Commercial Territory Design for a Distribution Firm with New Constructive and Destructive Heuristics

Jaime Cano-Belmán
Graduate Program in Systems Engineering
Universidad Autónoma de Nuevo León
AP 111 – F, Cd. Universitaria
San Nicolás de los Garza, NL 66450, Mexico
E-mail: *jaime@yalma.fime.uanl.mx*

Roger Z. Ríos-Mercado[1]
Graduate Program in Systems Engineering
Universidad Autónoma de Nuevo León
AP 111 – F, Cd. Universitaria
San Nicolás de los Garza, NL 66450, Mexico
E-mail: *roger.rios@uanl.edu.mx*

M. Angélica Salazar-Aguilar
HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal, Canada H3T 2A7
E-mail: *angelica.salazar@cirrelt.ca*

15 February 2010
Revised: 26 October 2011
Accepted: 26 January 2012

[1]Corresponding author

Abstract

A commercial territory design problem with compactness maximization criterion subject to territory balancing and connectivity is addressed. Four new heuristics based on Greedy Randomized Adaptive Search Procedures within a location-allocation scheme for this NP-hard combinatorial optimization problem are proposed. The first three (named GRLH1, GRLH2, and GRDL) build the territories simultaneously. Their construction phase consists of two parts: a location phase where p territory seeds are identified, and an allocation phase where the remaining basic units are iteratively assigned to a territory. In contrast, the other heuristic (named SLA) builds the territories one at a time. Empirical results reveals that GRLH1 and GRLH2 find near-optimal or optimal solutions to relatively small instances, where exact solutions could be found. The proposed procedures are relatively fast. We carried out a comparison between the proposed heuristic procedures and the existing method in larger instances. It was observed the proposed heuristic GRLH1 produced very good competitive results with respect to the existing approach. *Keywords:*

Combinatorial optimization; territory design, location-allocation heuristics; GRASP

1 Introduction

Territory design consists of grouping small geographic areas or *basic units* (BUs) into larger geographic clusters called *territories* or *zones* in such a way that the latter are acceptable according to relevant criteria. These criteria can be economically motivated or have a demographic background. Furthermore, spatial constraints like compactness and contiguity are often required.

The problem has applications in many fields such as political districting (Hess et al. [10]), districting for schools and social facilities (Ferland and Guénette [8]), sales territory design (Zoltners and Sinha [15]), territory design for winter service and solid waste collection (Muyldermans et al. [12]), territory design for emergency services (Bertolazzi, Bianco, and Ricciardelli [1]), delivery zones for distribution centers (Haugland, Ho, and Laporte [9]), or sales force deployment (Drexl and Haase [3]). Excellent and complete surveys on territory design problems can be found in Kalcsics, Nickel, and Schröder [11], Duque, Ramos, and Suriñach [4], and Zoltners and Sinha [15]).

Firms with sales forces normally require to divide the market into responsibility areas. Thus, territory design must be done in order to obtain service areas or to locate technical facilities. Several criteria such as organizational criteria (number of territories, number of BUs, exclusive assignment, location of sales representatives), geographical criteria (contiguity, accessibility, compactness), and activity related criteria (balancing, maximizing profit) are often used in territory design.

We address the version of the territory design problem with the following features. Each BU has associated two attributes or activities: number of customers and sales volume. Each BU must be assigned to only one territory. The size of each territory must be balanced with respect to each activity. Given that it is not easy to find a perfectly balanced solution, the balancing requirement is handled by a user-specified tolerance that allows a relative deviation from the target territory size. The requirement of contiguity is also taken into account, which means that, each pair of BUs belonging to the same territory must be joined by a path contained completely in that territory. In order to get compact territories a dispersion measure based on the objective function of the well-known p-center problem is minimized.

This problem was introduced by Ríos-Mercado and Fernández [13]. In their work (RF for short), they consider three activity measures (number of customers, sales volume, and workload). They propose a GRASP approach which incorporates reactivity and filtering. During the construction phase the territories are created one by one, in such a way that a territory is started with a BU and iteratively the assignment of BUs takes place. When the territory size reaches its upper bound limit it is closed and a new territory is started. The creation of territories in this manner does not necessarily produce the number of territories required, thus an adjustment phase is then carried out. Finally, an improvement phase (post-processing) is performed. They consider a neighborhood that consists of moving a specific BU from its current territory to an adjacent territory. They used a weighted merit function with two components, the dispersion measure and violation of the

balancing constraints. An important limitation of this approach is the high level of infeasibility with respect to the balancing constraints reached in the construction phase. A direct consequence of this is that the local search spends a tremendous amount of effort trying to reach feasibility.

In this work, we propose four different location-allocation heuristics. Three of them are constructive in nature and the other is destructive. Three of our proposed heuristics (called GRLH1, GRLH2, and GRDL) seek that all territories are growin uniformly. Their location-allocation scheme consist of two phases: a location phase whose role is to locate p BUs that would serve as initial seeds for the territory creation, and an allocation phase where the remaining BUs are assigned to these territories. Thus, it is expected that this simultaneous creation of p territories yields lower levels of infeasibility with respect to the balancing constraints than those obtained in previous work [13], where the territories are built one by one. A fourth heuristic called SLA builds the territories one at a time, in this heuristic an active territory is "closed" when its size reaches the upper limit allowed for any activity (number of customers or sales volume).

A detailed design of experiments following the guidelines of Coy et al. [2] was first carried out for parameter fine-tunning. Afterwards, the procedures were evaluated over a data set of randomly generated instances. Empirical results reveals that GRLH1 and GRLH2 find near optimal or optimal solutions to relatively small instances, where exact solutions could be found. The proposed procedures are relatively fast. We carried out a comparison between the proposed heuristic procedures and the existing method in larger instances. It was observed that GRLH1 produced very good competitive results with respect to the existing approach.

This paper is organized as follows. In Section 2 a detailed description and formulation of the problem is given. In Section 3, the proposed heuristics are described in detail. In Section 4 the empirical work is presented. Conclusions of this work are drawn in Section 5.

2 Problem Description

Let $G = (V, E)$ be a planar undirected graph, where V is the set of nodes (blocks or basic units), and E the set of edges. For this particular application an edge between nodes i and j exists if basic units i and j are adjacent. Each basic unit (BU) i has associated parameters such as coordinates (x^i, y^i), and activity values w_i^a, $a \in \{1, 2\}$. In our problem such activities correspond to the number of customers (a=1) and product demand (a=2). The number of territories is fixed and represented by p. A territory is a subset of nodes $X_k \subset V$ $(k = 1, \ldots, p)$. In addition, the problem solution requires that each BU is assigned only to one territory. Hence, territories are defined by a partition of V. Furthermore, the territories must be balanced according to the node activity measures (number of customers and sales volume). The size of a territory X^k with respect

to activity a is defined as follows:

$$w^a(X^k) = \sum_{i \in X^k} w_i^a \tag{1}$$

It is difficult to obtain a perfectly balanced solution due to the discrete problem structure and the exclusive assignment requirement. One way to represent the balance requirement is by introducing a constraint that allows a relative deviation from the target activity value. It is given by a tolerance parameter τ_a, specified by the user. The target average size is computed simply as $\mu^a = w^a(V)/p$. The balancing constraint becomes $w^a(X^k) \in [(1-\tau^a)\mu^a, (1+\tau^a)\mu^a]$. The contiguity requirement means that for any pair of BUs i and j in a given territory, there must exist an i-j path totally contained in the territory. In other words, territory X^k must induce a connected subgraph of G. A rigorous definition of compactness does not exist; but a territory is said to be compact if it is somewhat round-shaped and undistorted. Thus, a dispersion function must give a measure of how far a BU is from each other in each territory. One way to measure the dispersion of a territory X^k is using a p-center objective function. This measure computes the distance from the farthest BU to its corresponding territory center (given by $c(k)$), $k = 1, \ldots, p$.

$$f(X^k) = \max_{j \in X^k} \{d_{c(k),j}\} \tag{2}$$

where $d_{c(k),j}$ is the Euclidean distance from node j to its territory center, denoted by $c(k)$. We define a territory center $c(k)$ of territory k as the node with the smallest distance to its farthest node, that is:

$$c(k) = \arg\min_{i \in X^k} \{\max_{j \in X^k} \{d_{ij}\}\} \tag{3}$$

Let Π be the collection of all p-partitions of V. Then, the problem can be described as finding a p-partition $X = (X^1, ..., X^p)$ of V satisfying the specified planning criteria of balancing and contiguity, that minimizes the above distance-based dispersion measure. The combinatorial description of the problem is the following:

$$\underset{X \in \Pi}{\text{Minimize}}\, f(X) \quad = \quad \max_{\substack{k=1,\ldots,p \\ j \in X^k}} \{d_{c(k),j}\} \tag{4}$$

subject to

$$w^a(X^k)/\mu^a \quad \in \quad [1-\tau^a, 1+\tau^a] \quad a \in A \tag{5}$$

$$G(X^k, E(X^k)) \quad \text{is} \quad \text{connected.} \quad k = 1, \ldots, p \tag{6}$$

The objective function (4) minimizes the territory dispersion. Constraints (5) represent the balance in each territory. Constraints (6) assure the connectivity of each territory. The problem is NP-hard [13]. Our experience shows that one can optimally solve with branch-and-bound methods

instances of up to 150 BUs. The target instances have between 500 and 2000 BUs, so the choice of heuristics is clearly justified.

3 Solution Approach

The main goal of this work is to propose alternative constructive procedures that would generate better or more diverse solutions than those obtained in [13]. Following the previous work, we also derive GRASP-based algorithms but in a very different way. GRASP [7] is a multi-start metaheuristic widely used for solving many combinatorial problems. Basically, each GRASP iteration consists of two phases: construction and local search (Figure 1). In RF, a solution is built iteratively by creating one single territory at a time. The criterion for *closing* a territory and *opening* a new one is when the upper bound of the balancing constraint reach its limit. At the end, this leads to obtaining a number of territories different (and usually larger) than p. An adjustment phase consisting of a merging operation is then applied, but this produces a high degree of violation in the balancing constraints (5). In our work, we attempt to overcome this limitation by building the p territories simultaneously. Therefore, the basic difference between our proposed procedures and the previous work is the construction phase.

```
function procedure GRASP( )
Input: instance data.
Output: best solution found S*.

0   f* ← ∞;
1   while (stopping criterion not satisfied) do
2         S ← Greedy randomized construction;
3         S ← Local search(S);
4         if (f(S) < f*) then
5               f* ← f(S);
6               S* ← S;
7         end if
8   end while
9   return S*;
end GRASP
```

Figure 1: A GRASP pseudocode.

3.1 Constructive methods

Solutions are generated in two phases which we henceforth call *location* and *allocation*. In the location phase, p BUs are selected to be territory *seeds*. In the allocation phase, the unassigned BUs are allocated to one of the territories initialized with a *seed* basic unit, taking into account

contiguity and balancing constraints. Thus, p territories are initialized, one for each seed BU, where the seed basic unit is not necessarily the territory center $c(k)$.

3.1.1 Location phase

The location phase consists of finding p BUs that are used as seeds in the location phase. Given the nature of the objective function, it is desired these seed nodes are as disperse as possible. This can be done by solving a p-dispersion problem. Given a set of candidate points, the p-dispersion problem [5] consists of selecting a subset of p points, such that the minimum distance between any pair of these chosen points is as large as possible. In this work, three location heuristics are proposed. Two of them can be seen as extensions from the p-dispersion heuristics in Erkut, Ülküsal, and Yeniçerioğlu [6], and the other considers a 1-step look-ahead policy.

procedure GRLH1 (α, p)
Input: α= GRASP RCL location quality parameter; p = number of territories.
Output: p-seed set V_c.

0 $V_c \leftarrow \emptyset$;
1 Select i^* and j^* in V, such that $d(i^*, \{j^*\}) = \max\{d(i, \{j\}) : i, j \in V\}$;
2 $V_c \leftarrow V_c \cup \{i^*, j^*\}$;
3 **while** ($\mid V_c \mid < p$) **do**
4 Compute $d(j, V_c), j \in V \setminus V_c$;
5 Build $RCL = \{j : d(j, V_c) \geq d^{\max} - \alpha(d^{\max} - d^{\min})\}$;
6 Choose j randomly, $j \in RCL$;
7 $V_c \leftarrow V_c \cup \{j\}$;
8 **end while**
9 **return**(V_c);
end GRLH1

Figure 2: Location procedure GRLH1.

GRLH1: Greedy randomized constructive location heuristic. The greedy randomized constructive location heuristic (GRLH1) requires to find a set V_c of p disperse BUs. Initially $V_c = \emptyset$ (Figure 2). GRLH1 is started by choosing the two farthest points in V (which is the optimal solution for a 2-dispersion problem). Then, during $p - 2$ iterations, a new point is added to V_c. In step 4, distances from node j to the current V_c are measured as $d(j, V_c) = \min_{i \in V_c}\{d_{ij}\}$.

Then a restricted candidate list (RCL) is formed with the best elements (step 5). The new point j is chosen to maximize the minimum distance from j to the elements already in V_c. Once the new seed is randomly selected from the RCL (step 6) and added to the solution (step 7), a new RCL is built to select another BU, and so on. The restricted candidate list (RCL) follows the concept used normally in the GRASP methodology. The RCL is restricted by a quality parameter

α. The α value defines the quality of the elements in the RCL, when $\alpha = 0$, the construction is purely greedy location solution, and if $\alpha = 1$ the location is random.

procedure GRLH2 (α, p)
Input: α= GRASP RCL location quality parameter; p = number of territories.
Output: p-seed set V_c.

0 $V_c \leftarrow \emptyset$;
1 **if** (p mod(2)==0) **then**
2 Select i^* and j^* such that $d(i^*, \{j^*\}) = \max\{d(i, \{j\}) : i, j \in V\}$;
3 $V_c \leftarrow V_c \cup \{i^*, j^*\}$;
4 **else**
5 Select i^*, j^* and k^*; such that $d(i^*, \{j^*, k^*\}) = \max\{d(i, \{j, k\}) : i, j, k \in V\}$;
6 $V_c \leftarrow V_c \cup \{i^*, j^*, k^*\}$;
7 **end if**
8 **while** ($\mid V_c \mid < p$) **do**
9 Compute $\varphi(i, j), i, j \in V \setminus V_c$;
10 Build $RCL = \{(i, j) : \varphi(i, j) \geq \varphi^{\max} - \alpha(\varphi^{\max} - \varphi^{\min})\}$;
11 Choose (i, j) randomly, $(i, j) \in RCL$;
12 $V_c \leftarrow V_c \cup (i, j)$;
13 **end while**
14 **return**(V_c);
end GRLH2

Figure 3: Location procedure GRLH2.

GRLH2: 1-step look-ahead greedy randomized constructive heuristic. The look-ahead greedy randomized constructive heuristic (GRLH2) proceeds like the GRLH1 heuristic. The difference is that the GRLH2 evaluates not only the next unit to be included in the solution, but the next two simultaneously. The purpose of evaluating two basic units simultaneously is to reduce the greediness and, to favor the dispersion by searching in a different space than the one used in GRLH1.

The greedy function φ used to measure the desirability of adding both units i and j to set V_c is defined as follows:

$$\varphi(i, j) \quad = \quad \min\{d(i, V_c), d(j, V_c), d_{ij}\} \tag{7}$$

As usual $\varphi^{\max} = \max_{i,j} \varphi(i, j)$ and $\varphi^{\min} = \min_{i,j} \varphi(i, j)$. Note that the computation of φ is more expensive than the computation of the greedy function used in GRLH1. It is of particular interest if this extra effort pays off in terms of solution quality. Also note that Steps 5-6 in Figure 3 reflect the fact that p might be odd.

GRDL: Greedy randomized destructive location heuristic. The idea behind this procedure is based on the following observation. Given a set of nodes in the space, if a pair of the nearest nodes is identified and one of these is eliminated, the remaining nodes tend to be more disperse in terms

of the dispersion measure. Following this idea, the GRDL procedure (Figure 4) starts with all BUs in the solution, it is ($V_c = V$). Iteratively, the procedure deletes one BU from V_c until p BUs remain in the seeds set (V_c). Since it is desirable to eliminate one of the nearest BUs, we create an RCL such that $RCL = \{(i,j) : d_{ij} \leq d^{min} + \alpha(d^{max} - d^{min})\}$, where $d^{max} = \max_{i,j \in Vc}\{d_{ij}\}$ and $d^{min} = \min_{i,j \in Vc}\{d_{ij}\}$. Then, following the GRASP scheme, a pair (i,j) is randomly chosen from the RCL, and one of these is eliminated from V_c. The BU to be eliminated is determined using an index which relates the number of customers and product demand in a basic unit ($w_i^1 * w_i^2$). This index is used due to the interest in keeping in V_c the basic units with more sales and more customers.

procedure GRDL (α, p)
Input: α= GRASP RCL location quality parameter; p = number of territories.
Output: p-seed set V_c.

0 $V_c \leftarrow V$;
1 **while** ($|V_c| > p$) **do**
2 Build $RCL = \{(i,j) \in V_c : d_{ij} \leq d^{\min} + \alpha(d^{\max} - d^{\min})\}$;
3 Select a pair $(i,j) \in RCL$ randomly;
4 Eliminate basic unit $i^* = \arg\min\{w_i^1 * w_i^2, w_j^1 * w_j^2\}$;
5 $V_c \leftarrow V_c \setminus \{i^*\}$;
6 **end while**
7 **return**(V_c);
end GRDL.

Figure 4: Location procedure GRDL.

3.1.2 Allocation phase

Given the dispersion set $V_c = \{v_1, \ldots, v_p\}$ obtained in the location phase (see Figure 5), we initialize each territory as $X^k = \{v_k\}$, $k = 1, \ldots, p$. Let $V^u = V \setminus V_c$ be the set of unassigned BUs, that is, $|V^u| = |V| - p$. Each unassigned BU $j \in V_u$ must be allocated to only one of the k territories. We define the neighbor set N^k of territory k as those unassigned BUs that are connected by an edge with any BU in territory X^k, $k = 1, \ldots, p$. Hence $N^k = \{j \in V^u : X^k \cup \{j\}$ is connected$\}$. At a given iteration a greedy function $\phi(j,k), j \in V_u, k = 1, \ldots, p$, given by

$$\phi(j,k) = (|N^k|)^e \cdot \Phi(j,k) \tag{8}$$

is computed, where

$$\Phi(j,k) = \lambda f(X^k \cup \{j\}) + (1-\lambda) g(X^k \cup \{j\})$$

$$k \in K, j \in N^k \tag{9}$$
$$g(X^k \cup \{j\}) = \sum_{a \in A} \max\{w^a(X^k) + w_j^a - (1+\tau^a)\mu^a, 0\}$$

Here $f(x)$ is the dispersion measure given by (2) and $g(X^k \cup \{j\})$ corresponds to the total violation of the balancing constraints, when basic BU j is added to territory k. The user-specified λ parameter weighs both dispersion and infeasibility.

Function (8) considers both the neighborhood size of the territory k and the assignment cost (in dispersion and feasibility sense) of assigning BU j to territory k. With these values the RCL is built (Figure 5, step 3). This function is similar to the one used in RF but here it takes into account the size of the neighboring units of the territories. This greedy function is motivated by the observation, in previous approaches, that some territories can not grow because there are no available neighbors during the last iterations of the process. By introducing this function, small territories have more opportunity to grow early in the process, avoiding the stalling during the posterior phases.

procedure Allocation (V_c, β)
Input: V_c = set of disperse BUs; β = GRASP RCL allocation quality parameter.
Output: a solution $X = (X^1, \ldots, X^p)$.

0 $X^k \leftarrow \{v_k\}; V^u \leftarrow V \setminus V_c$;
1 **while** ($|V^u| > 0$) **do**
2 Build the RCL as in (10);
3 Select $(j,k) in RCL$ randomly;
4 Assign the basic unit j to a territory k, $X^k \leftarrow X^k \cup \{j\}$;
5 Update $V^u \leftarrow V^u \setminus \{j\}$;
6 Update center $c(k)$ in territory X^k if needed;
7 **end while**
8 **return**(X);
end Allocation

Figure 5: Allocation procedure.

According to the value of Φ, a candidate list of BUs is constructed as follows:

$$RCL = \{(j,k) : \Phi(j,k) \leq \Phi^{\min} + \beta(\Phi^{\max} - \Phi^{\min})\} \tag{10}$$

As usual $\Phi^{\max} = \max_{k \in K, i \in N^k} \Phi(i,k)$, and $\Phi^{\min} = \min_{k \in K, i \in N^k} \Phi(i,k)$. The parameter β determines the quality of the elements in the RCL. In each iteration the center of the territory must be updated. Initially, the center is the seed basic unit obtained in the location phase.

3.1.3 Sequential Location-Allocation

In our Location-Allocation constructive procedure (SLA) a territory is started and constructed sequentially by iteratively assigning BUs to the active territory. When the territory size of any of the BU activities reaches a predetermined upper limit (PUL) it is closed and a new territory is started. Such PUL can be equal or smaller than the right hand side of expression (5), which is the maximum feasible size that a territory can take for any activity. Proceeding this way, p territories are constructed. Then, the unassigned BUs can be allocated to some of the p territories according to the greedy function (8).

Figure 6 shows the pseudo-code of the SLA procedure. The first territory is initialized in line 1 with the element $j \in V$ with minimal degree. In line 2 j is removed from the set of unassigned BUs, and the neighboring units of territory X^k are determined in line 3. From lines 4-19, p territories are built sequentially until they reach the PUL^a. Both, the selection of a *seeding* BU and the selection of the neighboring BU to be included in the current territory X^k is done within the GRASP scheme. Once a territory has been initialized, the benefit of including a BU j in territory X^k is computed in line 5, according to expression (9). These values are used to build a restricted candidate list in lines 6 and 7. A unit j is randomly chosen from RCL in line 8, and included in the territory X^k in line 9. If the territory size $w^a(X^k)$ reaches a PUL^a value, the current territory is closed, and a new one is initialized (line 11). All the unassigned BUs in V^u are evaluated according to the distance measure $d(j, S) = \min_{i \in S}\{d_{ij}\})$ in line 12. A RCL is built in lines 13 and 14. A BU j is randomly selected from RCL in line 15 and included in the current territory X^k in line 16. In line 18 the neighboring set N^k is updated. In this point there are p partial territories and a set of unassigned units V^u, which are evaluated to be incorporated in some of the p territories (lines 20-26) in the same manner as in Section 3.1.2.

procedure SLA (α, p, PUL^a)

Input: α= GRASP RCL location quality parameter; p = number of territories; PUL^a = predetermined territory size upper limit for activity a.

Output: a solution $X = (X^1, \ldots, X^p)$.

0 $k \leftarrow 1$, $V^u \leftarrow V$;
1 $X^k \leftarrow \{j\}$, where $j \in \arg\min \{|N^j| : j \in V\}$;
2 $V^u \leftarrow V^u \setminus \{j\}$;
3 $N^k \leftarrow$ set of neighbors of X^k;
4 **while** ($k \leq p$ and $N^k \neq \emptyset$) **do**
5 Compute $\phi(j,k)$ in (9) for all $j \in N^k$;
6 $\Phi^{\min} \leftarrow \min_j\{\phi(j,k)\}$; $\Phi^{\max} \leftarrow \max_j\{\phi(j,k)\}$;
7 Build $RCL \leftarrow \{j \in N^k : \phi(j,k) \in [\Phi^{\min}, \Phi^{\min} + \alpha(\Phi^{\max} - \Phi^{\min})]\}$;
8 Choose $j \in RCL$ randomly;
9 $X^k \leftarrow X^k \cup \{j\}$; $V^u \leftarrow V^u \setminus \{j\}$;
10 **if** ($w^a(X^k) > PUL^a$ for any a) **then**
11 $k \leftarrow k + 1$;
12 Compute $d(j,S)$, $j \in V^u$;
13 $d^{\min} \leftarrow \min_j\{d(j,S)\}$; $d^{\max} \leftarrow \max_j\{d(j,S)\}$;
14 Build $RCL = \{j \in V^u : d(j,S) \geq d^{\max} - \alpha(d^{\max} - d^{\min})\}$;
15 Choose $j \in RCL$ randomly;
16 $X^k \leftarrow \{j\}$, $V^u \leftarrow V^u \setminus \{j\}$;
17 **end if**
18 Update N^k;
19 **end while**
20 **while** ($|V^u| > 0$) **do**
21 Build a RCL as in (10);
22 Select (j,k) randomly, $j,k \in RCL$;
23 Assign the basic unit j to a territory k, $X^k \leftarrow X^k \cup \{j\}$;
24 Update $V^u = V^u \setminus \{j\}$;
25 Update center $c(k)$ in territory X^k if needed;
26 **end while**
27 **return**(X);
end SLA

Figure 6: Sequential location-allocation procedure.

3.2 Postprocessing Phase

As usual, a solution constructed by any of the construction schemes is not necessarily a local optimum. In addition, feasibility with respect to the balancing constraints may not be entirely satisfied. Therefore, the local search procedure in Figure 7 aims at both, improving the dispersion (objective function) and obtaining feasibility. A neighborhood $N(X)$ consists of all solutions reachable from $X = (X^1, \ldots, X^p)$ by moving a basic unit i from its current territory $X_{t_{(i)}}$ into a neighboring territory $X_{t_{(j)}}$, where j is the corresponding BU in territory $X_{t_{(j)}}$ adjacent to i, to keep the connectivity requirement. The move is denoted by $move(i, j)$. In our case we measure the cost of a move by:

$$\psi(\text{move } (i,j)) \quad = \quad \gamma \Delta f_{ij} + (1-\gamma)\Delta g_{ij} \tag{11}$$

where Δf_{ij} is the variation in the objective value when the move (i, j) is performed. Similarly, Δg_{ij} is the variation in the value of the feasibility measure, after move (i, j). The user-defined parameter γ is used to weigh both the objective and feasibility. Thus, given a solution X the objective and feasibility values are obtained with (12) and (13), respectively.

$$f(X) \quad = \quad \left(\frac{1}{d_{\max}}\right) \max_{k=1,\ldots,p} \left\{ \max_{i,j \in X_k} d_{ij} \right\} \tag{12}$$

$$g(X) \quad = \quad \sum_{k=1}^{p} \sum_{a \in A} g^a(X_k) \tag{13}$$

The local search procedure starts with an initial solution X obtained in the construction phase.

procedure Postprocessing (S, γ)
Input: X = solution; γ = dispersion and feasibility weighting in postprocessing phase.
Output: an improved solution $S^{'}$.

0 $X' \leftarrow X$; $\psi(X') \leftarrow \psi(X)$; *stopping criteria* = FALSE;
1 **while** (*stopping criteria* = FALSE) **do**
2 Compute $\psi(X_{ij})$ for all $move(i, j) \in N(X)$;
3 **if** $(\psi(X_{ij}) < \psi(X'))$ **then** $X' \leftarrow X_{ij}$; $\psi(X') \leftarrow \psi(X_{ij})$;
4 update *stopping criteria*;
5 **end while**
6 **return**($X^{'}$);
end Postprocessing

Figure 7: Local search.

The merit function value becomes the incumbent value $\psi(X') \leftarrow \psi(X)$. Iteratively, all the possible *moves* in the neighborhood $N(X)$ of the current solution X are evaluated according to (11). If the merit function value $\psi(X_{ij})$ obtained with $move(i, j)$ is better than the incumbent value $\psi(X')$, the merit function value is updated to the incumbent value $\psi(X') \leftarrow \psi(X_{ij})$. The algorithm continues until a stopping criterion is reached.

4 Empirical Work

The proposed solution procedures have been coded in C++ and compiled with a Sun C++ compiler workshop 8.0 under Solaris 9 operating system. The tested instances were taken from the data set of [13]. These are randomly generated instances. All instances use a tolerance value τ^a equal to 5%. We used two instance sets with 1000 and 2000 BUs, and each instance was evaluated with p = 40 and p = 60 territories.

4.1 Parameter Fine Tunning

In this section, the parameter setting procedure used for SLA is described. This procedure has three parameters: α, λ, and γ which were fine-tunned by using the methodology described in [2]. It consists of four steps: i) selecting a subset of problems to analyze, ii) determining the variation range for each parameter, iii) selecting the best value for each parameter by carrying out an appropriate design and analysis of experiments, iv) combining the settings obtained in step 3 for obtaining high-quality parameter values. The details of these four steps are the following.

Step i: First of all, we select representative subsets of the available instances according to their characteristics. Four sets are available, instances with 1000 nodes and 40 territories, 1000 nodes and 60 territories, 2000 nodes and 40 territories, and 2000 nodes and 60 territories. A subset of three instances from each group is selected for the parameter setting experiment.

Table 1: Parameter values for the SLA parameter setting experiment.

Parameter	Lower value	Center	Upper value
α	0.2	0.3	0.4
λ	0.3	0.5	0.7
γ	1.0	-1.0	-3.0

Step ii: The second step consists of determining the parameter levels to carry out the experiments. This is done with a preliminary test in order to find out an approximation of the best parameter values for the experiment. Thus, as a result of this preliminary test, the center, the upper value and the lower value for each parameter is determined (Table 1). The change (Δ) in the value for each parameter are Δ_α=0.1, Δ_λ=0.2, and Δ_γ=2. Recall α defines the quality of the values contained in the RCL during the location phase, λ weights the dispersion and the feasibility in the merit function used to build the RCL in the allocation phase. In SLA procedure, the parameter γ_a defines an upper bound (PUL^a) for the territory size $w^a(X^k)$. When $\gamma_a = 0$ the upper limit size for all territories regarding activity a is equals to the target

size (μ_a). When $\gamma_a = 1$ the upper limit is $(1 + \tau_a)\mu_a$, and $\gamma_a = -1$ means the upper limit is $(1 - \tau_a)\mu_a$. This upper limit is defined in expression $\text{PUL}^a = \mu_a + (\gamma_a * \mu_a * \tau_a)$.

Step iii: This step consists of generating an experimental design. Since there are only three parameters to fit, we use a full factorial experiment. Three instances from each group are used in the experiment. Thus, for each instance in the group and for each combination of parameters, the SLA procedure is applied 10 times. The best solution found for each instance is kept for comparisons.

For example, three instances are selected from the group with 1000 nodes and 40 territories, and then these are solved with SLA using the following parameters: $\alpha = 0.2$, $\lambda = 0.3$, and $\gamma = 1$. Note that, these values are the lower values shown in Table 1. Then, we have the best solution found for each one of these three instances in the group. This is repeated with all parameter combinations (27 combinations). These 27 parameter combinations are used to solved the other groups of instances. Once all instances have been solved by applying SLA over the 27 parameter combinations, the average infeasibility value obtained is computed for each instance group. Such average values of infeasibility ($Infeas$) are detailed in Table 2. Our first goal is to obtain feasible solutions before improving the dispersion measure. Thus, the main objective of the parameter setting is to find the best combination of parameters that yield best balance in the territories. It means that, the relative deviation on each territory is within the allowed tolerance (then $Infeas = 0.0$). For that reason in this step of the parameter settings we try to find the best parameter combination that minimizes the infeasibility value.

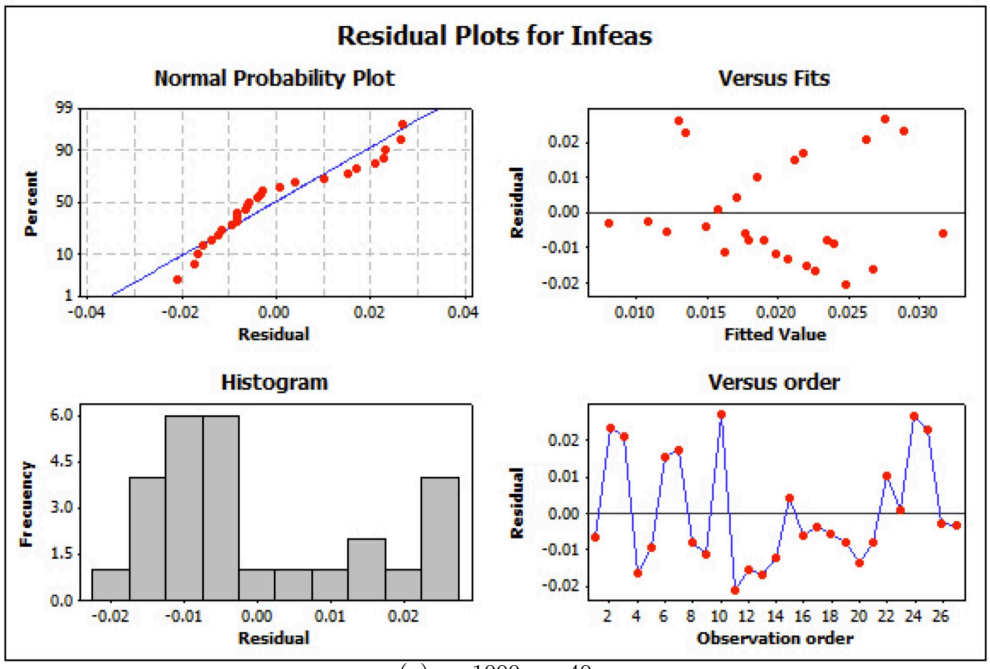
After all chosen instances have been solved by applying the SLA over the 27 combinations of parameters. The next step is to fit a linear model for each instance category, using the 27 experimental runs on each of them. The basic idea is to find a linear approximation of the response surface. The dependent variable in the models is the infeasibility value in Table 2. The independent variables are the parameters used in each experimental run (α, λ, and γ). Table 3 shows parameter coefficients in the linear regression equation obtained for each group of instances (see also Figures 8 and 9 that contain the residual plots from the models). The statistical significance of the parameter varies in each instance category.

Table 2: Average results for infeasibility values for each group.

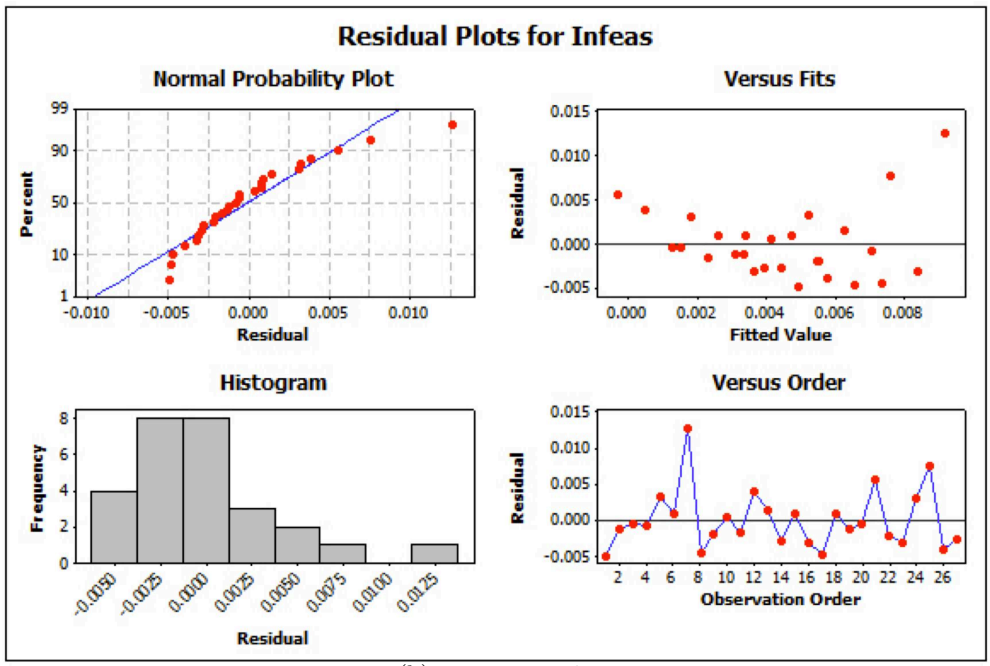
Parameter values			Instance categories			
α	λ	γ	n=1000, p=40	n=2000, p=40	n=1000, p=60	n=2000, p=60
0.2	0.3	1	0.02540	0.00000	0.09947	0.06237
0.2	0.3	-1	0.05220	0.00176	0.05993	0.03210
0.2	0.3	-3	0.04717	0.00073	0.05173	0.03395
0.2	0.5	1	0.01028	0.00622	0.06053	0.03983
0.2	0.5	-1	0.01470	0.00848	0.06403	0.01879
0.2	0.5	-3	0.03650	0.00429	0.04530	0.02630
0.2	0.7	1	0.03880	0.02180	0.12933	0.00988
0.2	0.7	-1	0.01084	0.00269	0.13300	0.03773
0.2	0.7	-3	0.00481	0.00348	0.07267	0.02953
0.3	0.3	1	0.05457	0.00453	0.10560	0.05376
0.3	0.3	-1	0.00392	0.00067	0.09163	0.03860
0.3	0.3	-3	0.00674	0.00436	0.11277	0.04477
0.3	0.5	1	0.00562	0.00770	0.15280	0.03137
0.3	0.5	-1	0.00776	0.00160	0.09320	0.01703
0.3	0.5	-3	0.02120	0.00347	0.20600	0.03883
0.4	0.7	1	0.01148	0.00516	0.09320	0.02593
0.4	0.7	-1	0.01093	0.00177	0.06760	0.03849
0.4	0.7	-3	0.00653	0.00557	0.09023	0.04449
0.4	0.3	1	0.01537	0.00217	0.11823	0.00674
0.4	0.3	-1	0.00731	0.00095	0.07667	0.05551
0.4	0.3	-3	0.00980	0.00530	0.06100	0.00973
0.4	0.5	1	0.02887	0.00334	0.07517	0.02326
0.4	0.5	-1	0.01656	0.00051	0.09003	0.03440
0.4	0.5	-3	0.03942	0.00494	0.09890	0.01745
0.4	0.7	1	0.03634	0.01523	0.12500	0.00962
0.4	0.7	-1	0.00799	0.00181	0.15600	0.04283
0.4	0.7	-3	0.00485	0.00120	0.06007	0.02080

Table 3: Linear regression coefficients.

Group	Intercept	α	λ	γ	F	P	Adj.R^2
n=1000, p=40	0.04610	-0.0412	-0.0250	0.00138	1.20	0.331	2.3%
n=2000, p=40	0.00237	-0.0078	0.0106	0.00091	2.47	0.067	16.6%
n=1000, p=60	0.05540	0.0805	0.0417	0.00446	0.92	0.448	0.0%
n=2000, p=60	0.05370	-0.0389	-0.0217	-0.00009	0.94	0.438	0.0%

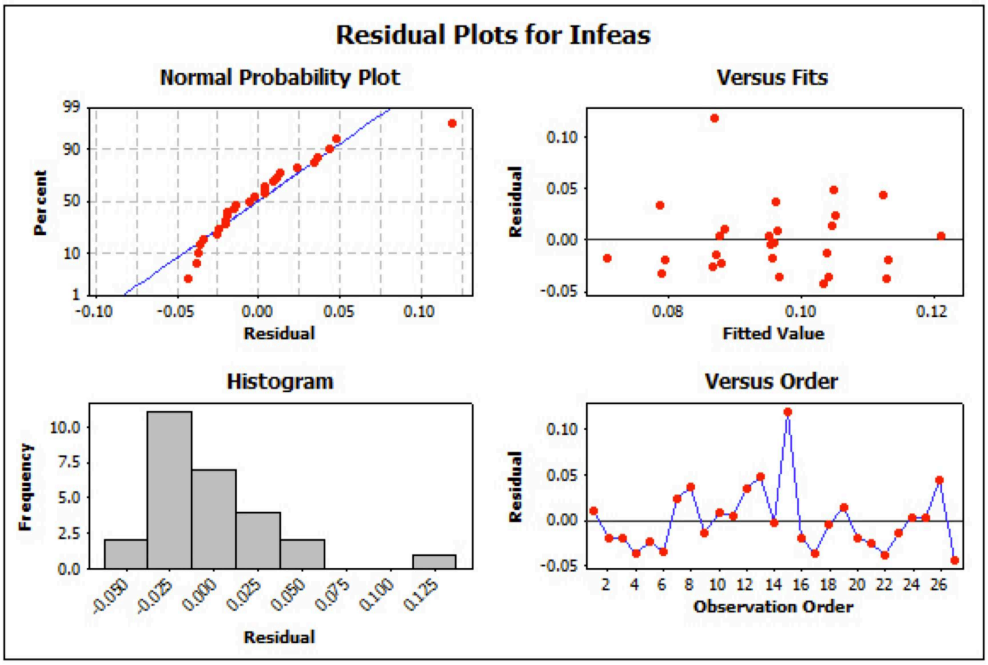


(a) n=1000, p=40

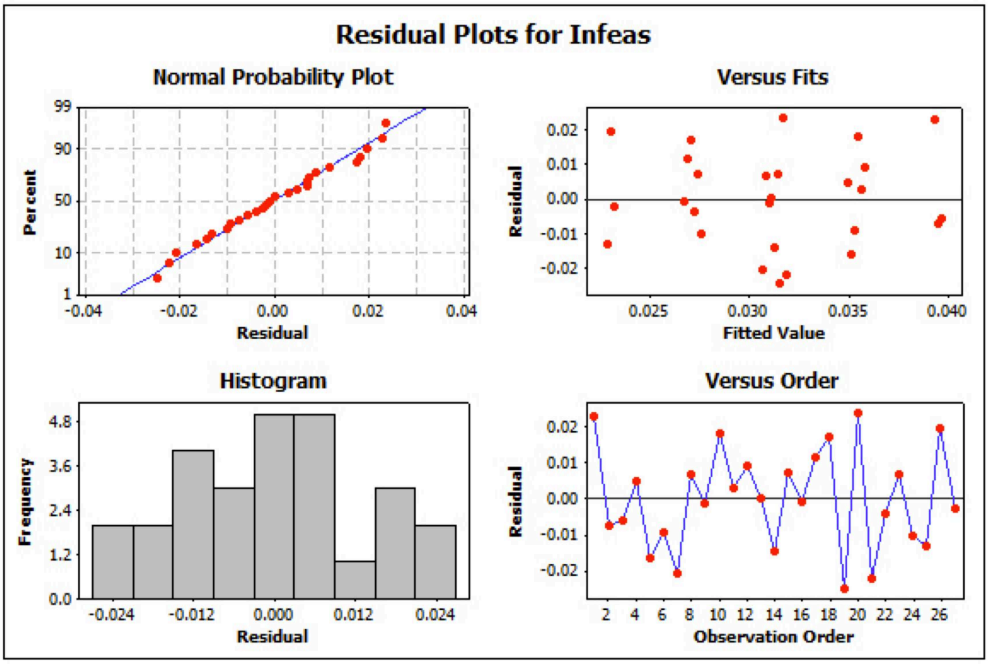


(b) n=2000, p=40

Figure 8: Residual plots for infeasibility (Infeas) for p=40 instances.



(a) n=1000, p=60



(b) n=2000, p=60

Figure 9: Residual plots for infeasibility (Infeas) for p=60 instances.

The following step in the parameter settings process is to obtain the path of the steepest descent(PSD). Given an estimated regression coefficient $\mathbf{b}= (b_\alpha, b_\lambda, b_\gamma)$, the PSD is the negative gradient of the linear model (-$\mathbf{b}$). Now we must travel along the path by making small movements from one initial point. In our case such point is the center of the design ($\alpha = 0.3, \lambda = 0.5, \gamma = -1$). To calculate a full step size, the following procedure is used: select the regression coefficient with maximum absolute value (b_{max}), divide each parameters coefficient (b_j) by b_{max}, and multiply the resulting value by the Δ value of the corresponding parameter. For example, by using the information in Table 3, for the group with n=1000 nodes and p=40 territories, the maximum absolute coefficient value is the one of α (b_{max}=0.0412). Then, the full step size for α is $b_\alpha/b_{max} * \Delta_\alpha$ = (-0.0412 / 0.0412) * 0.1 = -0.1. Similarly, for λ the full step size is (-0.0250 / 0.0412) * 0.2 = -0.1214, and for γ is (0.00138 / 0.0412) * 2 = 0.067. Nevertheless, as suggested in [2], it is desirable to avoid stepping over potential good local minima, so we make steps of 1/4 of the full step (authors explain the compromise between performance and complexity). The step size must be obtained for all the parameters and all the instance categories. All the step sizes are shown in Table 4 (1/4 of the full step size).

Table 4: Step size along the path of steepest descent.

Group	α	λ	γ
n=1000, p=40	-0.0250	-0.0300	0.0170
n=2000, p=40	0.0184	-0.03030	0.0167
n=1000, p=60	0.0250	0.02580	0.0227
n=3000, p=60	0.0250	0.02780	0.0011

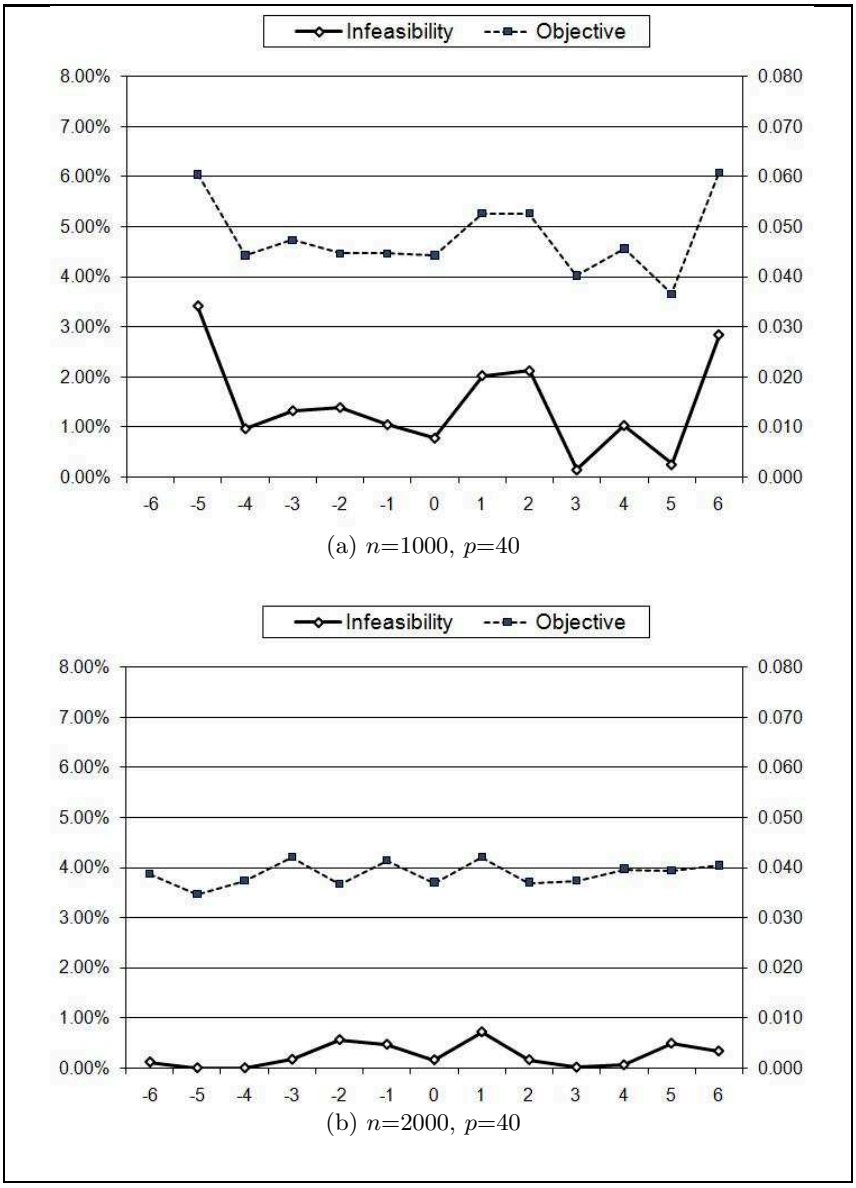
As mentioned before, to start the moves along the path of steepest descent, we start from the design center (step 0). We calculate steps forward and backward. Forward steps are done subtracting the step size value for each parameter from its previous value (at first, the previous value corresponds to the center). In backward steps, step size values are added. After a step has been calculated, we perform trials with the corresponding parameter values in the current step. In our test, we run the trials with the same 3 instances of each category, used in the previous experimental design. Table 5 shows the parameter values and the average from the objective function value, infeasibility, and CPU time obtained for the 3 trial instances with n=1000 nodes and p=40 territories. Figure 10(a) displays the average values obtained for each step when we used instances with 1000 nodes and 40 territories. The best infeasibility values are obtained in step 3, meanwhile the best objective function values are obtained with

step 5. As mentioned before, it is our particular interest to produce feasible solutions. Thus, for setting the final values of parameters, only the infeasibility values are taken into account.

Table 5: Average results for 3 instances with n=1000 and p=40.

	Parameter values			Average		
Step	α	λ	γ	Objective	Infeasibility	Time (sec)
-4	0.200	0.379	-0.933	0.04433	0.00965	14.197
-3	0.225	0.401	-0.950	0.04733	0.01323	14.247
-2	0.250	0.439	-0.967	0.04466	0.01386	14.393
-1	0.275	0.470	-0.983	0.04466	0.01044	14.393
0	0.300	0.500	-1.000	0.04433	0.00776	14.437
1	0.325	0.530	-1.017	0.05266	0.02022	14.357
2	0.350	0.561	-1.033	0.05266	0.02118	14.447
3	0.375	0.591	-1.050	0.04033	0.00144	14.547
4	0.400	0.621	-1.067	0.04566	0.01028	14.647
5	0.425	0.652	-1.084	0.03666	0.00246	16.107
6	0.450	0.682	-1.100	0.06066	0.02835	14.593

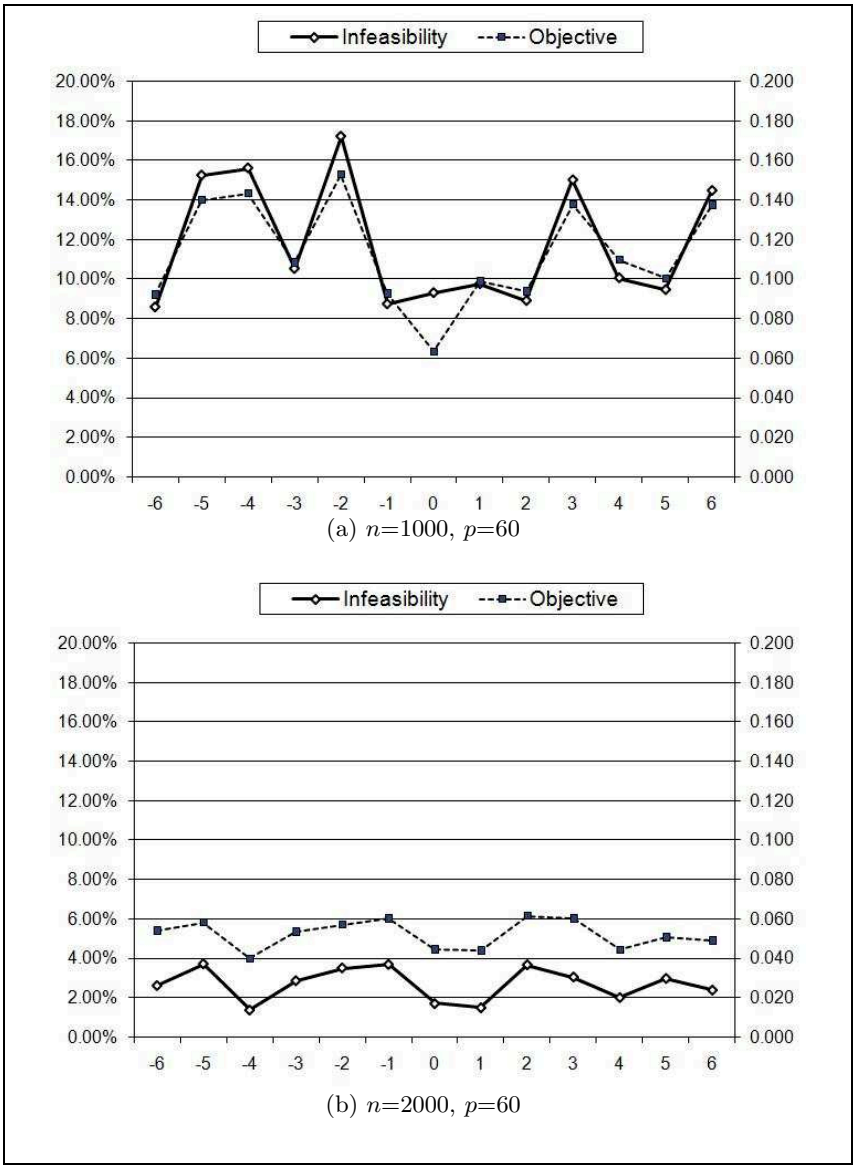
Step iv: After the steepest descent has been computed for all instance groups, and the better values have been identified for each parameter on each group, the final parameter values to perform the final test of the heuristic are defined by averaging the better parameter values from all instance groups. Thus, after following the guidelines in [2], the final parameter values are the following: α=0.368, λ=0.527, γ=-0.998.



(a) n=1000, p=40

(b) n=2000, p=40

Figure 10: Steepest descent average results for objective function and infeasibility values for p=40 instances.



(a) n=1000, p=60

(b) n=2000, p=60

Figure 11: Steepest descent average results for objective function and infeasibility values for p=60 instances.

4.2 Comparison of Greedy Procedures

In this part of our experimental work, we compared the four proposed procedures. The existing procedure RF [13] is also included in the experiment. It has been necessary to follow the parameter setting procedure for each heuristic: GRLH1, GRLH2, GRDL, and SLA. Table 6 shows the parameter values for each heuristic. The dash means that the procedure does not requires the corresponding parameter. Since the goal of this experiment is to compare the greedy nature of the procedures, then we used $\alpha = 0$. Table 7 displays the size and number of the tested instances.

Table 6: Parameter values used for comparing the greedy procedures.

Parameter	GRLH1	GRLH2	GRDL	SLA	RF
α	0.00	0.00	0.00	0.00	0.00
λ	0.10	0.10	0.10	0.527	0.6
e	0.90	0.90	0.90	-	-
γ	-	-	-	-0.998	-

Table 7: Size of test instances.

Instance group		
n	p	No. of instances
1000	40	50
2000	40	50
1000	60	50
2000	60	50
Total		200

The infeasibility values are compared after the construction and improvement phases. Recall we are interested in obtaining better (feasible if possible) solutions to make easier and more productive the improvement phase than the existing RF procedure. The relative deviation index (14) is used:

$$rel.dev. \quad = \quad \frac{|best\_infeas - infeas_h|}{(infeas_h + \epsilon) * 100\%} \tag{14}$$

where $best\_infeas$ is the smaller infeasibility value obtained with any of the 5 compared procedures, $infeas_h$ is the infeasibility value obtained with procedure h, (h=GRLH1, GRLH2, GRDL, SLA, RF). Since infeasibility can take a value equal to zero, expression (14) avoids division by zero.

Average relative deviation results are shown in Tables 8 and 9. In the greedy construction phase, procedures GRLH1 and GRLH2 obtain similar average relative deviation values for all instance groups, except for the group with n=2000 and p=60, where GRLH1 has 19.86% and

Table 8: Relative deviation for infeasibility results with greedy constructive procedures.

Instance group		Procedure				
n	p	GRLH1	GRLH2	GRDL	SLA	RF
1000	40	15.31%	15.12%	36.28%	69.61%	58.57%
2000	40	10.58%	13.30%	23.53%	61.24%	57.15%
1000	60	23.19%	22.11%	40.08%	65.76%	32.20%
2000	60	19.86%	11.78%	33.71%	66.08%	60.30%
Average		17.23%	15.34%	33.54%	65.57%	52.05%

Table 9: Relative deviation for infeasibility results after improvement.

Instance group		Procedure				
n	p	GRLH1	GRLH2	GRDL	SLA	RF
1000	40	50.73%	59.49%	77.81%	82.12%	70.00%
2000	40	48.12%	50.13%	65.73%	83,52%	80.31%
1000	60	51.62%	63.07%	67.30%	70.13%	36.54%
2000	60	50.69%	55.24%	71.96%	80.26%	66.64%
Average		50.29%	56.98%	70.70%	79.01%	63.37%

GRLH2 has 11.78%. Both procedures improve the performance of GRDL, SLA, and the existing procedure RF. The difference between the average relative deviation from GRLH1 and GRLH2 is 1.89% in favor of GRLH2 which uses 1-step look-ahead approach in the selection of seed nodes for starting the territories, which requires more computational effort. It was not possible to obtain any feasible solutions in the greedy construction phase.

After the improvement, relative deviation values increase because infeasible values are smaller than those in the construction phase. Even better, feasible solutions have been obtained. On average, the local search phase in each procedure obtains the following improvement: 95.03% for GRLH1, 94.85% in GRLH2, 94.07 in GRDL, 96.26% for SLA, and 94.60% for RF. Local search in GRDL improves more than any other procedure, but solutions in the construction phase are the worst regarding feasibility. Again, GRLH1 and GRLH2 outperform the results of GRDL, SLA, and RF. Regarding the global average relative deviation results for infeasibility, GRLH1 improves GRLH2 by 6.69%. GRLH1 finds feasible solutions for 10.5% of the 200 instances, GRLH2 for 9%, GRDL for 6%, SLA for 2.5%, and RF for 4.5% (see Table 10). Doing the analysis by groups, the ideal couples for group (n=1000, p=40) is GRLH1, for instances with (n=2000, p=40) is GRLH1, for (n=1000, p=60) is RF, and for (n=2000, p=60) is GRLH1.

Table 11 shows the CPU time (in seconds) employed on each phase for each procedure. Each

Table 10: Number of feasible solutions found with greedy procedures after improvement.

Instance group		Procedure				
n	p	GRLH1	GRLH2	GRDL	SLA	RF
1000	40	7	4	1	2	4
2000	40	12	12	9	3	4
1000	60	0	0	1	0	0
2000	60	2	2	1	0	1
Total		21	18	12	5	9

phase is indicated by L (location), A (allocation), and local search (LS). Observe that, the construction of territories in a sequential way, only requires allocation and local search processes. Recall that, in SLA each new seed node is selected by considering the farthest unallocated node from those already allocated to some territory. In RF, each new seed node is selected by considering the grade of the unassigned nodes. For these two procedures, the time required for selecting the new seeding node is included in the allocation phase (A), since it is small. The time required in local search is in general relatively small. Only in 2000-node instances local search requires more than 1 second in average. On the other hand, the location phase in GRLH1, GRLH2, and GRDL requires more computational effort, mainly in GRLH2. Except by the computational effort criteria, GRLH1 produce better results, obtaining more feasible solutions after local search.

Table 11: Average time (sec) by phase.

		Procedure												
Instance group		GRLH1			GRLH2			GRDL			SLA		RF	
n	p	L	A	LS	L	A	LS	L	A	LS	A	LS	A	LS
1000	40	3.31	2.31	0.21	9.84	2.26	0.19	2.78	2.06	0.22	1.12	0.25	0.21	0.20
2000	40	17.89	8.47	1.75	54.73	7.91	0.88	15.79	7.63	0.99	5.05	1.23	1.12	1.0
1000	60	3.50	3.08	0.24	23.72	3.17	0.21	2.75	3.06	0.23	1.57	0.26	0.21	0.18
2000	60	17.89	1.17	1.17	137.51	11.10	0.82	15.38	10.81	0.92	6.99	1.09	0.97	0.87

4.3 Comparison of GRASP Procedures vs Exact Procedure

In this experiment we compare the proposed heuristics GRLH1, GRLH2, GRDL, and SLA against an exact procedure. For this experiment we use relatively small instances with 60, 80, and 100 nodes and 4, 5, and 6 territories, respectively. Five instances of each set were generated according to the previously described specifications. A total of 5 x 3 = 15 DT instances were generated.

The exact procedure is the one used in [14], which is an iterative algorithm that solves the relaxed problem of the corresponding MILP model (with no connectivity constraints, recall there is an exponential number of them), and then, it identifies and adds violated connectivity constraints until optimality is reached. This exact procedure uses the number of iterations (10 iterations maximum) and time limit by iteration (7200 sec.) as stopping criteria.

Table 12: Summary of results for small instances.

	No. of optimal solutions					Av. relative gap (%)			
Data set	Exact	GRLH1	GRLH2	GRDL	GRSLA	GRLH1	GRLH2	GRDL	GRSLA
DT60	5	2	2	2	1	1.17	0.64	1.54	2.84
DT80	5	0	0	0	0	5.84	1.47	7.08	4.84
DT100	5	0	0	0	0	8.46	1.55	9.61	7.28

Table 13: Average CPU times for small instances.

	Procedure				
Data set	Exact	GRLH1	GRLH2	GRDL	GRLSA
DT60	190.8	0.6	1.0	10.4	0.4
DT80	2331.8	1.0	2.1	35.7	0.6
DT100	12179.0	1.5	4.7	94.9	1.0

Regarding the heuristics, they were run for 100 iterations (construction and improvement). The heuristics GRLH1, GRLH2, and GRDL were tested using the following parameter values: $\alpha = 0.3$, $\lambda = 0.1$ and $e = 0.9$. The parameters for SLA were those obtained in Section 4.1. Parameter values λ and e were also determined by following the procedure proposed by Coy et al. [2] which is based on an experimental design. Recall that, the parameter e is used to fit the number of the neighboring nodes for a territory in the allocation phase. In the first column of Table 12 the instance sets used are shown (DT60, DT80 and DT100). This table shows the number of optimal solutions found by applying each procedure over each instance set, and the average of the relative optimality (gap). Table 13 displays the CPU time in seconds. All those solutions obtained with the heuristics were feasible solutions. The optimal value with the exact method was found for all instances. Table 14 shows objective function values, and the CPU times in detail.

Heuristic procedures can find optima only for instances with 60 nodes. Over all instance sets, the average relative optimality (gap) computed for GRLH1, GRLH2, and GRDL is 5.1566%, 1.22%, and 6.0766%, respectively. GRLH1 is the fastest heuristic (it requires on average 1.04 seconds). GRLH2 and GRDL take longer than GRLH1, on average 2.57 and 47.01 sec., respectively. Due

Table 14: Results for small instances.

	Exact		GRLH1		GRLH2		GRDL		GRSLA	
Instance	Sol.	time	Sol.	time	Sol.	time	Sol.	time	Sol.	time
DT60-1	205.10	562	205.10	0.57	205.10	0.92	205.10	10.49	205.10	0.36
DT60-2	172.94	158	174.40	0.58	174.40	0.96	181.10	10.27	174.40	0.40
DT60-3	180.82	31	186.66	0.63	180.82	1.00	182.97	10.59	186.66	0.50
DT60-4	186.09	65	186.09	0.68	187.20	0.99	186.09	10.23	201.60	0.40
DT60-5	176.39	136	179.51	0.61	179.51	0.94	179.51	10.51	179.51	0.40
DT80-1	157.90	169	169.30	0.98	164.40	2.09	169.30	35.77	169.30	0.64
DT80-2	156.28	329	157.06	1.01	158.28	2.11	157.06	35.68	157.06	0.06
DT80-3	161.92	4693	176.05	1.00	162.12	1.99	177.39	35.65	164.94	0.80
DT80-4	148.56	2198	167.54	1.02	166.00	2.07	163.33	35.90	162.90	0.72
DT80-5	158.32	4270	158.32	1.04	170.74	2.00	171.25	35.69	166.19	0.69
DT100-1	144.06	10982	152.95	1.47	156.07	4.70	152.95	92.55	156.78	1.01
DT100-2	170.05	3493	182.29	1.56	170.53	4.68	184.62	94.04	174.01	1.08
DT100-3	147.21	8908	160.74	1.51	160.74	4.84	160.95	95.15	163.65	1.02
DT100-4	137.12	2743	157.03	1.51	147.18	4.68	162.03	98.88	152.75	0.99
DT100-5	159.99	34769	168.36	1.49	159.99	4.70	169.31	93.80	164.30	1.00

to the previous results, in the next subsection we compared only the heuristic GRLH1 against the existing procedure (RF).

4.4 Comparison with existing approach

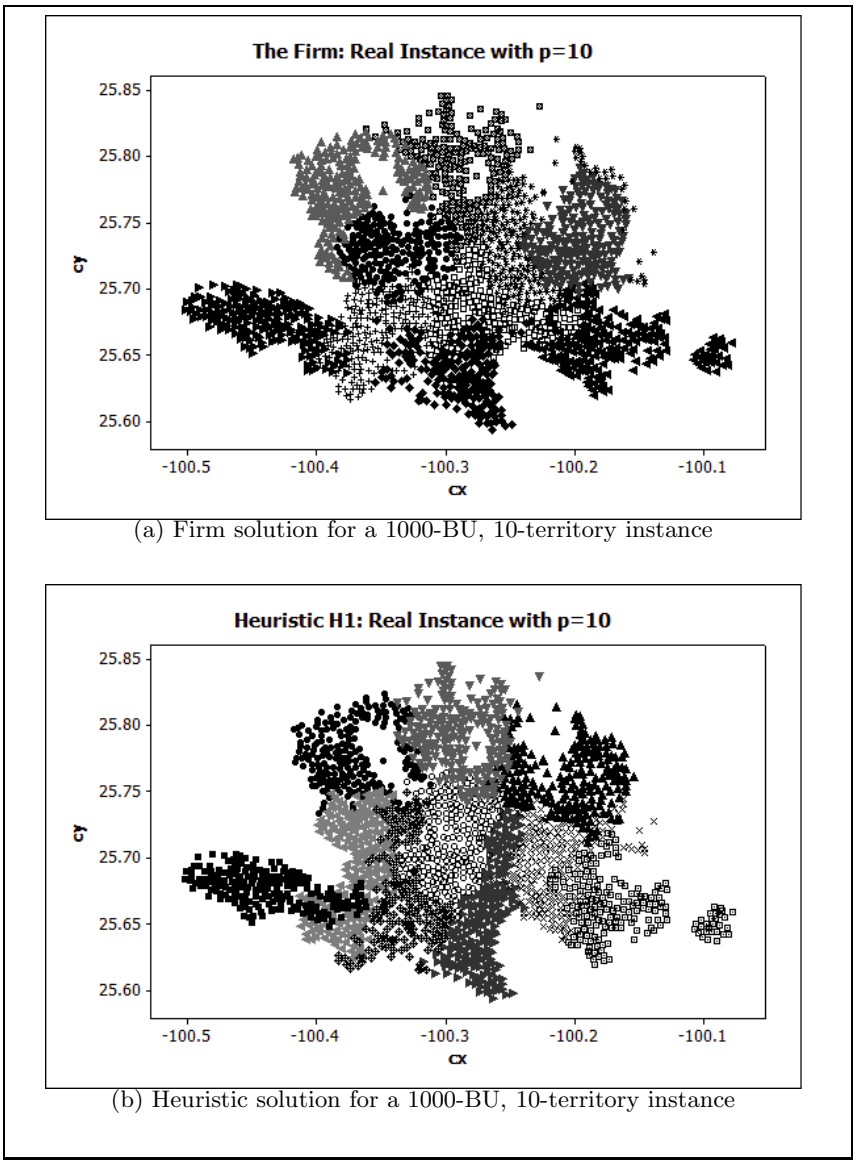
In this experiment the heuristic GRLH1 is compared with the original procedure RF. We decided to compare only the proposed heuristic GRLH1 because of the results in the previous sections, related to quality in greedy procedures, and CPU time required.

Table 15: Results for GRASP procedures

Instance group		*nfeas*		*ARLSI (%)*	
n	p	GRLH1	RF	GRLH1	RF
1000	40	44	50	96.4	94.7
2000	40	50	50	97.4	96.3
1000	60	1	2	92.4	93.7
2000	60	32	38	95.9	96.6

The procedures are executed during 100 iterations using the test bed of problem instances used in previous experiments with 1000 and 2000 nodes and 40 and 60 territories. In each iteration a construction and improvement phases are performed. The parameters values used for the GRLH1

are $\alpha = 0.3$, $\lambda = 0.1$, and $e = 0.9$. Table 15 shows a comparison between GRLH1 and RF. Column $nfeas$ represents the number of feasible solutions found (out of 50) with each procedure GRLH1 and RF separated by instance groups. The average relative improvement of the local search with respect to phase 1 solutions is denoted by ARLSI. One can see that even though procedure RF produces slightly better results (in terms of the number of feasible solutions), procedure GRLH1 is very competitive. In terms of the average relative local search improvement, GRLH1 showed a slightly better improvement value for the $p = 40$ instances. Both procedures report average local search improvements of more than 92 %. With this very good quality, it is strongly suggested that both procedures can be used in a collaborative way.



(a) Firm solution for a 1000-BU, 10-territory instance

(b) Heuristic solution for a 1000-BU, 10-territory instance

Figure 12: Comparison between designs obtained by the firm and proposed method.

Finally, Figures 12(a) and 12(b) show the resulting designs when the firm's method and the proposed GRLH1 heuristic, respectively, are applied to a 1000-BU, 10-territory real-world instance. For this case, under the firm method the best resulting design has an objective function value of 0.094, and it is infeasible with respect to the balancing constraints under a 0.05 tolerance. Our method obtains a significantly better design in terms of both objective function value and feasibility, with a dispersion function value of 0.076 (a relative improvement of 24%), and territory imbalances of less than 5%.

5 Conclusions

A version of a territory design problem motivated by a real-world application is addressed in this work. The problem planning requirements are compactness, contiguity, and balancing with respect to two activities (number of customers and sales volume). A location-allocation heuristic framework is proposed. In the location phase, three p-dispersion based heuristics are proposed. Such heuristics obtained p disperse seeds (nodes) for starting the territories. In the allocation phase all unassigned nodes are incorporated iteratively to some territory (building all the territories simultaneously). These procedures were incorporated within a GRASP scheme, including a local search phase. The empirical work reveals that two of the proposed heuristics find near-optimal or optimal solutions to relatively small instances, where exact solutions could be found. The heuristic solutions are found significantly faster. When we compared with the existing method in larger instances, it was found that the existing approach provides solutions with lower infeasibility violations. However, one of the proposed procedures found better solutions in terms of its dispersion measure than the ones found by the existing approach.

This means that the idea of building the territories simultaneously can in some cases provide solutions with lower degree of infeasibility after the construction phase, and therefore lead to better overall solutions. As lines of future work, developing more sophisticated local search procedures such as tabu search and memory-based strategies such as adaptive programming can be worthwhile.

Acknowledgements: The presentation of the paper was improved thanks to the comments by two anonymous reviewers. This research has been supported by the Mexican National Council for Science and Technology (CONACYT) through grants SEP-CONACYT 48499-Y and SEP-CONACYT 61343, and by Universidad Autóma de Nuevo León through its Scientific and Technological Research Support Program, grants UANL-PAICYT CA1478-07, CE012-09, and IT511-10. The first author also acknowledges the support by CONACYT's Support Program for Postdoctoral Researchers.

References

[1] P. Bertolazzi, L. Bianco, and S. Ricciardelli. A method for determining the optimal districting in urban emergency services. *Computers & Operations Research*, 4(1):1–12, 1977.

[2] S. P. Coy, B. L. Golden, G. C. Runger, and E. A. Wasil. Using experimental design to effective parameter settings for heuristics. *Journal of Heuristics*, 7(1):77–97, 2000.

[3] A. Drexl and K. Haase. Fast approximation methods for sales force deployment. *Management Science*, 45(10):1307–1323, 1999.

[4] J. C. Duque, R. Ramos, and J. Suriñach. Supervised regionalization methods: A survey. *International Regional Science Review*, 30(3):195–220, 2007.

[5] E. Erkut and S. Neuman. Comparison of four models for dispersing facilities. *INFOR*, 29(2):68–86, 1991.

[6] E. Erkut, Y. Ülküsal, and O. Yeniçerioğlu. A comparison of p-dispersion heuristics. *Computers & Operations Research*, 21(10):1103–1113, 1994.

[7] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.

[8] J. A. Ferland and G. Guénette. Decision support system for the school districting problem. *Operations Research*, 38(1):15–21, 1990.

[9] D. Haugland, S. C. Ho, and G. Laporte. Designing delivery districts for the vehicle routing problem with stochastic demands. *European Journal of Operational Research*, 180(3):997–1010, 2007.

[10] S. W. Hess, J. B. Weaver, H. J. Siegfeldt, J. N. Whelan, and P. A. Zitlau. Nonpartisan political redistricting by computer. *Operations Research*, 13(6):998–1006, 1965.

[11] J. Kalcsics, S. Nickel, and M. Schröder. Toward a unified territorial design approach: Applications, algorithms, and GIS integration. *Top*, 13(1):1–56, 2005.

[12] L. Muyldermans, D. Cattryse, D. Van Oudheusden, and T. Lotan. Districting for salt spreading operations. *European Journal of Operational Research*, 139(3):521–532, 2002.

[13] R. Z. Ríos-Mercado and E. Fernández. A reactive GRASP for a commercial territory design problem with multiple balancing requirements. *Computers & Operations Research*, 36(3):755–776, 2009.

[14] M. A. Salazar-Aguilar, R. Z. Ríos-Mercado, and M. Cabrera-Ríos. New models for commercial territory design. *Networks & Spatial Economics*, 11(3):487–507, 2011.

[15] A. A. Zoltners and P. Sinha. Sales territory design: Thirty years of modeling and implementation. *Marketing Science*, 24(3):313–331, 2005.