizing the maximal distance traveled by the users which exceeds the coverage radius and the amount of unmet demand on the centers. We propose a mathematical model to represent it and, due to its difficulty in finding good solutions, a Biased Random-Key Genetic Algorithm (BRKGA) was customized for this problem. Additionally, we propose several instances to evaluate the algorithms performance, and observe the effects of the coverage radius and the center capacity on the results.

The remainder of this paper 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

2.1 Related work

The p-center problem has frequently been used for the assignment and location of emergency services and facilities, specially in the last decade. Huang et al. [15] studies a version of the p-center problem in which a facility cannot assume the demand of its own location, and thus needs to be allocated to another center. This situation is frequent in disaster situations, and is solved with dynamic programming. Morgan et al. [21] focus on properly allocating emergency service facilities during Islamic pilgrimages, in which excessive crowds are a concern. The authors point that, in regards to distance, coverage, and cover inequality, a genetic algorithm appears to have good balance between quality and computational time. Lastly, Yu et al. [28] considers that damages to the transport network can affect accessibility of emergency facilities, and thus introduces a multi-objective model that tries to guarantee minimum reachability of the facilities. Reachability is defined as a function of the level of damage on a given trajectory. The authors use a p-center problem to try to avoid the bi-level structure of the model, and test the approach on the Sioux Falls transportation network.

The capacitated facility location problem considers the presence of demand and capacity constraints on the facilities, something that interferes with the allocation of users to centers [11,26]. Among the works focused on this problem, Biajoli et al. [5] uses a BRKGA to solve the two-stage capacitated facility location problem. The same problem is the focus of Souto et al. [23], who use a hybrid matheuristic, composed by clustering search, adaptive large neighborhood search, and local branching techniques. Mauri et al. [20] studies a multi-product version of the previous problem. The authors use a BRKGA to solve it, and prove that it outperforms a clustering search approach.

Meanwhile, the maximal covering location problem focus on the coverage radius, i.e., the maximal distance between user and center allocation [9,22]. In this category, the study of Taiwo et al. [25] use a maximal covering location problem to identify potential locations for Covid-19 testing in Nigeria. The author tests several potential coverage radius, and observes their impact in the resulting network. Yang et al. [27] presents a continuous version of the maximal coverage location problem, with the aim of dynamically optimizing the organization of rescue in natural disasters. Amarilies et al. [2] uses greedy heuristics to solve the maximal coverage problem in the context of trashcan location. The results of this study were implemented on a village in Indonesia.

Several works unify the cited location problems, with the aim of increasing realism and modelling real-life-based situations appropriately. Karatas et al. [16] proposes a multi-objective facility location problem, with elements from both p-median, p-center, and maximum coverage. The authors focus on the design of a public emergency service network, and use a combination of branch-and-bound and iterative goal programming techniques to solve it. The study of Chauhan et al. [7] mixes the capacitated facility location problem with the maximal coverage problem. This was done in the context of drone launching sites. This study is extended in Chauhan et al. [8], where the uncertainty in battery consumption is also considered.

2.2 Problem formulation

The formulation of the pNCPCR is based on the two-indexed formulation introduced by [1]. In this formulation, there are nodes named $\{1, \cdots, n\}$ inside a network. Those nodes represent locations such as neighborhoods or cities, with the related distance between them. If a facility is located on one node, then it is responsible for the demand of both that point and of the closest locations without facilities. One should note that we use indexes i, j, and k to refer either to facility or non-facility nodes, which may also be referred as users. The parameters and variables of this model are presented in Table 1.

Table 1: Parameters and decision variables definitions.

Parameters	
$C \in \mathbb{N}$	Maximal capacity of the nodes
$N_i \in \mathbb{N}$	Demand of node i
$d_{ij} \in \mathbb{N}$	Distance between nodes i and j
$R \in \mathbb{N}$	Maximal distance between user and backup center
$p \in \mathbb{N}$	Number of facilities to be assigned
Decision variables	
$y_j \in \{0,1\}$	$y_j = 1$ if a facility is opened on node j
$x_{ij} \in \{0,1\}$	$x_{ij} = 1$ if center j is the closest to node i
$w_{kj} \in \{0,1\}$	$w_{kj} = 1$ if center k is the closest to center j
$z \in \mathbb{N}$	Maximal traveled distance
$t_{kj} \in \mathbb{N}$	Capacity transported from center k to center j
$u_j \in \mathbb{N}$	Used capacity in center j
$\kappa_j \in \mathbb{N}$	Unmet demand in center j
$\delta \in \mathbb{N}$	Exceeded traveled distance

$$\min \quad \delta^2 + \sum_{i=1}^{n} \kappa_i^2 \tag{1a}$$

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

$$\sum_{\substack{j=1\\ j\neq i}}^{n} x_{ij} = 1 \qquad \forall i \in \{1,\dots,n\} \tag{1c}$$

$$x_{ij} \leq y_j \qquad \forall_{\substack{i\neq j}}^{i,j\in\{1,\dots,n\}} \tag{1d}$$

$$y_j + \sum_{\substack{k=1\\ d_{ik}>d_{ij}}}^{n} x_{ik} \leq 1 \qquad \forall_{\substack{i\neq j}}^{i,j\in\{1,\dots,n\}} \tag{1e}$$

$$z \geq \sum_{\substack{k=1\\ k\neq j}}^{n} d_{jk}\cdot x_{jk} \qquad \forall j \in \{1,\dots,n\} \tag{1f}$$

$$z \geq d_{ij}\cdot(x_{ij} - y_i) + \sum_{\substack{k=1\\ k\neq j}}^{n} d_{jk}\cdot x_{jk} \qquad \forall_{\substack{i\neq j}}^{i,j\in\{1,\dots,n\}} \tag{1g}$$

$$z \leq R + \delta \tag{1h}$$

$$w_{jk} \leq x_{kj} \qquad \forall_{\substack{j\neq k}}^{j,k\in\{1,\dots,n\}} \tag{1i}$$

$$w_{jk} \leq y_k \qquad \forall_{\substack{j\neq k}}^{j,k\in\{1,\dots,n\}} \tag{1j}$$

$$t_{jk} \leq M\cdot w_{jk} \qquad \forall_{\substack{j\neq k}}^{j,k\in\{1,\dots,n\}} \tag{1k}$$

$$C\cdot y_j \geq u_j + \sum_{\substack{k=1\\ k\neq j}}^{n} t_{jk} \qquad \forall j \in \{1,\dots,n\} \tag{1l}$$

$$\sum_{\substack{i=1\\ j\neq i}}^{n} (x_{ij} - w_{ij})\cdot N_i + \sum_{\substack{k=1\\ k\neq j}}^{n} t_{jk} + N_j\cdot y_j - \kappa_j \leq u_j + \sum_{\substack{k=1\\ k\neq j}}^{n} t_{kj} \quad \forall j \in \{1,\dots,n\} \tag{1m}$$

$$\kappa_j \leq M\cdot y_j \qquad \forall j \in \{1,\dots,n\} \tag{1n}$$

One must note that variable x_{ij} has different meanings that depend on i being a user or a facility. In the former case, the variable indicates the assignment of a facility to a user. In the latter, j is the backup center of an existing facility. Related to this, variable w_{ji} only exists if both indexes belong to facilities, and always corresponds to the assignment of j as the backup of another. In this formulation, Objective Function (1a) focuses on minimizing the total assignment cost of the network. This cost has two components. The first component is the squared excess of the maximal distance between a user and its backup center, when compared with the coverage radius. This is done to simulate the higher chance of death due to an elapsed time between injury and treatment higher than the threshold for after-effects and death. The second component is the sum of the squared value of unmet demands of each center. In a situation such as a snake attack, an unmet demand would mean an avoidable death.

Constraint (1b) guarantees that only p centers exist. Constraints (1c) and (1d) assign a reference center for each node alongside preventing self-assignment of user nodes. Constraint (1e) imposes a minimal distance when allocating a user to a reference center. Constraints (1f) and (1g) ensure the correct value of the highest distance between a user and its backup center. This value is either the distance between a reference center and its corresponding backup, or the sum of the distances between a user and its reference center, and between that center and its backup. The value of the exceeded travelling distance in relation to the coverage radius is obtained in Constraint (1h).

Constraints (1i) and (1j) guarantee that the variable w_{ji} only exists if both indexes belong to facilities, and if j is the backup center of i. The existence of this variable permits the transport of capacity between the centers, as Constraint (1k) indicates. As an example, this transferred capacity could be the transport of medicine between facilities. Constraint (1l) ensures that the used and transported capacities do not exceed the total available capacity of the center. Finally, Constraints (1m) and (1n) regulate the flow of demand and capacity in a given center, alongside the existence of unmet demands. The sum of the demands of the users allocated to a center, of the capacity transported to other centers, and of the demand in the center should not exceed the sum of the capacity used in it and of the capacity transported to it. If this equilibrium is violated there is an amount of unmet demand in the center, which is then penalized in the objective function.

3 Customizing the BRKGA for the pNCPCR

We developed an algorithm based on the Multi-Parent Biased Random-Key Genetic Algorithm with Implicit Path-Relinking (BRKGA-MP-IPR [4]), which is a multi-parent variant of the standard BRKGA [13]. This algorithm was chosen due to its good performance in capacitated location problems [5,20]. In addition, BRKGA is the state-of-the-art algorithm for the pNCP [17].

3.1 Evolutionary process

The Biased Random-Key Genetic Algorithm (BRKGA) begins by creating p populations composed by $|\mathcal{P}|$ individuals, which are called chromosomes. Each gene of a chromosome is a real-value number in the interval $[0, 1]$.

The *decoder* procedure associates a solution and the corresponding fitness value with a chromosome. The individuals are then ranked by their fitness values. The solutions with highest quality belong to the elite set $\mathcal{P}_e$, while the remaining are in the non-elite set.

On each generation three procedures are performed to obtain new populations. The *Reproduction* procedure copies all chromosomes in the elite set. *Mutants generation* deletes $|\mathcal{P}_m|$ individuals from the non-elite set and randomly creates the same amount of individuals. The remaining $|\mathcal{P}| - |\mathcal{P}_e| - |\mathcal{P}_m|$ chromosomes are generated with *Crossover*.

For the Multi-Parent BRKGA, π_t parents are selected for the *Crossover* procedure. From those, π_e belong to the elite set. The parents' fitness values are ranked and associated with probabilities by the bias function $\Phi(r)$. Then, each gene is taken from a parent according to its rank, defined by the comparison of its fitness value among all parents. The steps of reproduction, mutants generation, and crossover procedures are repeated until a stopping criterion is met. If there is an improvement in the best solution on a given generation, the local search procedure detailed in Section 3.2 is performed on the new best solution.

An intensification strategy for BRKGA used in this study is the Implicit Path Relinking (IPR) procedure. Path-relinking explores the neighborhood obtained in the path between two distinct solutions [12]. IPR is considered implicit due to being performed on the chromosomes of the BRKGA solution space, not on the decoded solution [4]. After path-relinking, the algorithm may migrate some elite solutions between different populations, if g iterations have passed without improvement in the best solution. Likewise, if I_s generations have passed without improvement, then the shaking procedure presented in Section 3.2 may be called. Finally, if $I_s \cdot R_m$ iterations passed without improvement, then a full reset is performed.

3.2 Chromosome representation and decoder

For the pNCPCR, the chromosome is a vector with n genes, with n being the number of nodes in the network. The decoder procedure may be divided in four phases. The first phase is the selection of centers. In it, the chromosome is sorted and the first p nodes are chosen as reference centers. The second phase corresponds to the allocation of user nodes to the closest centers, and the computation of used capacity in each center.

In the third and fourth phases, we allocate backup centers to the reference centers. However, in the third phase the backup center is only chosen if it has enough extra capacity to comport the excess demand of the reference center. If there are no candidate backup centers that obey this constraint, the center is allocated its closest center as backup in the fourth phase, and the unmet demand of the center is recorded.

The fitness value of the solutions is obtained as such: the highest value from the distance to backup center among all nodes is compared with the coverage radius. If higher, the excess distance is penalized with its squared value. The squared value of the unmet demand is then added to the squared excess distance.

Warm-start solution. The introduction of good solutions in the initial population is noted to increase performance of the algorithm. For the pNCPCR, the constructive heuristic starts by selecting the first node as a center. The remaining are selected from the $p-1$ closest nodes to the first node. The decoding procedure is then performed to the resulting set of centers and non-centers.

Exploitation strategies. We consider three exploitation strategies apart from IPR on our approach. The local search procedure, the shaking procedure, and the reset procedure are noted to lead to better algorithm performance. The local search observes the solutions found by swapping a user and a center node. It may use the first improvement or the best improvement strategies. The shaking procedure randomly exchanges an amount of user and center nodes on the elite chromosomes, and the random restart of the non-elite solutions. The reset procedure is the random restart of all individuals.

4 Experimental results

4.1 Instances

We generated 1,652 instances derived from pNCP instances.There are four instance groups, whose differences lie in individual center capacity and coverage radius. The instances were based on the 132 proposed by [1] and the 281 proposed by [17].

The process of generating instances for the pNCPCR has three phases. The first is the definition of demand for each node, which was randomly obtained from the interval $[10, 50]$. The second phase is the calculus of the individual center capacity $C = (\sum_{i=1}^{n} D_i)/(p \cdot DC)$. The parameter "DC" in this equation refers to the demand/capacity equilibrium of a given instance. If the instance is in category "high demand", then the sum of demands is 85% of the total capacity. If it belongs to "low demand", then this percentage is equal to 40%.

The third phase of the instance generation process is the selection of the coverage radius. As the original instance solution values have a median of approximately 95, and a minimum of about 45, the "high radius" category has its distance limit randomly chosen in the interval $[45, 95]$. It is expected that, due to the capacity and demand of the nodes, the maximal distances will increase in relation to the original instances. The "low radius" category has values randomly chosen in the interval $[15, 30]$, i.e. at one-third of the previous category.

Each instance name refers to its number of nodes, number of centers, demand category, and radius category, in that order. As an example, instance `pmed1_10_5_h_l` has 10 nodes and 5 centers, and belongs to the "high demand" and "low radius" categories.

4.2 Computational environment and parameter settings

The computational experiments were performed in a cluster of identical machines with an Intel Xeon E5530 CPU at 2.40GHz and 120 GB of RAM running CentOS Linux. The formulation proposed in Section 2 was solved with IBM ILOG CPLEX 20.1 solver. Heuristics proposed in this paper were implemented in C++ language using the BRKGA-MP-IPR framework [4]. All BRKGA variants use four threads, and all runs are limited to 30 wall-clock minutes or 1,000 generations without improvement on the best solution.

We run CPLEX with three 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 of 30 minutes (`CPLEX-30min`). 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 day, using 24 threads (`CPLEX-1d`). Such configuration looks to find optimal solutions or, at least, compute the best possible bounds. Since this configuration uses far more time and computer power per run than the other configuration, we use the results only for reporting. The third setup removes the parameter κ_i from the formulation, which turns constraint (1m) into a non-relaxed version (`CPLEX-ST`). In practice, this means that it is forbidden to have unmet demands on the centers. This configuration was meant to observe the effect of the relaxation of the constraint, and its results portrait a more realistic approach.

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. The parameters used for the BRKGA variations were suggested by the `irace` package [18], and may be seen in Table 2.

4.3 Mathematical model results

Our first task is to find optimal solutions or as-best-as-possible solutions for the 1,652 instances.

Note that Model (1) admits at least one feasible solution for each instance. However, this is not true for `CPLEX-ST` due to the use of the hard demand constraint, for which there are instances without feasible solutions.

Table 3 shows the performance of the three CPLEX variants. `CPLEX-1d` had the least amount of instances without any solution found, at 722 (44%), with a significantly smaller average Gap%. This is expected, as it has considerably more computational power and available time to explore the different solutions. Following it is `CPLEX-30min`, with 1,169 (70%) infeasible instances, and, lastly, `CPLEX-ST`, with 1,258 (76%). Again, this scenario is not unexpected, as the use of a hard constraint would diminish the pool of feasible solutions.

Now the effect of the radius and demand category is explored in regards to instance feasibility and non-optimum runs. One should note that several instances considered infeasible by `CPLEX-30min`, with the higher computational time from `CPLEX-1d`, were considered feasible and, on some cases, an optimum solution was found. This means that, in this case, we may consider an infeasible instance as a non-optimal one, and need to combine the results of those two categories in the analysis. Table 4 presents the percentage of instances without an optimal solution for the three CPLEX variants regarding their demand and radius categories. Note that for all cases, CPLEX has more difficulty in finding feasible solutions on the high demand category. In fact, the effect of the demand is more severe than that of the radius – something reasonable considering Model (1).

Table 2: Best parameter configurations suggested by `irace` for BRKGA variantions.

	BRKGA						IPR				Shaking		
	$\|\mathcal{P}\|$	$\mathcal{P}_e$%	$\mathcal{P}_m$%	π_e,π_t	Φ	p	*md*	*sel*	*ps*%	I_{ipr}	I_s	R_m	LS%
`BRKGA-BI`	4085	0.22	0.22	5,10	r^{-2}	2	0.27	RE	0.23	226	60	1.86	0.96
`BRKGA-FI`	3912	0.28	0.46	5,10	r^{-2}	3	0.05	RE	0.63	471	100	1.77	–
`BRKGA-NLS`	4075	0.20	0.12	5,10	r^{-2}	1	0.08	RE	0.83	206	96	1.71	–

Table 3: Model performance on all instances. Columns "Dem." and "Rad." detail the demand and radius categories, respectively. Column "# Opt" presents the number of instances with optimum found. Column "# NoSol" shows the amount of instances considered infeasible by the respective algorithm, i.e. the configuration found no feasible solutions for the instance. Column "# Fea" has the number of instances in which feasible and non-optimal solutions were found. "Avg. Gap%" is the average percentage of gap for non-optimal and non-infeasible instances. The fifth line in each section presents the summary of results for all instances, independently of demand and radius categories.

	Dem.	Rad.	# Opt	# NoSol	# Fea	Avg. Gap%
`CPLEX-30min`	High	High	43	78	292	84
	High	Low	45	77	291	63
	Low	High	87	290	36	93
	Low	Low	81	296	36	80
	Both	Both	256	1,169	227	78
`CPLEX-1d`	High	High	120	164	120	89
	High	Low	108	193	108	64
	Low	High	36	173	36	75
	Low	Low	35	192	35	63
	Both	Both	299	722	299	75
`CPLEX-ST`	High	High	41	337	41	94
	High	Low	43	333	43	82
	Low	High	31	295	31	84
	Low	Low	38	293	38	74
	Both	Both	153	1,258	153	84

The effect of the radius category is more subtle, and becomes more apparent with the longer running time and computational power of `CPLEX-1d`.

4.4 BRKGA results

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 [3], with a small modification to prevent division by zero. Let $\mathcal{I}$ be a set of instances. Let $\mathcal{A}$ be the set of algorithms, and assume that set $\mathcal{R}_A$ enumerates the independent runs for algorithm $A \in \mathcal{A}$ (as defined in Section 4.2, 30 runs for the heuristics and one run for `CPLEX-30min`). 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. In order to deal with the possibility of having a best known solution equal to zero, which happens on 11 instances, we introduce the constant $c^* > 0$ on the computation. This is done to prevent the division by

Table 4: CPLEX results in regards to categories and percentage of infeasible and non-optimal instances. The percentages presented refer to the amount of non-optimal and infeasible instances in relation to the total.

		High Rad. (%)	Low Rad. (%)
`CPLEX-30min`	High Dem.	90	89
	Low Dem.	79	80
`CPLEX-1d`	High Dem.	69	73
	Low Dem.	50	55
`CPLEX-ST`	High Dem.	91	91
	Low Dem.	79	80

zero that would happen on those cases. The RPD from the best solution i is, thus, defined as

$$RPD_{ir}^{A} = \frac{C_{ir}^{A} - C_{i}^{best} + c^{*}}{C_{i}^{best} + c^{*}} \times 100, \quad \forall A \in \mathcal{A},\ i \in \mathcal{I},\ \text{and}\ r \in \mathcal{R}_A. \tag{2}$$

The BRKGA variants did not have the same difficulties as `CPLEX-30min` with infeasible solutions, finding at least one feasible solutions for all instances. Figure 1 presents a boxplot with RDP distributions for each algorithm. Note that the y-axis is plotted on a log scale to enhance the visualization. The reason is that most of the algorithms found optimal solutions frequently, skewing the display on a linear scale. In fact, the median of the RPD distributions of the three BRKGA variants was zero, which indicates that at least half of the instances reached the best or optimal solution.

However, all three algorithms had runs with considerably high RPDs, something that skews the results. In fact, the mean of all distributions was higher than the value of the 0.75 percentile, and the maximal RPD for `BRKGA-BI` and `BRKGA-FI` variants was 15,132. `BRKGA-NLS` had a maximal RPD of 15,902. `BRKGA-NLS` presented slight better results than its counterparts, with $1,486 \pm 2,952\%$; `BRKGA-FI` produced $1,475 \pm 2,937\%$; and `BRKGA-BI` had an average of $1,480 \pm 2,943\%$. Since those results are too close to call, we applied the pairwise Wilcoxon rank-sum test with Bonferroni p-value adjustment method among all algorithms. With a confidence interval of 95%, we cannot affirm there is a significant difference on the results of the three BRKGA variations.

Table 5 presents the results in regards to the instances with optimal and non-optimal found by `CPLEX-30min`. One may note the BRKGA variants had a poor performance on the instances with known optima, finding the optimal solution on only 5% of instances and on circa 5% of runs. The performance of all three BRKGA variants was considerably similar for those instances. For instances with unknown optima, BRKGA performance was considerably better. In fact, all three variants found the best known solution for those instances in, in average, 80% of instances. This was done on at least 70% of runs. One

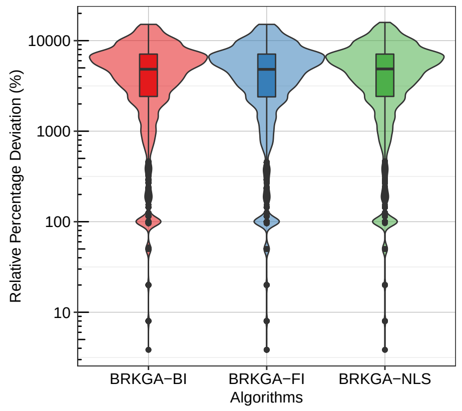

Fig. 1: Distribution of relative percentage deviations for each algorithm. Note that, since the data is plotted in log scale, zero deviations are not shown, although the algorithms have reached them.

Table 5: Algorithm performance on all instances.

	Known Optima (256 instances)			Unknown Optima (1340 instances)		
Algorithm	# Opt	% Opt	% Run	# Best	% Best	% Run
`BRKGA-BI`	13	5.08	5.01	1106	79.23	71.91
`BRKGA-FI`	13	5.08	5.08	1146	82.09	74.35
`BRKGA-NLS`	13	5.08	5.07	1143	81.88	73.03

may note that `BRKGA-FI` had the best performance among the algorithms, with higher percentages on all criteria. `BRKGA-BI` had the worst performance of the algorithms.

Finally, Figure 2 presents the cumulative probability of finding the best or optimal solution in relation to running time of the algorithms. Note that all BRKGA variants had better performances than `CPLEX-30min` on 1,800 seconds. In addition, `BRKGA-FI` is shown to be better than the other variants, with higher chance of finding the best solution on lower times. Note also that all three BRKGA variants tended to finish before the maximum permitted time. This means the BRKGA is reaching the upper limit on the number of iterations without improving the best solution, which indicates quick convergence.

When considering all results presented in this section, there are many conclusions that may be observed. The first is that BRKGA was shown to have a better performance than that of `CPLEX-30min`, when considering feasible solutions and cumulative probability of finding the best solution. The second consideration is that `BRKGA-FI` performed better than the other BRKGA variants, even if we cannot affirm the presence of a difference between the RPD distributions of the three algorithms. The last conclusion is that the BRKGA variants converged quickly, and tends to find the optimal solutions, as shown by the medians of the RPD distributions being zero.

5 Conclusions

In this work, we presented the p-next center problem with capacity and coverage radius constraints, referred as pNCPCR. This is an extension of the p-next center problem introduced by [1], itself an extension of the classical p-center problem. The novel pNCPCR was inspired on a situation of a snake attack, in which the victim cannot be treated on the closest facility due to a lack of medicine. We formulate a mathematical model for the pNCPCR, and develop a Biased Random-Key Genetic Algorithm (BRKGA) to solve this problem.

In order to observe the effectiveness of our proposed approach, we generated 1,652 instances based on the ones used in [17], and divided in four categories. To find optimal solutions to those instances, we used three CPLEX configurations. Of those, `CPLEX-30min` is used in the comparison with BRKGA. We also observed the effects of the flow constraint in the proposed model, by running `CPLEX-ST` with a harder version of this constraint.

`CPLEX-30min` results were compared with BRKGA variants, which differ in local search approaches. One may node that all three BRKGA variants had better performances than `CPLEX-30min`, and that the performances of those algorithms was very similar, with `BRKGA-FI` being slightly better than the other variants.

There are several possible future works based on this research. One may consider non-uniform facility capacities, something that the model and instances proposed in this paper do not. In addition, the problem could also be formulated as a two-stage problem with stochastic demand.

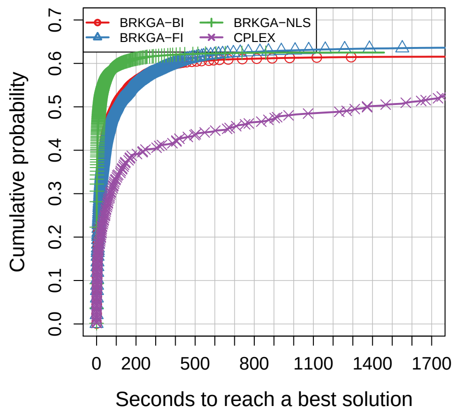

Fig. 2: Running time empirical distributions to the best solution values for all instances. The identification marks correspond to 2% of the points plotted for each algorithm.

Acknowledgements This study was financed in part by CNPq, PUC–Rio, FAPERJ (Project Numbers E-26/211.086/2019, E-26/211.389/2019, and E-26/211.588/2021), and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

References

1. Albareda-Sambola, M., Hinojosa, Y., Marín, A., Puerto, J.: When centers can fail: A close second opportunity. Computers & Operations Research **62**, 145–156 (2015). https://doi.org/10.1016/j.cor.2015.01.002
2. Amarilies, H.S., Redi, A.P., Mufidah, I., Nadlifatin, R.: Greedy heuristics for the maximum covering location problem: A case study of optimal trashcan location in kampung cipare–tenjo–west java. In: IOP Conference Series: Materials Science and Engineering. vol. 847, p. 012007. IOP Publishing (2020)
3. Andrade, C.E., Silva, T., Pessoa, L.S.: Minimizing flowtime in a flowshop scheduling problem with a biased random-key genetic algorithm. Expert Systems with Applications **128**, 67–80 (Aug 2019). https://doi.org/10.1016/j.eswa.2019.03.007
4. Andrade, C.E., Toso, R.F., Gonçalves, J.F., Resende, M.G.C.: The multi-parent biased random-key genetic algorithm with implicit path-relinking and its real-world applications. European Journal of Operational Research **289**(1), 17–30 (2021). https://doi.org/10.1016/j.ejor.2019.11.037
5. Biajoli, F.L., Chaves, A.A., Lorena, L.A.N.: A biased random-key genetic algorithm for the two-stage capacitated facility location problem. Expert Systems with Applications **115**, 418–426 (2019). https://doi.org/10.1016/j.eswa.2018.08.024
6. Bochner, R., Fiszon, J.T., Machado, C., et al.: A profile of snake bites in brazil, 2001 to 2012. Clinical Toxicology (2014). https://doi.org/10.4172/2161-0495.1000194
7. Chauhan, D., Unnikrishnan, A., Figliozzi, M.: Maximum coverage capacitated facility location problem with range constrained drones. Transportation Research Part C: Emerging Technologies **99**, 1–18 (2019). https://doi.org/https://doi.org/10.1016/j.trc.2018.12.001
8. Chauhan, D.R., Unnikrishnan, A., Figliozzi, M., Boyles, S.D.: Robust maximum coverage facility location problem with drones considering uncertainties in battery availability and consumption. Transportation Research Record **2675**(2), 25–39 (2021)
9. Drezner, Z., Hamacher, H.W.: Facility location: applications and theory. Springer Science & Business Media (2004)
10. Feo, T.A., Resende, M.G.: Greedy randomized adaptive search procedures. Journal of global optimization **6**(2), 109–133 (1995). https://doi.org/https://doi.org/10.1007/BF01096763
11. Filippi, C., Guastaroba, G., Speranza, M.G.: On single-source capacitated facility location with cost and fairness objectives. European Journal of Operational Research **289**(3), 959–974 (2021). https://doi.org/https://doi.org/10.1016/j.ejor.2019.07.045
12. Glover, F.: Tabu search and adaptive memory programming—advances, applications and challenges. In: Interfaces in computer science and operations research, pp. 1–75. Springer (1997)
13. Gonçalves, J.F., Resende, M.G.C.: Biased random-key genetic algorithms for combinatorial optimization. Journal of Heuristics **17**(5), 487–525 (2011). https://doi.org/10.1007/s10732-010-9143-1

14. Hansen, P., Mladenović, N.: Variable neighborhood search. In: Search methodologies, pp. 211–238. Springer (2005). https://doi.org/https://doi.org/10.1007/0-387-28356-0_8
15. Huang, R., Kim, S., Menezes, M.B.: Facility location for large-scale emergencies. Annals of Operations Research **181**(1), 271–286 (2010). https://doi.org/10.1007/s10479-010-0736-8
16. Karatas, M., Yakıcı, E.: An iterative solution approach to a multi-objective facility location problem. Applied Soft Computing **62**, 272–287 (2018)
17. Londe, M.A., Andrade, C.E., Pessoa, L.S.: An evolutionary approach for the p-next center problem. Expert Systems with Applications **175** (2021). https://doi.org/10.1016/j.eswa.2021.114728
18. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3**, 43–58 (2016). https://doi.org/10.1016/j.orp.2016.09.002
19. López-Sánchez, A.D., Sánchez-Oro, J., Hernández-Díaz, A.G.: GRASP and VNS for solving the p-next center problem. Computers & Operations Research **104**, 295–303 (2019). https://doi.org/10.1016/j.cor.2018.12.017
20. Mauri, G.R., Biajoli, F.L., Rabello, R.L., Chaves, A.A., Ribeiro, G.M., Lorena, L.A.N.: Hybrid metaheuristics to solve a multiproduct two-stage capacitated facility location problem. International Transactions in Operational Research **28**(6), 3069–3093 (2021). https://doi.org/10.1111/itor.12930
21. Morgan, A.A., Khayyat, K.M.J.: Improving emergency services efficiency during islamic pilgrimage through optimal allocation of facilities. International Transactions in Operational Research **29**(1), 259–300 (2022). https://doi.org/10.1111/itor.13026
22. Murray, A.T.: Maximal coverage location problem: impacts, significance, and evolution. International Regional Science Review **39**(1), 5–27 (2016). https://doi.org/https://doi.org/10.1177/0160017615600222
23. Souto, G., Morais, I., Mauri, G.R., Ribeiro, G.M., González, P.H.: A hybrid matheuristic for the two-stage capacitated facility location problem. Expert Systems with Applications **185**, 115501 (2021)
24. Suzuki, A., Drezner, Z.: The p-center location problem in an area. Location science **4**(1-2), 69–82 (1996). https://doi.org/https://doi.org/10.1016/S0966-8349(96)00012-5
25. Taiwo, O.J.: Maximal covering location problem (mclp) for the identification of potential optimal covid-19 testing facility sites in nigeria. African Geographical Review **40**(4), 395–411 (2021)
26. Wu, L.Y., Zhang, X.S., Zhang, J.L.: Capacitated facility location problem with general setup cost. Computers & Operations Research **33**(5), 1226–1241 (2006). https://doi.org/https://doi.org/10.1016/j.cor.2004.09.012
27. Yang, P., Xiao, Y., Zhang, Y., Zhou, S., Yang, J., Xu, Y.: The continuous maximal covering location problem in large-scale natural disaster rescue scenes. Computers & Industrial Engineering **146**, 106608 (2020)
28. Yu, W.: Reachability guarantee based model for pre-positioning of emergency facilities under uncertain disaster damages. International journal of disaster risk reduction **42**, 101335 (2020). https://doi.org/10.1016/j.ijdrr.2019.101335