Improving the Quality of Heuristic Solutions for the Capacitated Vertex p-Center Problem through Iterated Greedy Local Search and Variable Neighborhood Descent

Dagoberto R. Quevedo-Orozco
Graduate Program in Systems Engineering
Universidad Autónoma de Nuevo León, Mexico
dago@yalma.fime.uanl.mx

Roger Z. Ríos-Mercado
Graduate Program in Systems Engineering
Universidad Autónoma de Nuevo León, Mexico
roger.rios@uanl.edu.mx

November 2013

Abstract

The capacitated vertex p-center problem is a well-known location problem that consists of placing p facilities and assigning customers to each of these facilities so as to minimize the largest distance between any customer and its assigned facility, subject to demand capacity constraints for each facility. In this work, a metaheuristic for this location problem that integrates several components such as greedy randomized adaptive search procedures with probabilistic sampling, iterated greedy local search and variable neighborhood descent, is presented. Empirical evidence over a widely used set of benchmarks on location literature reveals the positive impact of each of the developed components. Furthermore, it is found empirically the proposed heuristic outperforms the best existing heuristic for this problem.

Keywords: Combinatorial optimization; Discrete location; Capacitated p-Center Problem; Metaheuristics; Iterated greedy local search; Variable neighborhood search.

1 Introduction

In this paper, the capacitated vertex p-center problem (CpCP) is addressed. This is a well-known location problem that can be defined as follows. Given a set of vertices V representing customers and potential facility location points, where each node $j \in V$ has a demand w_j and a facility capacity s_j, a set of edges E representing distance between nodes with value d_{ij}, and a given number p of desired facilities, we must decide where to locate p facilities and how to assign all the customers to these facilities so as to minimize the largest distance between any customer and its assigned facility subject to the demand capacity constraint. The CpCP is $\mathcal{NP}$-hard [9]. Several practical applications can be modeled as a CpCP, particularly problems that arise on emergency situations such as providing emergency medical service by carefully locating emergency facilities or ambulances, or the location of fire stations that will provide a prompt response and rapid transportation of equipment. In these situations it is clear that the cost for preserving human life is more important [3].

Plenty of work has been devoted to the uncapacitated version of this problem. Elloumi et al. [6] provide an extensive review of the literature. However, the research on the CpCP has been more limited. The most significant contributions from the exact optimization perspective are due to Özsoy and Pınar [14] and Albareda-Sambola et al. [1]. In the former, the authors presented an exact method based on solving a series of set covering problems using an off-the-shelf mixed-integer programming (MIP) solver while carrying out an iterative search over the coverage distances. In the latter, the authors proposed an exact method based on Lagrangian relaxation and a covering reformulation.

To the best of our knowledge, the most significant heuristic method for the CpCP is due to Scaparra et al. [18]. In their work, the authors developed a heuristic based on large-scale local search with a multiexchange neighborhood represented by an improved graph, exploiting principles from network optimization theory. They address instances of up to 402 nodes and 40 facilities.

The main purpose of our work is to integrate several of the components that have been very successful in other combinatorial optimization problems into a metaheuristic algorithm for the CpCP. To this end, a construction heuristic based on a greedy randomized adaptive search procedure with probabilistic sampling is developed. In addition, this is enhanced by a local search phase combining iterated greedy local search and variable neighborhood descent. In each of these components, the particular problem structure is adequately exploited. Our empirical work indicates the positive impact of each of these components when integrated into the metaheuristic. When compared to the existing work, the proposed heuristic provides solutions of better quality than those found by the other heuristic under similar, and in many cases significantly less, computational effort.

The rest of the paper is organized as follows. The combinatorial optimization model for the CpCP is given in Section 2. In Section 3, a detailed description of the proposed algorithm is prsented. Section 4 shows the empirical results providing a descriptions of the data sets, parameter fine-tunning, comparison among methods, and component analysis of the proposed heuristic. Concluding remarks are given in Section 5.

2 Problem Formulation

Let V be the set of nodes representing customers or potential locations for the p facilities. The integer distance between nodes i and j is represented for d_{ij}. Each node $j \in V$ has a demand or weight w_j and each node $i \in V$ has a capacity defined by s_i. For the combinatorial model, a p-partition of V is denoted by $X = \{X_1, ..., X_p\}$ and $K = \{1 : p\}$, where $X_k \subset V$ is called a subset of V. Each subset X_k is formed by a subset of nodes such that $\bigcup_{k \in K} X_k = V$ and $X_k \cap X_l = \varnothing$ for all $k, l \in K, k \neq l$. The set of centers is denoted by $P \subset V$ such that $P = \{c(1), ..., c(p)\}$ where $c(k)$ is the active location for subset X_k, i.e., the node that hosts the facility serving the customers in X_k. The problem can be represented by the following combinatorial model.

$$\min_{X \in \Pi} \quad \max_{k \in K} f(X_k), \tag{1}$$

where Π is the collection of all p-partitions of V. For a given territory X_k its cost function, also called the bottleneck cost, is computed as $f(X_k) = \max_{j \in X_k}\{d_{j,c(k)}\}$ where the center $c(k)$, taking into account the capacity, is given by

$$c(k) = \arg\min_{i \in X_k} \left\{ \max_{j \in X_k} \left\{ d_{ij} : \sum_{j' \in X_k} w_{j'} \leq s_i \right\} \right\}. \tag{2}$$

Here, by convention, if for a given X_k there is not any $i \in X_k$ such that $\sum_{j \in X_k} w_j \leq s_i$ then $f(X_k) = \infty$. We define by $\ell(j)$ the center k serving customer j.

3 Description of the Heuristic

In the recent years, one important trend in the metaheuristic field is that of integrating different components resulting in successful hybrid methods that attempt to better exploit the specific problem structure. As a direct consequence of this, sometimes it is not clear how to name a specific heuristic as it uses ideas from several methods. In this regard, the proposed solution method uses a greedy randomized adaptive search procedure as its guiding framework. In its construction phase, a selection based on biased sampling is employed. Then Iterated Greedy Local Search (IGLS) and Variable Neighborhood Descent (VND) are applied in the improvement phase. IGLS is a technique originally proposed by Ruiz and Stützle [17] that extends the Iterated Local Search (ILS) heuristic. IGLS iteratively applies destruction and reconstruction to a given input solution focusing on the space of locally optimal solutions. IGLS iterates over a greedy reconstruction heuristic instead of iterating over a local search as done in ILS.

VND is an iterative improvement algorithm that realizes the general idea behind Variable Neighborhood Search (VNS). In VND, k neighborhoods are used, which are typically ordered according to increasing size. The algorithm starts with the first neighborhood and performs iterative improvement steps until local optimality is reached. Whenever no further improving step is found for the i-th neighborhood and $i+1 \leq k$, VND continues the search in the $(i+1)$-th neighborhood; if an improvement is found, the search process starts again at the first neighborhood. It has been shown that VND can considerably improve the performance of iterative improvement algorithms with respect to both the solution quality of the final solution and the time required for finding high-quality solutions compared to using standard iterative improvement in large neighborhoods [8]. The proposed approach is presented in Algorithm 1.

Algorithm 1 IGLS-VND

```
 1: procedure IGLS_VND($V, p, \alpha, Iter_{\max}$)
 2:     $X \leftarrow$ Construction($V, p$)                         ▷ Construction
 3:     $X \leftarrow$ VND($X$)
 4:     $X^{\text{best}} \leftarrow X$
 5:     while ¬(stopping criteria) do                               ▷ Local Search
 6:         $X \leftarrow$ Perturbation($X, \alpha$)
 7:         $X \leftarrow$ VND($X$)
 8:         if $X$ is better that $X^{\text{best}}$ then
 9:             $X^{\text{best}} \leftarrow X$
10:         else
11:             $X \leftarrow$ Shake($X$)
12:         end if
13:         $Iter_{\max} \leftarrow Iter_{\max} - 1$
14:     end while
15:     return $X^{\text{best}}$
16: end procedure
```

An initial solution is obtained in Steps 2–3. Within the main loop (Steps 5–14), the local search (Steps 6–7) is performed as long as the solution keeps improving. We use an effective improvement criterion proposed in [18] which includes the reduction of bottleneck elements, defined as:

$$f(X') < f(X) \vee (f(X') = f(X), \mathcal{B}(X') \subseteq \mathcal{B}(X), \mathcal{J}(X') \subset \mathcal{J}(X)), \tag{3}$$

where $\mathcal{B}(X)$ denote the set of bottleneck subsets in X, i.e., $\mathcal{B}(X) = \{k \in K : f(X_k) = f(X)\}$

and $\mathcal{J}(X)$ contains the demand nodes with maximum distance from the active location in each subset X_k, i.e., $\mathcal{J}(X) = \{j \in X_k : d_{jc(k)} = f(X), k \in \mathcal{B}(X)\}$. This criteria is met if it decreases the objective function value or if it reduces the number of bottleneck customers while not worsening the total cost, without creating new bottleneck subsets and new bottleneck customers. The incumbent solution X^{best} is updated if a better feasible solution is found according to the criterion (3) otherwise a shake (Step 11) of the solution X is applied. The approach stops when the maximum number of iterations is met. These components are described next.

3.1 Construction

The construction phase consists of two stages: (a) center location and (b) customer allocation. In the former, we used a strategy based on greedy randomized adaptive search with probabilistic selection which diversifies the set of potential centers. In the latter, a deterministic greedy approach based on a distance criterion and capacity constraints is performed.

Stage (a): Center location

The goal here is to choose an adequate set of p centers or seed nodes. The choice of these centers is made through a greedy randomized adaptive construction procedure, taking into account the distance factors and the capacity of each vertex $j \in V$. This phase is based on the greedy method proposed by Dyer and Frizie [5] for the p-center problem. The location phase starts by choosing the first center randomly. Then, we iteratively choose the next center seeking a node whose weighted distance from its nearest center is relatively large. The motivation of this is to try to obtain centers that are as disperse as possible, but also to favor the choice of centers with large capacity such we can assign more customers to it in the allocation phase. Within a greedy randomized procedure method this is done as follows. Let P be a partial set of chosen centers. Then for each $j \in V \setminus P$, its nearest center is given by $i^* = \arg\min_{i \in P}\{d_{ij}\}$. The we compute the greedy function as

$$\gamma(j) = s_j d_{i^*j}. \tag{4}$$

We choose the next element i^* using a probability value determined by greedy function (4). The probability $\pi(j)$ of selecting element $j \in V \setminus P$ can be computed as $\pi(j) = \gamma(j)/\sum_{j' \in V \setminus P} \gamma(j')$.

Stage (b): Customer allocation

Once the centers are fixed, the second stage consists of allocating the customers to these centers. This stage is performed in a deterministic greedy manner. As some preliminary testing showed, performing this step under a randomized greedy strategy did not bring any value to the quality of the solution. In addition, the pure greedy approach in this phase is more efficient. The customers are defined by the remaining nodes $j \in V \setminus P$. To this end we define a greedy function that measures the cost of assigning a customer j to a center k located in $c(k)$ as follows:

$$\phi(j,k) = \max\left\{\frac{d_{jc(k)}}{\bar{d}}, -r(k) + w_j\right\}, \tag{5}$$

where $\bar{d} = \max_{i,j \in V}\{d_{ij}\} + 1$ is a normalization factor and $r(k) = s_{c(k)} - \sum_{j' \in X_k} w_{j'}$ is the residual capacity for the set whose center k located in $c(k)$. If the capacity constraint is satisfied, the function only takes into account the distance factor; otherwise, the function returns an integer value that penalizes the assignment. Then each node j is assigned to its nearest center, namely $X_{k^*} \leftarrow X_{k^*} \cup \{j\}$ where $k^* = \arg\min_{k \in K} \phi(j,k)$. Finally, once the assignment is done, the centers for the entire partition are updated using (2). Algorithm 2 depicts the construction method.

Algorithm 2 Construction

```
 1: procedure CONSTRUCTION($V, p$)
 2:     $P \leftarrow \varnothing$                                        ▷ Stage ($a$)
 3:     Choose $i^* \in V$ randomly
 4:     $P \leftarrow P \cup \{i^*\}$
 5:     Update $\gamma(j)$ and $\pi(j), j \in V \setminus P$
 6:     while $|P| < p$ do
 7:         Choose $i^* \in V \setminus P$ randomly using a probability $\pi(j)$
 8:         $P \leftarrow P \cup \{i^*\}$
 9:         Update $\gamma(j)$ and $\pi(j), j \in V \setminus P$
10:     end while
11:     $X \leftarrow \varnothing$                                        ▷ Stage ($b$)
12:     for all $j \in V \setminus P$ do
13:         $k^* \leftarrow \arg\min_{k \in K} \phi(j,k)$
14:         $X_{k^*} \leftarrow X_{k^*} \cup \{j\}$
15:         Update $c(k^*)$
16:     end for
17:     return $X$
18: end procedure
```

3.2 Local Search

To attempt to improve the solution, a local search phase that uses VND within an IGLS framework is performed (Steps 5-14 in Algorithm 1). In this phase a perturbation consisting of unassigning and reassigning customers to centers according to the IGLS idea is done first (Step 6). Then, this is followed by VND (Step 7), where two different neighborhood structures are used. Finally, a more aggressive destruction/reconstruction procedure is performed (Step 8). Each of these components is explained in detail next.

This idea of hybridizing ILS with VND/VNS has been successfully used in the past for other combinatorial optimization problems. For instance, Ribeiro et al. [16] implement a variant of this to solve a real-life car sequencing problem; Subramanian et al. [19] used a parallel version for the the VRP with simultaneous pickup and delivery; Martins et al. [13] proposed a method for solver a routing and wavelength assignment problem for optical networks. Another interesting advantage of IGLS is that it allows to diversify and improve along the search without the need to employ complex memory structures.

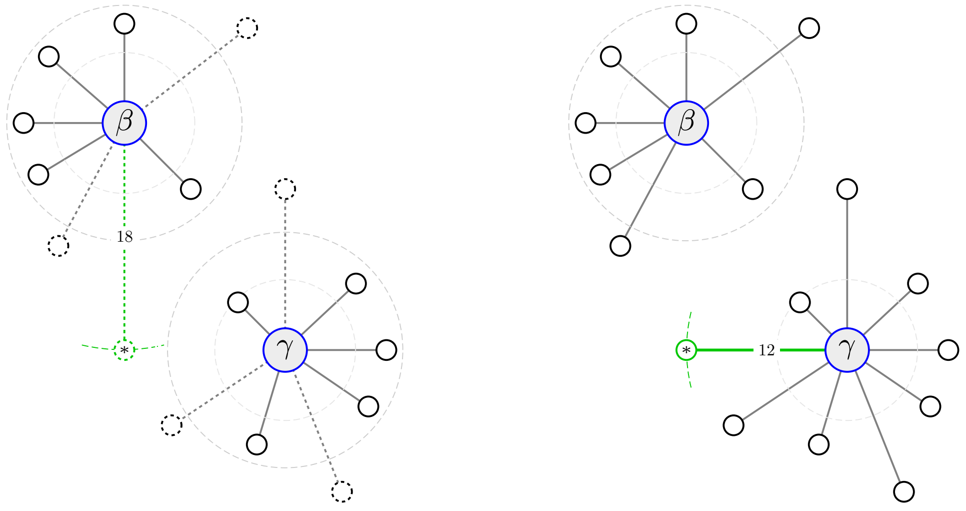
Algorithm 3 Perturbation

```
 1: procedure PERTURBATION($X, \alpha$)
 2:     $W \leftarrow \varnothing$                                        ▷ Destruction
 3:     for all $k \in K$ do
 4:         Update $\rho(j), j \in X_k \setminus \{c(k)\}$
 5:         while $\frac{|W|}{|X_k|-1} < \alpha$ do
 6:             Choose $j \in X_k \setminus \{c(k)\}$ randomly using a probability $\rho(j)$
 7:             $W \leftarrow W \cup \{j\}$
 8:         end while
 9:     end for
10:     Sort $W$ from worst to best $d_{j\ell(j)}, j \in W$
11:     $X \leftarrow X \setminus W$
12:     for all $j' \in W$ do                                  ▷ Reconstruction
13:         $k^* \leftarrow \arg\min_{k \in K} \phi(j', k)$
14:         $X_{k^*} \leftarrow X_{k^*} \cup \{j'\}$
15:     end for
16:     Update $c(k), k \in K$
17:     return $X$
18: end procedure
```

3.2.1 Perturbation

This method takes a solution as an input and applies destruction and reconstruction steps. The objective of this procedure is to reduce bottleneck elements using specific criteria of disconnection for each subset. In this specific case, for a chosen X_k, the destruction step is done by unassigning the $\alpha\%$ of nodes located in X_k, with high values of the probability function defined by $\rho(j) = d_{jc(k)} / \sum_{j' \in X_k} d_{jc(k)}$. The choice of this function is motivated by the fact that the nodes farther from the center are the ones affecting more the dispersion function, i.e., any node disconnected belonging to $\mathcal{J}(X)$. The reconstruction step reassigns each disconnected node to a nearest center, namely $X_{k^*} \leftarrow X_{k^*} \cup \{j\}$ where $k^* = \arg\min_{k\in K} \phi(j,k)$. A priority assignment is given to the bottleneck nodes, i.e., nodes whose previous assignment matched the value of the objective function value. Finally, once the reconstruction is done, the centers for the entire partition are updated using (2). The pseudo-code of this perturbation method is shown in Algorithm 3.

The destruction and reconstruction steps of the perturbation method are illustrated in Figure 1. In this example, β and γ represent the node centers of two subsets. The nodes in green indicate bottleneck nodes or nodes that generate the worst cost. In this case, before destruction $f(X) = 18$. In the destruction phase, the nodes with dash line represent the set of nodes selected to be disconnected from its center based on the probability $\rho(j)$. In the reconstruction phase, the green node is given priority allocation and changes its location from β to γ, reducing the cost of the current solution to 12.



(a) Destruction (b) Reconstruction

Figure 1: Destruction and reconstruction steps in the perturbation method.

It is important to point out that an improvement in the objective function value is not necessarily achieved by the perturbation method. This is due to the fact that there might be some nodes that can worsen the objective value of the current solution during the reconstruction step. However; this is not precisely bad news because, it allows more diversification to a given solution which can later be improved by VND in the following step. As it is usual in IGLS, the value of the destruction parameter α plays an important role in the quality of the solutions found by the method. Appropriate values are empirically investigated and fine-tuned for a large number of instances in the benchmark data sets as shown in Section 4.2.

3.2.2 VND

Our VND implementation, depicted in Algorithm 3, employs two different neighborhood structures denoted by $\mathcal{N}_1$ and $\mathcal{N}_2$. For each of the two neighborhoods, the potential move takes into account both distance and capacity factors. Given the nature of the objective function, it makes perfect sense to consider only a subset of promising moves, not the entire neighborhood. Therefore a best improving move strategy is adopted through the search. Each neighborhood is described next.

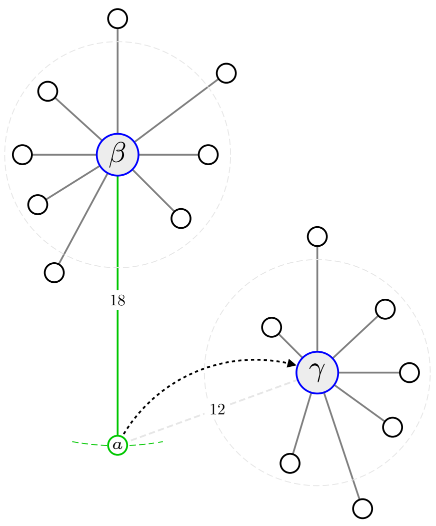
Algorithm 4 Variable Neighborhood Descent

```
1: procedure VND($X$)
2:     while $k \leq k_{\max}$ do
3:         $X' \leftarrow \arg\min_{y \in \mathcal{N}_k(X)} f(y)$
4:         if $X'$ is better that $X$ then
5:             $X' \leftarrow X$
6:             $k \leftarrow 1$
7:         else
8:             $k \leftarrow k + 1$
9:         end if
10:    end while
11:    return $X$
12: end procedure
```

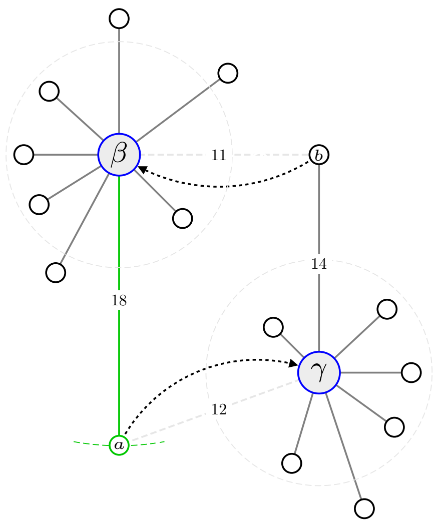
$\mathcal{N}_1$) *Reinsertion*: This neighborhood considers moves where a node $j \in \mathcal{J}(X)$ (currently assigned to center of set X_q) is assigned to set X_k, with $k \neq q$, i.e., given $X = (X_1, \ldots, X_p)$ $\mathit{reinsertion}(j,k) = \{X_1, \ldots, X_q \setminus \{j\}, \ldots, X_k \cup \{j\}, ..., X_p\}$ where j must be a bottleneck node for the move to be attractive. Note that a move is considered valid if and only if the move does not exceed the capacity of the target subset and the distance between j and $c(k)$ is strictly less than $f(X)$. These restrictions rule out infeasible movements, reducing the size of the neighborhood to be explored, and therefore, increasing the performance of the procedure.

$\mathcal{N}_2$) *Exchange*: This neighborhood considers moves where two nodes $i \in \mathcal{J}(X)$ and j in a different subset X_q, with $k \neq q$, are swaped, i.e., given $X = (X_1, \ldots, X_p)$, $\mathit{swap}(i,j) = \{X_1, ..., X_q \cup \{j\} \setminus \{i\}, ..., X_k \cup \{i\} \setminus \{j\}, ..., X_p\}$, where i must be a bottleneck node for the move to be attractive. Similarly, the move is valid if and only if it does not exceed the capacity of the respective target subset and the distance between i and $c(k)$, j and $c(q)$ is strictly less than $f(X)$.

These two moves are illustrated in Figure 2. A green node indicates a bottleneck node. Figure 2(a) represents a reinsertion move. Here, the node labeled "a" is designated as the most attractive move, and it changes its location from β to γ, reducing the cost of the current solution to 12. Figure 2(b) represents an exchange move, where nodes "a" and "b" are swapped, reducing the cost of the current solution to 12.



(a) Reinsertion



(b) Exchange

Figure 2: Illustration of neighborhood structures in the VND method.

3.2.3 Shake

The previously described perturbation and VND methods consider moves where centers tend not to change too much. The computational effort of every move evaluation is reasonable. However, it is possible to force a more aggressive destruction/reconstruction mechanism that would cause centers to change considerably. This is achieved by a "shake" mechanism that is applied after a local optima is reached under the iterative application of the perturbation and VND. This, of course, requires more computational effort but it pays off. The shake method performs removal and reconstruction of several subsets, which greatly diversifies the search path. The subsets to be destructed are chosen according to the following criterion:

$$\mathcal{K} \leftarrow \{\eta^1(j), \eta^2(j), \eta^3(j) : \eta^1(j) = l(j), j \in \mathcal{J}(X)\} \tag{6}$$

where $\eta(j) = \arg\min_{k \in K} d_{jc(k)}$. Here $\eta^1(j)$, $\eta^2(j)$, and $\eta^3(j)$ are the first, second, and third nearest centers to $j \in \mathcal{J}(X)$, respectively, under the distance criterion. Let $W \leftarrow \cup_{k \in \mathcal{K}} X_k$. We then now remove these sets from the current solution $X \leftarrow X \setminus W$. Now, using the construction method, we construct a new solution X' by reassigning the nodes in W with $p' = |\mathcal{K}|$. Finally $X \leftarrow X \cup X'$ is the new current solution.

Algorithm 5 Shake

```
1: procedure Shake($X$)
2:     $\mathcal{K} \leftarrow \{\eta^1(j), \eta^2(j), \eta^3(j) : \eta^1(j) = \ell(j), j \in \mathcal{J}(X)\}$
3:     if $\mathcal{L} \neq \varnothing$ then
4:         $W \leftarrow \varnothing$
5:         $W \leftarrow \bigcup_{k \in \mathcal{K}} X_k$
6:         $X \leftarrow X \setminus W$
7:         $p' \leftarrow |\mathcal{K}|$
8:         $X' \leftarrow$ Construction($W, p'$)
9:         $X \leftarrow X \cup X'$
10:    end if
11:    return $X$
12: end procedure
```

Note that this reconstruction step introduces a randomized mechanism, influenced by the construction phase. This shake mechanism does not guarantee reducing bottleneck nodes nor improving the objective value; however, the aggressive destruction diversifies the partial structure of the current solution, selecting other subsets of potential centers which may be desirable to guide the method to other most promising search regions during the improvement phase to be performed later.

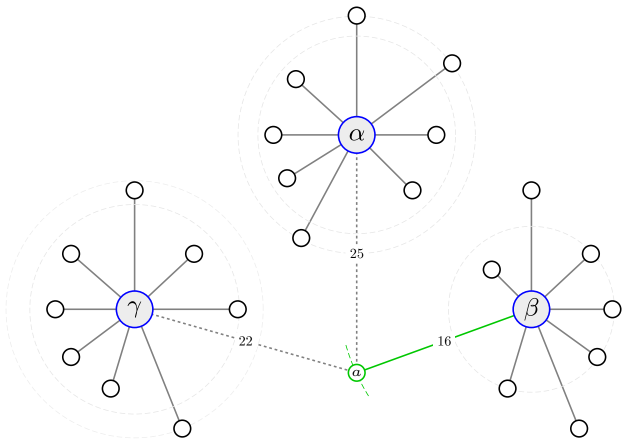


Figure 3: Segment of a solution which is eligible for applying the shake method.

Figure 3 shows a segment of a solution which is eligible for applying the shake method. In this example, node a is a bottleneck node currently assigned to its closest center β under the distance criterion, i.e., $\eta(a) = \ell(a)$. Therefore, under the selection criterion defined in (6), the region formed by the subsets with centers α, β, and γ is destroyed. Then these three sets are rebuilt using the construction step defined in Algorithm 2, yielding a new solution.

4 Computational Results

This section shows the overall performance of the heuristic which is empirically assessed on widely used benchmarks on location literature. We use the exact methods of Özsoy and Pınar (OP) [14] (OP) and Albareda-Sambola et al. (ADF) [1] to compute exact solutions or lower bounds to some problems to have a better estimates of the optimality gap for the heuristic solutions. ILOG CPLEX 12.5 is used as LP solver in these exact methods. A time limit of 1 hour and a memory usage limit of 2 Gb was set as stopping criteria. Each of the experiments was carried out on a machine with AMD Opteron 2.0 GHz (x16), 32 GiB RAM under Debian 6.0.8 GNU/Linux Kernel 2.6.32-5, 64 bit architecture.

4.1 Benchmark Instances

For the experiments, we used four different data sets. No benchmark instance data sets for the CpCP exist in the literature; however, we tested and compared all methods using data sets generated for other location problems and used in previous work on this problem.

(Set A) Beasley OR-Library: This data set, proposed in [2] for the capacitated p-median problem, contains two groups of 10 instances with equal facility capacity. One has 50 demand nodes and 5 facilities to be located, and the other has 100 demand nodes and 10 facilities to be located.

(Set B) Galvão and ReVelle: This data set was generated by Scaparra et al. [18] specifically for the CpCP based on the data set of Galvão and ReVelle for the maximal covering location problem [15]. The data sets contais instances with 100 and 150 customers, and 5 to 15 centers, with variable facility capacity. The original set is composed by two networks that were randomly.

(Set C) Lorena and Senne: This set, proposed in [12] por the CpMP, includes 6 large instances whose size ranges from 100 to 402 customers, and from 10 to 40 centers, and equal facility capacity. This is considered a large scale set given the number of nodes.

(Set D) Ceselli and Righini: There exists a recent data set added to the OR-Library proposed in [4]. This is regarded as a very hard set to solve for capacitated location problems such as p-median and p-center problems, and may be significantly influenced by the ratio between n and p. This set is composed by 4 subsets α, β, γ and δ, each subset consisting of forty instances: 20 of them concern graphs with 50 and 100 vertices and $p = \frac{n}{10}$; the other 20 instances were randomly generated on graphs with cardinality 150 (15 centers) and 200 (20 centers), also all capacities were fixed to 120. Therefore, the same 40 instances were solved with different number of centers: α with $p = \lfloor \frac{n}{10} \rfloor$, β with $p = \lfloor \frac{n}{4} \rfloor$, γ with $p = \lfloor \frac{n}{3} \rfloor$ and δ with $p = \lfloor \frac{2n}{5} \rfloor$. The overall capacity was preserved in all subsets by setting $w_j = \left\lceil 12\frac{n}{p} \right\rceil$. We refer to this as four different data sets, namely D-α, D-β, D-γ, and D-δ.

4.2 Fine-tuning

The purpose of this first experiment is to fine-tune the heuristic with respect to parameter α for each data set. This is achieved by a detailed statistical analysis described below. The response variable studied is the average relative percentage deviation or gap as follows:

$$\text{GAP} = \frac{f(X) - f^{lb}}{f^{lb}}, \tag{7}$$

where $f(X)$ is the resulting objective function value found for each particular instance and f^{lb} is the best known lower bound. This bound was obtained by applying either of the exact methods [14, 1]. In some cases this computed best known lower bound turned out to be the optimal solution. The heuristic iteration limit was set to 100 for each value $\alpha \in \{0.0, 0.1, \ldots, 1.0\}$ on all data sets, performing 5 replicates of this experiment. We separate the results into seven blocks, according to the number of data sets with subsets and study a single factor α of each block composed by 5 replicates, where GAP is the response variable. First, we analyzed the data using the Kruskal-Wallis non-parametric test [10]. For each data set we obtained p-value < 0.05; therefore, concluding that α affects the response variable.

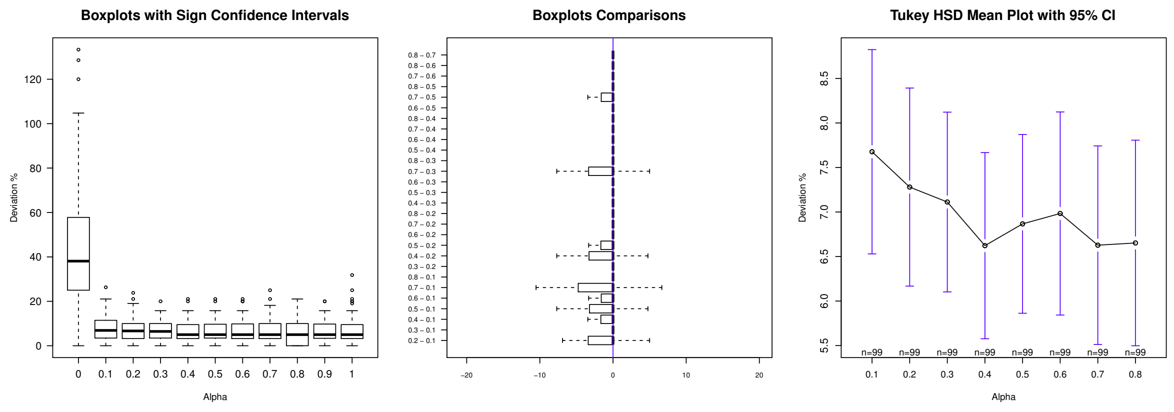


Figure 4: Graphs of statistical analysis for data set A.

Figures 4–10 show a set of three plots for each data set. The first plot shows a simple boxplot based on the relationship between the levels (α) and GAP. It can be seen clearly that α affects the response variable. When $\alpha = 0.0$, the GAP has inferior quality than it has in $0.1 \leq \alpha \leq 0.9$. For the for the rest of the plots, we only consider analyzing the range $[0.1, 0.8]$. It should be noted, that as α increases so does the level of destruction and therefore the computational effort increases. The second plot displays boxplots of the groups with their sign confidence intervals for the medians using a Friedman's Test [7]. This graph is interesting because one can visually see the locations of the groups with respect to the others, defining the significance by a gray color in the bar. Finally, the third graph is a means plot with Tukey's Honest Significant Difference (HSD) 95% confidence intervals. This plot gives us a more accurate view of what level (α) the mean achieves its lowest value against the response variable. Based in the last test, it was observed that the set $\{0.4, 0.5, 0.6, 0.6, 0.5, 0.4, 0.6\}$ of α values gave the best resuts for each data set A, B, C, D-α, D-β, D-γ and D-δ, respectively. For the next experiments, we used these selected values for α and run our heuristic using 1000 as iteration limit for data sets A, B, and C, and 500 as iteration limit for data sets D-*, and 30 repetitions.

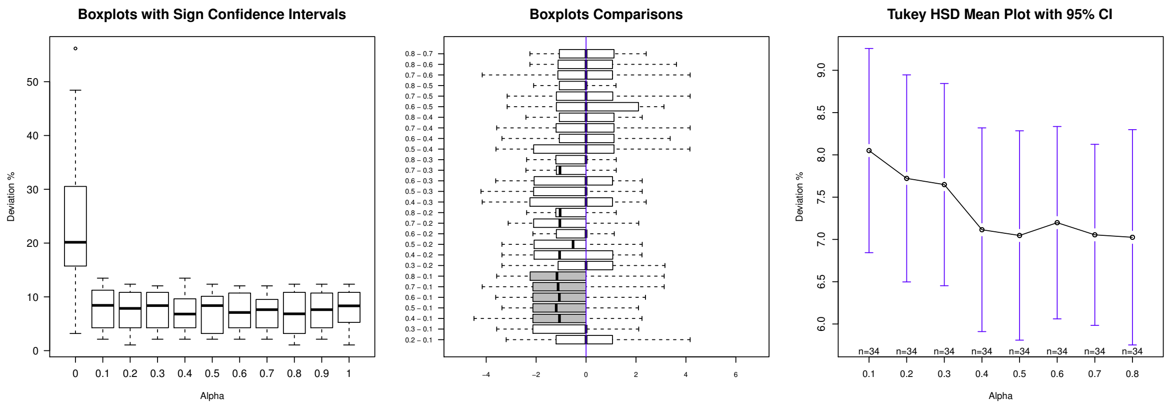


Figure 5: Graphs of statistical analysis for data set B.

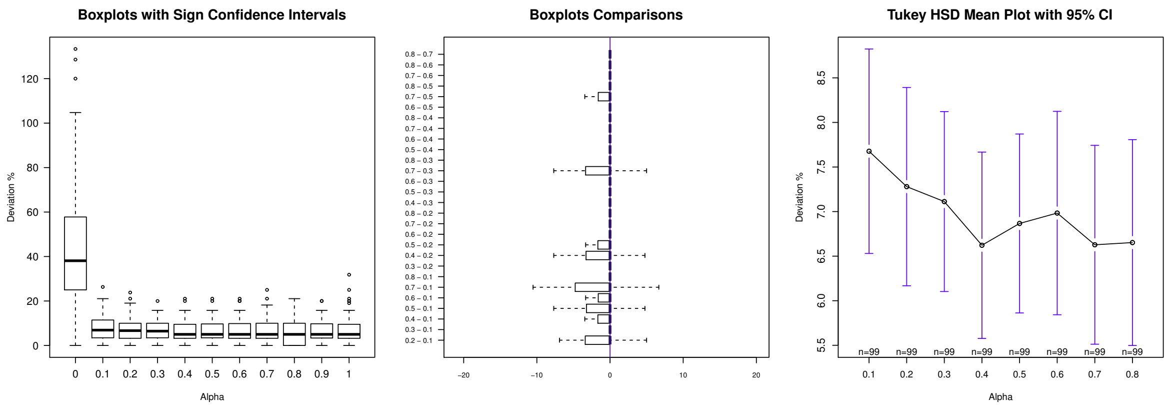


Figure 6: Graphs of statistical analysis for data set C.

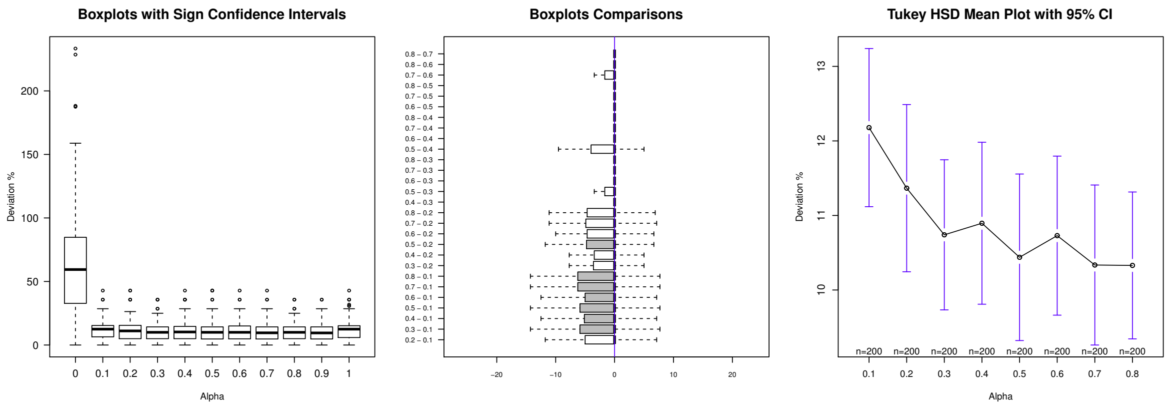


Figure 7: Graphs of statistical analysis for data set D-α.

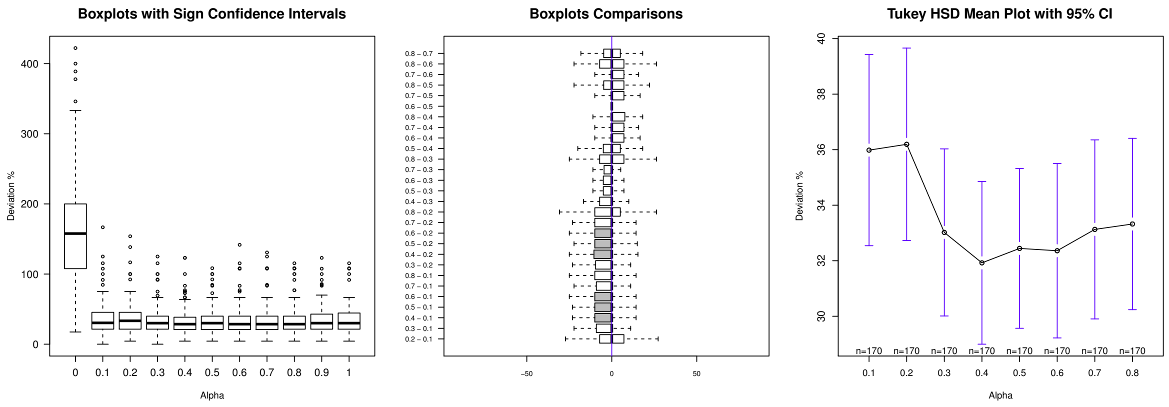


Figure 8: Graphs of statistical analysis for data set D-β.

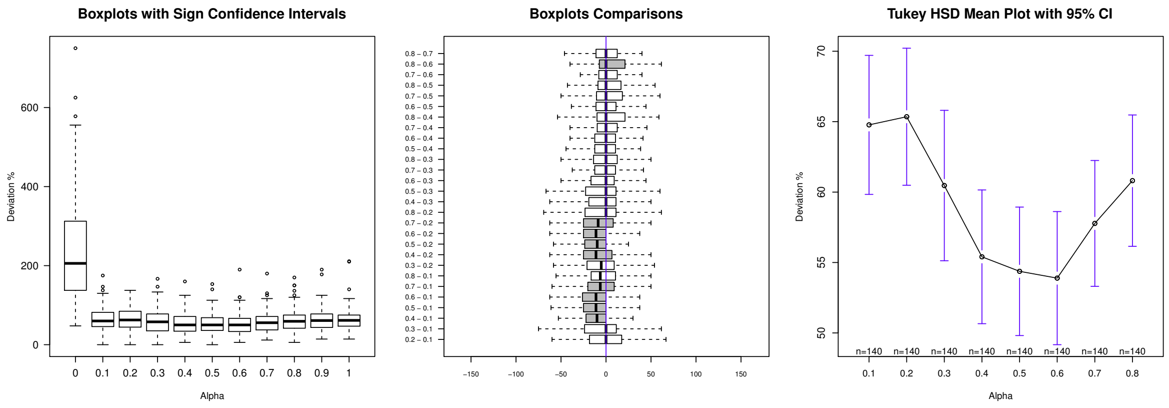


Figure 9: Graphs of statistical analysis for data set D-γ.

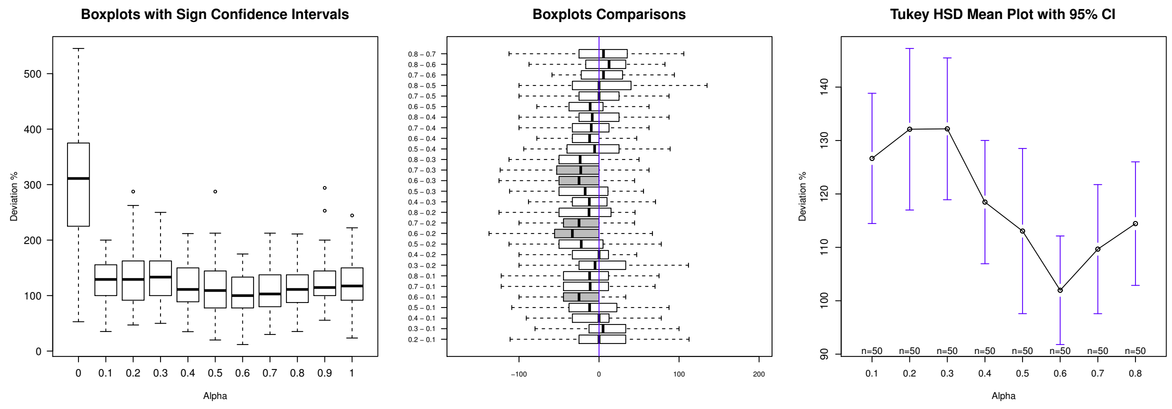


Figure 10: Graphs of statistical analysis for data set D-δ.

4.3 Comparison between heuristics

The main purpose of these experiments is to provide a detailed comparison between the proposed heuristic (QR) and the one by Scaparra et al. (SPS) [18]. In addition, we have decided to include the results obtained by the exact methods (Özsoy and Pınar (OP) [14] and Albareda-Sambola et al. (ADF) [1]) for two reasons. First, since we have the codes of all four methods, running them all in the same platform allows a true comparison among them in terms of computational effort. Most of the instances were solved by either of the exact methods, thus this allows for a true optimality gap computation for the heuristic solutions. Then, while it is expected that solution quality reported by the heuristic might not be as good as the one delivered by the exact methods, we can certainly assess the trade-off between solution quality and computational effort. Finally, to the best of our knowledge, this full comparison among these other exact and heuristic methods had not been done before.

Tables 1–7 display the comparison of methods for each data set. In each table the first two columns represent the instance size measured by number of nodes n and number of partitions p. "Instance" is the name of the particular problem instance and "Optimal" indicates the optimal value of the instance or the best known lower bound denoted by "*" beside the value. The section "Time (s)" gives the execution time in seconds and "Deviation (%)" expresses the percent of relative deviation or gap with respect to the optimal value or best known lower bound for each method. In the case of exact methods, the gap represent the deviation between best lower and upper bound found. It should be noted that for the proposed method QR, we show the average time performance over the 30 independent repetitions, also "QR^1 %" and "QR^2 %" denote the average and best GAP, respectively, over all repetitions. Table 8 summarizes the comparison among methods for all data sets in terms of their average relative optimality gap, running time, and number of infeasible solutions. An infeasible solution (indicated by "-" in the column) is reported if no feasible solution

Table 1: Comparison of methods on data set A.

n	p	Instance	Optimal	Time				Deviation %				
				ADF	OP	SPS	QR	ADF	OP	SPS	QR^1	QR^2
50	5	cpmp01	29	0.07	0.32	0.71	0.34	0.00	0.00	0.00	0.00	0.00
		cpmp02	33	0.13	2.05	0.97	0.34	0.00	0.00	0.00	0.00	0.00
		cpmp03	26	0.29	0.47	0.91	0.36	0.00	0.00	7.69	0.00	0.00
		cpmp04	32	0.09	1.00	1.01	0.36	0.00	0.00	0.00	0.00	0.00
		cpmp05	29	2.22	1.79	1.11	0.38	0.00	0.00	0.00	0.19	0.00
		cpmp06	31	2.19	2.72	1.06	0.38	0.00	0.00	3.23	3.07	0.00
		cpmp07	30	0.16	0.82	1.23	0.38	0.00	0.00	0.00	0.80	0.00
		cpmp08	31	0.18	1.10	1.09	0.37	0.00	0.00	0.00	1.29	0.00
		cpmp09	28	3.46	7.22	1.29	0.37	0.00	0.00	0.00	3.25	0.00
		cpmp10	32	4.41	6.39	2.50	0.37	0.00	0.00	9.38	12.81	0.00
		Average		1.32	2.39	1.19	0.36	0.00	0.00	2.03	2.14	0.00
100	10	cpmp11	19	6.38	5.48	9.34	1.17	0.00	0.00	5.26	9.62	0.00
		cpmp12	20	0.33	5.98	10.21	1.10	0.00	0.00	15.00	5.22	0.00
		cpmp13	20	11.27	6.54	9.32	1.15	0.00	0.00	10.00	0.64	0.00
		cpmp14	20	2.40	4.96	8.70	1.11	0.00	0.00	10.00	2.44	0.00
		cpmp15	21	2.66	7.28	9.64	1.19	0.00	0.00	9.52	3.73	0.00
		cpmp16	20	22.84	12.38	10.35	1.16	0.00	0.00	5.00	4.42	0.00
		cpmp17	22	78.51	553.31	9.91	1.20	0.00	0.00	4.55	4.70	4.55
		cpmp18	21	10.14	9.77	8.21	1.10	0.00	0.00	9.52	1.77	0.00
		cpmp19	21	5.00	19.06	10.16	1.18	0.00	0.00	9.52	5.66	0.00
		cpmp20	21	397.88	20.55	8.97	1.20	0.00	0.00	9.52	9.20	0.00
		Average		53.74	64.53	9.48	1.16	0.00	0.00	8.79	4.74	0.46
		Overall average		27.53	33.46	5.34	0.76	0.00	0.00	5.41	3.44	0.23

was found after the stopping criteria was met.

As far as data set A is concerned (Table 1), the exact method was found very efficient for the smaller instance group (size 50 × 5), performing better than any heuristic. However, when attempting the larger group (size 100 × 10), there are a couple of instances for which the exact method struggled. The performance of both heuristics was more robust than that of the exact method as they both took less than 0.65 seconds to solve each instance. In terms of solution quality, the proposed heuristic found better solutions than the ones reported by the SPS heuristic.

Table 2: Comparison of methods on data set B.

n	p	Instance	Optimal	Time				Deviation %				
				ADF	OP	SPS	QR	ADF	OP	SPS	QR^1	QR^2
100	5	G1	94	159.94	9.02	7.00	1.06	0.00	0.00	2.13	1.80	1.06
100	5	G2	94	53.37	12.41	7.09	1.06	0.00	0.00	2.13	1.83	0.00
100	10	G3	83	55.19	122.17	14.02	1.39	0.00	0.00	8.43	8.68	4.82
100	10	G4	84	113.05	41.76	15.21	1.44	0.00	0.00	7.14	8.60	5.95
150	10	G5	95	430.61	302.30	40.55	2.29	0.00	0.00	7.37	4.45	2.11
150	10	G6	96	155.52	224.46	36.85	2.20	0.00	0.00	7.29	4.23	2.08
150	15	G7	89	208.94	102.64	49.78	3.20	0.00	0.00	8.99	8.04	5.62
150	15	G8	89	251.59	421.37	47.88	3.21	0.00	0.00	10.11	8.79	6.74
		Overall average		178.53	154.52	27.30	1.98	0.00	0.00	6.70	5.80	3.55

When analyzing data set B (Table 2) we can observe that the exact method takes considerably longer than both heuristics to reach an optimal solution. On average, the exact method takes about an order of magnitude longer. In terms of heuristic solution quality, our heuristic obtains slightly better solutions (average gap of 6.04 %) than the SPS heuristic (average GAP of 6.70%).

Table 3: Comparison of methods on data set C.

n	p	Instance	Optimal	Time				Deviation %				
				ADF	OP	SPS	QR	ADF	OP	SPS	QR[1]	QR[2]
100	10	SJC1	364	262.10	368.97	14.59	0.71	0.00	0.00	26.67	25.52	5.28
200	15	SJC2	304	58.24	114.24	79.90	2.36	0.00	0.00	19.02	7.58	2.91
300	25	SJC3a	278	142.70	283.95	234.15	7.66	0.00	0.00	36.94	10.30	3.24
300	30	SJC3b	253	111.08	277.79	247.68	10.29	0.00	0.00	30.86	8.48	3.27
402	30	SJC4a	284	576.16	1100.08	631.14	13.82	0.00	0.00	42.29	9.06	4.51
402	40	SJC4b	239	251.51	343.50	505.12	22.68	0.00	0.00	34.93	9.15	3.60
		Overall average		233.63	414.76	285.43	9.59	0.00	0.00	31.79	11.68	3.80

Regarding data set C (Table 3), we can observe that the best exact method takes on average near 4 minutes while our heuristic takes under than 7 seconds. When comparing our heuristic with the SPS heuristic, we can see that ours is faster and finds solutions of significantly better quality.

The results of data sets D-* are displayed in Tables 4–5. As discussed in previous sections, this set is considered complex because their value of p has a significant effect on the performance of the methods. When analyzing subset D-α (Table 4), we observed out heuristic outperforms heuristic SPS in terms of both solution quality and computational effort. Regarding subset D-β (Table 5), SPS shows a significant number infeasible solutions while our heuristic provides feasible solutions for all instances. Finally, regarding subsets D-γ and D-δ (Tables 6–7) the infeasibility level of SPS is increased considerably, while QR provides a feasible solution for all instances of the subset D-γ and has only two instances with infeasible solutions in D-δ. The decrease in the quality of the solution by the heuristic methods is notable, but the computation effort is considerably less. In particular, our heuristic is considerably faster than any other method.

Table 8 summarizes the comparison among methods. Analyzing this table, we observe that QR is considerably faster that gets an SPS for all data sets. Regarding solution quality, the proposed method provides acceptable solutions for data sets A, B, C and for subsets D-α and D-β. For subsets D-γ and D-δ, the comparison between QR and SPS in terms of solution quality does not make too much sense because SPS fails in finding feasible solutions to a large number of instances. In that regards, our heuristic is still better as it was able to find feasible solutions to practically every instance. When compared to the exact methods, our heuristic is still more reliable in terms of number of feasible solutions found. Exact method ADF and OP delivered 36 and 13 infeasible solutions while QR delivered only 2 unfeasible solutions.

Figure 11 shows a comparison of the methods in terms of their asymptotic running time and used memory resources with respect to the number of nodes. The memory statistic indicates the maximum resident set size used [11], in bits, that is, the maximum number of bits of physical memory that each approach used simultaneously. As can be seen, the resources used by the proposed approach are lower than those used by the other three methods. In particular, the memory usage requirements of the two exact methods, as expected, are considerably larger.

Table 4: Comparison of methods on data set D-α.

n	p	Instance	Optimal	Time				Deviation %				
				ADF	OP	SPS	QR	ADF	OP	SPS	QR^1	QR^2
50	5	01	29	0.07	0.33	0.70	0.20	0.00	0.00	0.00	0.00	0.00
		02	33	0.12	2.04	0.98	0.18	0.00	0.00	0.00	0.00	0.00
		03	26	0.30	0.46	0.90	0.19	0.00	0.00	7.69	0.09	0.00
		04	32	0.09	1.01	1.01	0.21	0.00	0.00	0.00	0.00	0.00
		05	29	2.26	1.81	1.11	0.22	0.00	0.00	0.00	1.04	0.00
		06	31	2.17	2.72	1.05	0.20	0.00	0.00	3.23	3.18	0.00
		07	30	0.16	0.81	1.22	0.22	0.00	0.00	0.00	2.02	0.00
		08	31	0.19	1.07	1.08	0.20	0.00	0.00	0.00	2.49	0.00
		09	28	2.89	8.37	1.30	0.20	0.00	0.00	0.00	3.55	0.00
		10	32	4.27	6.88	2.51	0.21	0.00	0.00	9.38	14.44	0.00
		Average		1.25	2.55	1.19	0.20	0.00	0.00	2.03	2.68	0.00
100	10	11	19	6.31	5.52	9.18	0.63	0.00	0.00	5.26	11.73	0.00
		12	20	0.34	5.75	10.21	0.65	0.00	0.00	15.00	6.08	0.00
		13	20	11.12	6.65	9.67	0.65	0.00	0.00	10.00	1.75	0.00
		14	20	2.45	4.70	8.68	0.64	0.00	0.00	10.00	4.06	0.00
		15	21	2.68	7.28	9.60	0.63	0.00	0.00	9.52	4.63	0.00
		16	20	23.69	12.34	10.34	0.65	0.00	0.00	5.00	4.83	0.00
		17	22	77.74	554.23	9.78	0.65	0.00	0.00	4.55	5.86	4.55
		18	21	10.16	9.86	8.21	0.63	0.00	0.00	9.52	2.38	0.00
		19	21	4.94	19.84	10.20	0.64	0.00	0.00	9.52	6.85	0.00
		20	21	395.24	20.38	8.87	0.65	0.00	0.00	9.52	9.52	9.52
		Average		53.47	64.65	9.48	0.64	0.00	0.00	8.79	5.77	1.41
150	15	21	16	29.86	31.96	27.48	1.50	0.00	0.00	25.00	11.25	6.25
		22	17	103.79	1386.81	24.76	1.58	0.00	0.00	11.76	4.83	0.00
		23	16	33.12	37.78	23.35	1.54	0.00	0.00	18.75	12.50	0.00
		24	16	54.09	56.31	29.18	1.60	0.00	0.00	25.00	12.64	6.25
		25	16	0.72	32.71	22.99	1.55	0.00	0.00	12.50	0.87	0.00
		26	16	14.66	57.28	28.48	1.57	0.00	0.00	12.50	9.86	6.25
		27	18	3303.97	130.39	27.22	1.60	0.00	0.00	11.11	10.28	5.56
		28	17	21.43	48.93	28.06	1.57	0.00	0.00	5.88	0.00	0.00
		29	15	18.68	40.25	25.69	1.58	0.00	0.00	20.00	13.11	6.67
		30	15	4.58	52.53	28.87	1.59	0.00	0.00	20.00	7.08	6.67
		Average		358.49	187.50	26.61	1.57	0.00	0.00	16.25	8.24	3.77
200	20	31	14	45.00	29.83	52.55	3.01	0.00	0.00	21.43	6.31	0.00
		32	14	3600.75	1798.05	86.42	3.30	7.14	0.00	28.57	27.94	14.29
		33	14	58.70	101.90	57.16	3.14	0.00	0.00	28.57	11.07	7.14
		34	15	974.15	168.16	60.03	3.17	0.00	0.00	26.67	12.81	6.67
		35	14	94.71	83.85	66.88	3.07	0.00	0.00	14.29	10.91	7.14
		36	14	48.37	46.66	57.54	3.16	0.00	0.00	14.29	10.91	7.14
		37	14	8.03	89.52	65.99	3.08	0.00	0.00	28.57	7.54	0.00
		38	14	710.26	321.27	66.66	3.15	0.00	0.00	28.57	18.18	7.14
		39	13	66.72	215.80	56.29	3.07	0.00	0.00	23.08	14.87	7.69
		40	15	475.44	186.77	66.56	3.12	0.00	0.00	20.00	6.82	0.00
		Average		608.21	304.18	63.61	3.13	0.71	0.00	23.40	12.74	5.72
		Overall Average		255.36	139.72	25.22	1.38	0.18	0.00	12.62	7.36	2.72

Table 5: Comparison of methods on data set D-β.

n	p	Instance	Optimal	Time				Deviation %				
				ADF	OP	SPS	QR	ADF	OP	SPS	QR^1	QR^2
50	5	01	29	1.20	32.27	1.18	0.36	0.00	0.00	0.00	5.20	0.00
		02	33	0.93	3.07	2.19	0.38	0.00	0.00	10.00	13.75	10.00
		03	26	0.85	39.52	1.96	0.39	0.00	0.00	10.00	12.97	10.00
		04	32	7.92	5.35	2.51	0.39	0.00	0.00	8.33	20.78	16.67
		05	29	46.47	15.40	3.28	0.42	0.00	0.00	9.52	21.48	14.29
		06	31	8.60	5.11	3.73	0.40	0.00	0.00	4.55	16.74	0.00
		07	30	2.32	2.10	4.99	0.42	0.00	0.00	0.00	10.39	0.00
		08	31	2.28	2.63	3.78	0.44	0.00	0.00	4.35	4.35	4.35
		09	28	7.60	14.65	0.45	0.40	0.00	0.00	-	23.10	21.05
		10	32	2.55	3.87	0.15	0.38	0.00	0.00	-	21.43	4.35
		Average		8.07	12.40	2.42	0.40	0.00	0.00	5.84	15.02	8.07
100	10	11	19	209.89	435.85	8.72	2.07	0.00	0.00	0.00	15.16	0.00
		12	20	2.67	28.05	9.67	2.17	0.00	0.00	15.38	11.28	7.69
		13	20	1723.90	151.62	9.13	2.02	0.00	0.00	23.08	19.96	0.00
		14	20	3600.00	35.88	10.12	2.06	15.38	0.00	7.14	7.26	7.14
		15	21	14.01	13.76	10.07	2.08	0.00	0.00	21.43	20.80	14.29
		16	20	40.18	13.41	11.56	2.13	0.00	0.00	7.14	13.34	7.14
		17	22	37.76	59.74	12.55	2.13	0.00	0.00	21.43	26.98	21.43
		18	21	18.74	43.47	10.68	2.09	0.00	0.00	14.29	22.42	0.00
		19	21	17.21	31.33	14.01	2.14	0.00	0.00	30.77	38.37	30.77
		20	21	3078.14	3600.00	0.58	2.04	23.08	7.69	-	81.80	38.46
		Average		874.25	441.31	9.71	2.09	3.85	0.77	15.63	25.74	12.69
150	15	21	16	363.19	79.51	36.69	5.63	0.00	0.00	27.27	27.07	18.18
		22	17	171.87	103.24	26.91	5.72	0.00	0.00	18.18	34.95	18.18
		23	16	763.98	3600.01	28.26	5.71	9.09	10.00	27.27	45.86	27.27
		24	16	504.81	166.45	26.09	5.74	0.00	0.00	18.18	17.73	9.09
		25	16	9.10	56.96	15.78	5.39	0.00	0.00	20.00	16.56	10.00
		26	16	12.44	88.53	33.45	5.44	0.00	0.00	30.00	22.83	10.00
		27	18	164.87	111.24	213.91	5.54	0.00	0.00	25.00	68.84	33.33
		28	17	111.83	67.75	31.11	5.60	0.00	0.00	18.18	28.13	9.09
		29	15	6.33	98.90	28.81	5.42	0.00	0.00	20.00	22.11	10.00
		30	15	32.17	116.24	28.23	5.29	0.00	0.00	30.00	26.00	10.00
		Average		214.06	448.88	46.92	5.55	0.91	1.00	23.41	31.01	15.51
200	20	31	14	872.93	194.76	48.09	16.28	0.00	0.00	22.22	24.07	22.22
		32	14	2152.14	3600.03	9.27	16.67	44.44	11.11	-	287.13	144.44
		33	14	3473.79	3059.78	71.79	17.27	22.22	12.50	44.44	44.44	44.44
		34	15	714.62	1489.29	25.50	16.46	20.00	11.11	-	105.89	60.00
		35	14	1645.72	118.15	42.46	17.58	22.22	0.00	20.00	20.28	10.00
		36	14	1282.98	251.35	60.29	17.10	0.00	0.00	33.33	33.76	22.22
		37	14	66.58	183.14	45.71	16.79	0.00	0.00	22.22	19.81	11.11
		38	14	1436.34	507.41	63.14	17.76	11.11	0.00	20.00	25.39	10.00
		39	13	339.13	165.87	46.96	16.27	0.00	0.00	22.22	16.97	11.11
		40	15	2679.24	1331.90	55.01	17.90	22.22	12.50	33.33	54.70	33.33
		Average		1466.34	1090.17	46.82	17.01	14.22	4.72	27.22	63.24	36.89
		Overall Average		640.68	498.19	26.47	6.26	4.74	1.62	18.03	33.75	18.29

Table 6: Comparison of methods on data set D-γ.

n	p	Instance	Optimal	Time				Deviation %				
				ADF	OP	SPS	QR	ADF	OP	SPS	QR[1]	QR[2]
50	16	01	17	0.12	37.06	1.23	0.79	0.00	0.00	0.00	10.78	0.00
		02	18	15.81	23.62	2.67	0.76	0.00	0.00	5.56	12.13	5.56
		03	15	2.13	18.80	1.99	0.72	0.00	0.00	0.00	66.85	0.00
		04	21	3.92	3.28	3.40	0.73	0.00	0.00	4.76	14.29	14.29
		05	18	0.08	22.87	3.07	0.76	0.00	0.00	0.00	32.96	16.67
		06	19	34.18	2.29	6.69	0.73	0.00	0.00	5.26	34.56	15.79
		07	18	11.46	3.02	7.20	0.75	0.00	0.00	5.56	22.37	5.56
		08	20	10.39	71.76	0.53	0.77	0.00	0.00	-	28.17	20.00
		09	17	5.06	26.33	0.62	0.72	0.00	0.00	-	33.43	11.76
		10	23	6.86	171.56	0.19	0.74	0.00	0.00	-	71.52	13.04
		Average		9.00	38.06	2.76	0.75	0.00	0.00	3.02	32.71	10.27
100	33	11	11	7.85	32.90	12.61	4.11	0.00	0.00	27.27	30.65	9.09
		12	12	37.11	31.91	12.73	4.20	0.00	0.00	8.33	29.77	8.33
		13	12	903.74	59.72	9.68	4.06	0.00	0.00	8.33	29.26	0.00
		14	12	1053.49	48.20	16.82	4.28	0.00	0.00	16.67	35.42	16.67
		15	13	49.23	55.04	11.29	4.24	0.00	0.00	7.69	22.82	7.69
		16	13	1864.90	79.53	17.15	4.15	8.33	0.00	15.38	38.12	15.38
		17	13	408.38	58.80	24.23	4.02	0.00	0.00	7.69	12.69	0.00
		18	14	3600.01	170.91	18.27	4.21	7.69	0.00	7.14	22.38	7.14
		19	*11	3600.00	3600.00	28.15	4.35	9.09	10.00	18.18	65.15	18.18
		20	12	96.22	102.24	4.76	4.20	0.00	0.00	-	95.79	41.67
		Average		1162.09	423.93	15.57	4.18	2.51	1.00	12.96	38.20	12.42
150	50	21	11	3600.00	1592.62	58.20	16.64	20.00	0.00	18.18	52.43	18.18
		22	10	1077.10	159.58	59.59	16.42	0.00	0.00	30.00	120.00	70.00
		23	11	33.99	208.82	32.27	16.69	0.00	0.00	-	114.90	63.64
		24	10	136.11	168.12	44.21	17.71	0.00	0.00	20.00	41.22	30.00
		25	9	98.25	99.75	24.29	16.49	0.00	0.00	11.11	40.31	11.11
		26	9	15.88	107.44	26.44	16.86	0.00	0.00	22.22	54.14	33.33
		27	*11	617.72	3600.00	22.10	17.00	9.09	10.00	-	159.90	81.82
		28	10	3600.00	122.35	28.33	16.74	11.11	0.00	10.00	32.61	10.00
		29	9	6.92	140.94	30.29	17.36	0.00	0.00	22.22	47.22	33.33
		30	9	226.47	194.87	20.30	16.74	0.00	0.00	22.22	48.58	22.22
		Average		941.24	639.45	34.60	16.86	4.02	1.00	19.49	71.13	37.36
200	66	31	8	19.26	95.66	52.80	36.61	0.00	0.00	25.00	40.49	12.50
		32	9	3600.01	1498.20	9.73	36.02	22.22	0.00	-	393.83	244.44
		33	8	3600.00	1531.59	67.81	38.38	12.50	0.00	37.50	64.65	50.00
		34	10	876.98	769.40	67.84	36.22	33.33	0.00	-	185.17	110.00
		35	8	717.64	182.56	65.16	37.35	0.00	0.00	37.50	54.38	37.50
		36	8	3600.01	283.53	40.23	36.59	12.50	0.00	37.50	62.29	37.50
		37	8	21.10	313.41	58.08	37.72	0.00	0.00	25.00	56.32	25.00
		38	*8	3600.00	3600.00	73.45	36.63	12.50	14.29	50.00	70.42	50.00
		39	8	27.09	259.93	46.90	34.97	0.00	0.00	25.00	27.92	25.00
		40	9	489.26	302.57	70.91	37.37	12.50	0.00	22.22	54.26	33.33
		Average		1655.14	883.69	55.29	36.79	10.56	1.43	32.47	100.97	62.53
		Overall Average		941.87	496.28	27.05	14.65	4.27	0.86	16.99	60.75	30.64

Table 7: Comparison of methods on data set D-δ.

n	p	Instance	Optimal	Time				Deviation %				
				ADF	OP	SPS	QR	ADF	OP	SPS	QR^1	QR^2
50	20	01	17	0.23	107.37	1.78	1.02	0.00	0.00	0.00	69.09	5.88
		02	18	0.70	78.12	4.24	1.02	0.00	0.00	5.56	59.93	16.67
		03	17	2.38	87.34	6.29	0.88	0.00	0.00	0.00	130.85	41.18
		04	20	29.45	102.04	4.71	0.90	0.00	0.00	10.00	49.42	20.00
		05	19	21.26	63.37	0.36	0.92	0.00	0.00	-	88.76	42.11
		06	21	4.12	18.01	0.22	0.94	0.00	0.00	-	60.93	40.00
		07	19	4.81	17.83	0.23	0.93	0.00	0.00	-	70.10	10.53
		08	23	3600.00	159.20	0.23	0.91	20.00	0.00	-	64.06	8.70
		09	21	21.21	68.26	0.22	0.94	0.00	0.00	-	92.28	42.86
		10	23	14.20	52.95	0.24	0.99	0.00	0.00	-	159.78	86.96
		Average		369.84	75.45	1.85	0.95	2.00	0.00	3.89	84.52	31.49
100	40	11	12	52.88	125.81	37.62	5.96	0.00	0.00	16.67	76.48	41.67
		12	11	3600.01	89.80	25.44	6.73	20.00	0.00	18.18	74.30	45.45
		13	12	284.85	138.65	22.88	5.35	0.00	0.00	0.00	73.80	41.67
		14	12	12.69	90.96	70.71	6.37	0.00	0.00	-	121.62	58.33
		15	13	4.25	145.28	5.78	6.29	0.00	0.00	-	132.61	69.23
		16	13	272.28	143.99	103.63	5.70	0.00	0.00	-	66.41	30.77
		17	14	3600.01	154.08	5.49	5.14	7.69	0.00	-	134.98	64.29
		18	14	3600.01	209.06	3.39	6.65	7.69	0.00	-	111.15	50.00
		19	12	3600.00	289.10	29.40	5.46	9.09	0.00	-	113.64	58.33
		20	14	73.41	509.37	3.93	6.34	0.00	0.00	-	278.57	207.14
		Average		1510.04	189.61	30.83	6.00	4.45	0.00	11.62	118.36	66.69
150	60	21	11	3600.00	290.33	242.81	21.61	20.00	0.00	18.18	119.80	63.64
		22	12	3600.00	2930.40	10.23	21.24	9.09	0.00	-	275.37	116.67
		23	11	270.09	231.48	5.87	21.55	0.00	0.00	-	273.67	127.27
		24	9	3002.52	180.47	81.27	22.74	0.00	0.00	33.33	120.43	66.67
		25	9	5.21	142.60	24.77	21.71	0.00	0.00	11.11	73.89	22.22
		26	9	43.51	194.92	36.84	23.77	0.00	0.00	22.22	86.36	44.44
		27	13	3600.04	3600.00	7.74	21.80	7.69	11.11	-	238.46	238.46
		28	10	3600.00	1615.84	35.98	22.77	11.11	0.00	20.00	72.17	30.00
		29	9	8.04	232.82	38.47	24.83	0.00	0.00	22.22	91.05	44.44
		30	9	14.84	240.06	31.32	21.60	0.00	0.00	11.11	77.22	33.33
		Average		1774.43	965.89	51.53	22.36	4.79	1.11	19.74	142.84	78.71
200	80	31	8	26.43	233.07	55.32	49.17	0.00	0.00	37.50	75.00	37.50
		32	10	3600.03	1277.43	11.98	-	40.00	37.50	-	-	-
		33	8	84.32	402.58	107.37	49.81	0.00	0.00	37.50	112.57	62.50
		34	12	1577.88	421.93	11.98	-	40.00	0.00	-	-	-
		35	8	3600.00	1318.10	125.24	50.53	12.50	14.29	50.00	124.79	62.50
		36	9	57.93	171.37	403.80	49.85	0.00	0.00	-	135.37	66.67
		37	8	32.87	334.25	72.87	53.18	0.00	0.00	25.00	109.10	75.00
		38	8	3600.00	433.88	18.90	48.00	12.50	0.00	-	245.88	100.00
		39	8	85.02	302.22	42.66	50.10	0.00	0.00	25.00	80.90	50.00
		40	8	3600.01	927.86	94.24	49.45	12.50	14.29	-	186.80	87.50
		Average		1626.45	582.27	94.44	50.01	11.75	6.61	35.00	133.80	67.71
		Overall Average		1320.19	453.30	44.66	19.83	5.75	1.93	17.56	119.88	61.15

Table 8: Summary of comparison among methods on all data sets.

Data set	Average relative gap (%)				Average time (s)				Infeasible solutions			
	ADF	OP	SPS	QR^2	ADF	OP	SPS	QR	ADF	OP	SPS	QR
A	0.00	0.00	5.41	0.23	27.53	33.46	5.34	0.76	0	0	0	0
B	0.00	0.00	6.70	3.55	178.53	154.52	27.30	1.98	0	0	0	0
C	0.00	0.00	31.79	3.80	233.63	414.76	285.43	9.59	0	0	0	0
D-α	0.18	0.00	12.62	2.72	255.36	139.72	25.22	1.38	1	0	0	0
D-β	4.74	1.62	18.03	18.29	640.68	498.19	26.47	6.26	9	6	5	0
D-γ	4.27	0.86	16.99	30.64	941.87	496.28	27.05	14.65	12	3	8	0
D-δ	5.75	1.93	17.56	61.15	1320.19	453.30	44.66	19.83	14	4	21	2

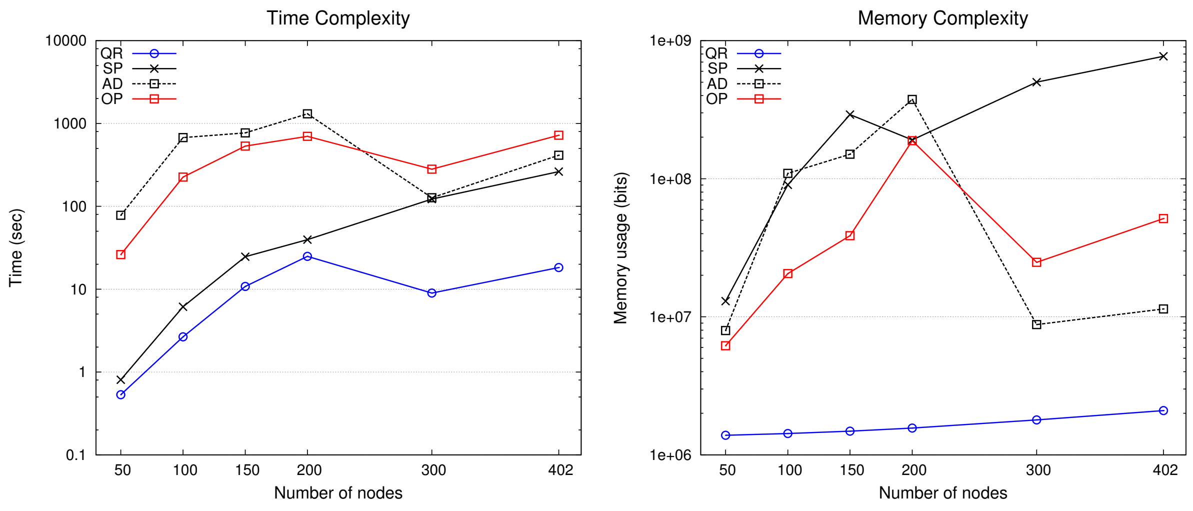


Figure 11: Comparison among methods in terms of asymptotic running time and memory usage. The vertical axis is on a logarithmic scale.

4.4 Component analysis

In this last experiment, we assess the value that each individual component gives to the QR heuristic. We consider the three essential components of the method: Perturbation, VND and Shake. The experiment consists of disabling one component at a time and running the heuristic using 500 as iteration limit, 30 repetitions, and the same set of α values of the previous experiment.

Table 9: Component analysis within heuristic QR.

Data set	All	Components (Average GAP (%))			Contribution (%) to obj. fn.		
		Perturbation	VND	Shake	Perturbation	VND	Shake
A	0.23	12.21	3.15	1.48	74.14	18.12	7.74
B	3.55	10.14	4.66	5.51	71.27	9.59	19.13
C	3.80	26.58	9.47	14.61	58.25	14.28	27.47
D-α	2.72	22.33	7.12	5.11	78.55	14.93	6.52
D-β	18.29	73.50	45.97	23.95	62.34	31.26	6.40
D-γ	30.64	123.92	73.96	43.75	60.85	29.16	10.00
D-δ	61.15	198.11	122.18	69.30	67.26	29.51	3.23

Table 9 displays the comparison of the components for each data set. In this table, the column "All" represents the GAP when all components are enabled, which matches the value displayed in Table 8, column QR^2. Each column in the section "Components" represent the component disabled during the experiment and shows the value obtained. The section "Contribution %" displays the percentage value that the specific component provides with respect to total value shown in "All." It is remarkable that the most influential component within the heuristic is the Perturbation, followed by VND and Shake. This is consistent with the statistical analysis, which showed that the parameter α, used in the perturbation, influences the response variable. Nevertheless, all other two components add value to the overall performance. The benefit of VND ranges from 9.59 to 31.26 %, and the benefit of the Shake methid ranges from 3.23 to 27.47 %.

5 Conclusions

We have proposed a metaheuristic framework that integrates several components such as a greedy randomized adaptive procedure with probabilistic selection in its construction phase and iterated greedy with a variable neighborhood descent in its local search phase. The results indicate the proposed heuristic outperforms the best heuristic in terms of both solution quality and running time. The performance of the proposed approach is more robust than that of the exact methods, requiring less computational effort and memory for obtaining solutions reasonably good objective values for data sets A, B, and C. For the harder instances in data sets D-γ and D-δ, the optimality gaps of the heuristic solutions are not as good; however, they are still obtained very quickly. For this harder set, our heuristic found feasible solutions to almost all instances tested, which is clearly superior to the SPS heuristic as it failed in several instances. In a detailed component analysis, we have seen the success of the heuristic is mainly due to the perturbation and VND methods. However, for data sets A, B, and C, the shake method proved very worthwhile as well. The proposed method provides robust solutions in a short time to the problems previously discussed in the literature, for the test set D introduced in this paper, the method ensures greater feasibility and speed compared to the existing heuristic.

Acknowledgements: This work was supported by the Mexican National Council for Science and Technology (grant CONACYT CB-2011-01-166397) and Universidad Autónoma de Nuevo León (grant UANL-PAICYT CE728-11). We also thank Maria Scaparra and Juan A. Díaz for providing us with the source code of their methods.

References

[1] M. Albareda-Sambola, J. A. Díaz, and E. Fernández. Lagrangean duals and exact solution to the capacitated p-center problem. *European Journal of Operational Research*, 201(1):71–81, 2010.

[2] J. E. Beasley. OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

[3] B. Ç. Tansel. Discrete center problems. In H. A. Eiselt and V. Marianov, editors, *Foundations of Location Analysis*, volume 155 of *International Serios in Operations Research & Management Science*, chapter 5, pages 79–106. Springer, New York, 2011.

[4] A. Ceselli and G. Righini. A branch-and-price algorithm for the capacitated p-median problem. *Networks*, 45(3):125–142, May 2005.

[5] M. Dyer and A. Frieze. A simple heuristic for the p-center problem. *Operations Research Letters*, 3(6):285–288, February 1985.

[6] S. Elloumi, M. Labbé, and Y. Pochet. A new formulation and resolution method for the p-center problem. *INFORMS Journal on Computing*, 16(1):84–94, 2004.

[7] M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[8] P. Hansen and N. Mladenović. An introduction to variable neighborhood search. In S. Voss, S. Martello, I. H. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 433–458. Kluwer, Boston, 1999.

[9] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems, Part I: The p-centers. *SIAM Journal on Applied Mathematics*, 37:513–538, 1979.

[10] W. H. Kruskal and W. A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[11] S. Loosemore, R. M. Stallman, R. McGrath, A. Oram, and U. Drepper. *The GNU C Library Reference Manual: for version 2.17.* Free Software Foundation, 2012.

[12] L. A. N. Lorena and E. L. F. Senne. A column generation approach to capacitated p-median problems. *Computers & Operations Research*, 31(6):863–876, 2004.

[13] A. X. Martins, C. Duhamel, M. C. Souza, R. R. Saldanha, and P. Mahey. A VND-ILS heuristic to solve the RWA problem. In Julia Pahl, Torsten Reiners, and Stefan Voß, editors, *Network Optimization*, volume 6701 of *Lecture Notes in Computer Science*, pages 577–582. Springer Berlin, 2011.

[14] F. A. Özsoy and M. Ç. Pınar. An exact algorithm for the capacitated vertex p-center problem. *Computers & Operations Research*, 33(5):1420–1436, 2006.

[15] C. S. ReVelle and H. A. Eiselt. Location analysis: A synthesis and survey. *European Journal of Operational Research*, 165(1):1–19, August 2005.

[16] C. C. Ribeiro, D. Aloise, T. F. Noronha, C. Rocha, and S. Urrutia. An efficient implementation of a VNS/ILS heuristic for a real-life car sequencing problem. *European Journal of Operational Research*, 191(3):596–611, 2008.

[17] R. Ruiz and T. Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033 – 2049, 2007.

[18] M. P. Scaparra, S. Pallottino, and M. G. Scutellà. Large-scale local search heuristics for the capacitated vertex p-center problem. *Networks*, 43(4):241–255, 2004.

[19] A. Subramanian, L. M. A. Drummond, C. Bentes, L. S. Ochi, and R. Farias. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 37(11):1899–1911, 2010.