A Location-Allocation-Improvement Heuristic for Districting with Multiple-Activity Balancing Constraints and p-Median-Based Dispersion Minimization[1]

Roger Z. Ríos-Mercado[2]
Universidad Autónoma de Nuevo León (UANL)
Graduate Program in Systems Engineering
San Nicolás de los Garza, NL 66455, Mexico
roger.rios@uanl.edu.mx

Ada M. Álvarez-Socarrás
Universidad Autónoma de Nuevo León (UANL)
Graduate Program in Systems Engineering
San Nicolás de los Garza, NL 66455, Mexico
ada.alvarezsc@uanl.edu.mx

Andrés Castrillón
Xpertal Global Services
General Anaya 601, Col. Bella Vista
Monterrey, NL 64410, Mexico
andres.castrillon@xpertal.com

Mario C. López-Locés
Universidad Autónoma de Nuevo León (UANL)
Graduate Program in Systems Engineering
San Nicolás de los Garza, NL 66455, Mexico
mariocesar@lopezloc.es

October 2019; Revised August 2020, September 2020

[1] *Computers & Operations Research* (forthcoming)
[2] Corresponding author

Abstract

A districting problem consisting of the minimization of a dispersion function subject to multiple-activity balancing and contiguity constraints is addressed. This problem arises from a real-world application in a commercial firm responsible for distributing bottled beverage products. The objective is to find a partition of a set of city blocks into a given number of territories that minimizes a p-median problem-based dispersion function. Minimizing this measure allows for territory compactness. Districts must be connected and balanced with respect to both activities: number of customers and workload. A heuristic procedure based on a location-allocation scheme is proposed for this NP-hard problem. This technique has been successfully used for similar territory design problems with single-activity balancing constraints. The proposed method is an iterative algorithm that consists of two phases: first centroids are determined or fixed (location phase) and then units are assigned to centroids (allocation phase). The success of this technique relies on some theoretical properties that no longer hold for problems having multiple-activity balancing constraints. The novelty of our work is to design a framework for handling both connectivity and multiple-activity balancing constraints simultaneously. The solution framework is enhanced by a local search phase that attempts to improve the quality of solutions found in the allocation phase. The empirical work includes a thorough study of the different components of the algorithm, solution of very large scale instances, and comparison with existing work. Extensive empirical testing reveals the effectiveness of the proposed approach, including the impact of both the location-allocation component and its local search component. It is based on the intelligent exploitation of problem structure. The algorithm significantly outperforms one of the best-known heuristics for this problem in terms of feasibility success and solution quality. It obtains similar quality solutions than the ones obtained by the other known heuristic in less computational effort.

Keywords: Discrete optimization; territory design; location-allocation method; local search; heuristics; p-median problem.

1 Introduction

Territory design consists basically of grouping small geographical areas into larger geographical clusters called territories according to specific planning criteria. These problems belong to the family of *districting problems* that have many different applications, such as political districting, design of school territories, social facilities, emergency services, sales, and service territory design. An extensive recent study of districting problems and methodologies to find suitable solutions to these problems can be found in Kalcsics and Ríos-Mercado [19].

The territory design problem (TDP) studied in this paper was motivated by a real-world application from a bottled beverage distribution company. Given a set of city blocks or basic units (BUs) with known measurable attributes such as geographic coordinates, number of customers, and product demand, the company needs to partition its customers into a fixed number of territories p in the city according to specific planning criteria. One of these requirements is that of territory balancing, that is, the firm wishes to obtain balanced territories (similar in size) with respect to each of two different activity measures (number of customers and product demand). Territory contiguity is also required, that is, for each pair of BUs in the territory, there must be a path contained in that territory. In addition, compact territories are desired; that is, customers within a territory should be as close to each other as possible. This criterion is typically achieved by minimizing a dispersion function.

The choice of this dispersion measure leads to different models and solution methodologies. One popular way of measuring dispersion is by using the function of the p-Median Problem (pMP), a well-known problem from the location literature. For districting problems with pMP objective, one of the most successful methods found in literature is the location-allocation algorithm. This scheme consists of a two-stage iterative process where territory centers are first located, and then customers are allocated to centers by solving a linearly relaxed transportation-type problem. After solving this problem, there are a few fractional variables that must be an integer. This new problem is called the split resolution subproblem that must be solved to complete one iteration of the algorithm. Successful application of this location-allocation heuristic applied to districting problems can be found in the works of Hess et al. [17], Fleischmann and Paraschis [12], George et al. [14]. Although these applications have unique features that make each have a different structure, a common feature is that they all share single-activity balancing constraints. This particular feature makes the location-allocation algorithm work because there is a beautiful theoretical result [18] establishing that the number of split units (fractional variables of the relaxed subproblem) is bounded by $p - 1$. This result no longer holds when the model has multiple-activity balancing constraints.

The model studied in this paper has been addressed before, particularly from the exact optimization perspective [7, 36]. The exact algorithms developed for this problem are limited to

relatively small- and medium-size instances of around 200-250 BUs and below. We aim to target instances of 1000-2000 BUs. This paper's main contribution is to extend the location-allocation technique to handle multiple-activity balancing and connectivity constraints simultaneously. In addition, our proposed location-allocation algorithm is enhanced by a local search phase whose purpose is to improve the quality of a solution. To the best of our knowledge, there are two other known heuristics for this particular model: the divide-and-conquer heuristic developed by Salazar-Aguilar et al. [38] and a tabu search metaheuristic developed by Gliesch and Ritt [15]. Our work includes a comparison of our method with these two heuristics.

Other commercial territory design models have focused on p-center-based objectives [32], or diameter-based objectives [31]. None of these methods apply to our problem due to the different model structures rendered by the different objective measures. Furthermore, location-allocation types of methods do not work for center-based dispersion measures because the number of split units in the transportation subproblem may result in a hugenumber.

The proposed algorithm was implemented and widely tested over a variety of randomly generated instances based on real-world data provided by our industrial partner. The results show the effectiveness of the proposed approach and several of its components, as it was able to obtain solutions of good quality (both in terms of its compactness measure and feasibility with respect to the balancing constraints).

In particular, it was observed the positive impact of the location-allocation phase delivering feasible or nearly feasible solutions of relatively good quality. In addition, the local search component of the algorithm showed average relative improvements from 50-75% with respect to the solution from the allocation phase. We also observed that the number of split nodes grows in a well-behaved manner as the size of the instance grows. This observation shows that although we no longer have a theoretical bound on the number of split nodes, the empirically found growth is relatively modest. We also found that the algorithm is 100% reliable; that is, it can find feasible solutions in all instances tested. This aspect was a key issue to investigate empirically. Compared with existing work, the proposed heuristic significantly outperformed the divide-and-conquer heuristic in feasibility success, solution quality, and running time. When compared with the tabu search heuristic [15], it was observed that both heuristics were very competitive. Both were very reliable in finding feasible solutions all the time. The quality of the solutions delivered by both heuristics was very similar, although ours achieved these results in less computational effort.

The rest of the paper is organized as follows. Section 2 presents a description of the problem and its corresponding mixed-integer programming formulation. A literature review of the most relevant work is given in Section 3. This is followed by Section 4, which contains a full description of the proposed heuristic approach, describing each of its main components. Extensive computational testing assessing many aspects of the proposed method is carried out in Section 5. We wrap up with conclusions and closing remarks in Section 6.

2 Problem Description

A graph $G = (V, E)$ is used to model the problem, where a city block or basic unit i is represented by a node, and an edge between nodes i and j exists if nodes i and j represent adjacent blocks. Every node $i \in V$ has the following parameters: geographical coordinates (c_i^x, c_i^y) and two measurable activities. Let n be the number of nodes, that is, $n = |V|$. Let w_i^a be the measure of activity a in node i, $a \in A$, where $A = \{1, 2\}$ is the set of activities in each node. A territory is represented by a subset of nodes $V_k \subset V$, where each node must be assigned to only one territory. The number of territories is a given parameter p.

A solution to this problem must balance every territory with respect to each activity. Nevertheless, due to the unique assignment constraints and to the discrete nature of the problem is practically impossible to get perfectly balanced territories. To handle this issue, we measure the degree of balance by computing the relative deviation of each territory with regard to its ideal weight given by $\mu^a = \sum_{i \in V} w_i^a / p$. In addition, each territory must be connected. This means that each territory in the solution must induce a connected subgraph of G to guarantee the existence of a path entirely contained in the territory for every pair of nodes.

Finally, it is desired to get compact territories so that customers are relatively near to each other. This can be achieved by minimizing a dispersion measure. There are several measures of compactness used in the literature [42]. In this particular case, a p-median dispersion function of the form $\sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij}$ is used, where x_{ij} takes the value of 1 if unit j is assigned to the territory with center in i and 0 otherwise, $i, j \in V$; and d_{ij} represents the Euclidean distance between nodes i and j. The motivation for this choice is twofold. First, it is essential for the company to have a sensitive measure with respect to all basic units, which is achieved by this type of function. In other words, the p-median metric has the advantage that it changes whenever a basic unit is moved from one territory to another, therefore appearing more responsive, something measures such as the p-center metric lack. Second, it has been shown that this type of function renders tighter polyhedral representations of the convex hull of integer solutions, and therefore, it provides tighter LP-relaxations that speed up branch-and-bound running times. In addition, the mixed-integer linear programming (MILP) model under this type of function has some nice theoretical properties that can be exploited within a location-allocation solution scheme. This scheme will be further elaborated in Section 4.

that the p-median value has the advantage that it changes whenever a basic unit is moved, therefore appearing more "responsive", something which measures like the p-center lack

Therefore the problem can be stated as finding a p-partition of V, satisfying all the specific planning criteria, that minimizes the given dispersion measure. The problem can be formulated as the following MILP:

$$(P_0) \quad \text{Minimize} \quad f(x) = \sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{i \in V} x_{ij} = 1 \qquad j \in V \tag{2}$$

$$\sum_{i \in V} x_{ii} = p \tag{3}$$

$$\sum_{j \in V} w_j^a x_{ij} \leq (1 + \tau^a) \mu^a x_{ii} \qquad i \in V,\ a \in A \tag{4}$$

$$\sum_{j \in V} w_j^a x_{ij} \geq (1 - \tau^a) \mu^a x_{ii} \qquad i \in V,\ a \in A \tag{5}$$

$$\sum_{j \in \cup_{v \in S} N^v \setminus S} x_{ij} - \sum_{j \in S} x_{ij} \geq 1 - |S| \qquad i \in V,\ S \subset V \setminus (N^i \cup \{i\}) \tag{6}$$

$$x_{ij} \in \{0, 1\} \qquad i, j \in V \tag{7}$$

The objective (1) represents the measure for territory dispersion and it is based on the p-median problem dispersion function as explained before. Constraints (2) guarantee that each node j is assigned to a territory. Constraint (3) forces the number of p territories. Note that according to the definition of the binary variables x_{ij}, a variable $x_{ii} = 1$ implies that basic unit i is a territory center. Constraints (4)-(5) represent the territory balance with respect to each activity measure and establish that the territory size must fall within a range (measured by a tolerance parameter τ^a) of μ^a, the average size. This average size is computed as $\mu^a = \sum_{j \in V} w_j^a / p$ and τ^a is in (0,1) typically between 0.05 and 0.10. The upper bound balancing constraints (4) also ensure that if no center is placed at i, no customer can be assigned to it. Constraints (6) guarantee the connectivity of the territories, where N^i represents the set of nodes adjacent to node i. These constraints (6) were proposed by Drexl and Haase [8], and they are similar to those used in routing problems to guarantee the connectivity of routes. Note that there is an exponential number of such constraints. Preliminary work reveals that the largest instance that can be solved by exact methods is on the order fo 100-120 nodes and 4-8 territories. The intended target instances have around 500-1000 nodes and 40 territories, which are intractable by exact methods. This observation motivates the development of the proposed work.

Computational complexity: Problem P_0 is NP-hard [36]. Note that although the number of connectivity constraints is exponential on the size of the problem, given a p-partition X of V, checking whether each component of X is connected can be done in polynomial time by breadth-first-search for instance. In other words, feasibility can be verified in polynomial time, that is P_0 is in NP. Then, by taking a particular case when G is a complete graph (which implies that the graph is always connected) and τ^a takes a very high value (which implies the balancing constraints are

always satisfied), the problem becomes the p-Median Problem, which is well-known to be NP-hard. That is pMP is polynomially reducible to P_0, therefore P_0 is NP-hard.

3 Related Work

Districting or territory design has a broad range of applications covering traditional areas such as political districting [3, 24, 26, 27], sales territory design [44], school districting [4], power districting [1, 6], and public services [2, 5, 25], and more non-traditional areas such as health care [13, 23] and waste electric and electronic equipment collection [11, 30, 35], to name a few. The reader can find in the works of Kalcsics and Ríos-Mercado [19], and Duque et al. [9] state of the art surveys on models, algorithms, and applications to districting problems. In addition, Zoltners and Sinha [44] present a survey focusing on sales districting, Ricca et al. [28] present a survey on political districting, Liberatore et al. [20] present a literature review on police districting, and Yanık and Bozkaya [43] highlight recent works on districting models in health care. A recent book by Ríos-Mercado [29] presents recent advances in districting and territory design models and algorithms. In this section, we focus the review on two areas relevant to our research: (i) commercial districting and (ii) location-allocation approaches to districting problems.

3.1 Commercial Districting

Ríos-Mercado and Fernández [32] introduced the commercial TDP (CTDP) by incorporating a territory compactness criterion and a fixed number of territories p. They seek to maximize this compactness criterion subject to planning requirements such as exclusive BU-to-territory assignment, territory connectivity, and territory balancing with respect to three BU attributes: number of customers, product demand, and workload. In their work, the authors consider minimizing a dispersion function based on the well-known p-center problem's objective function. After establishing the NP-completeness of the problem, the authors propose a Reactive GRASP for obtaining high-quality solutions to this problem. The core of their GRASP is a three-phase iterative procedure composed of a construction phase, an adjustment phase, and a local search phase. In the construction phase, a solution with q territories, where q is usually larger than p, satisfying the connectivity constraints is built. Then, an adjustment phase based on a pairwise merging mechanism is applied to obtain a solution with p territories. Afterward, a local search phase attempting to eliminate the infeasibility with respect to the balancing requirements and improve the dispersion objective function is applied. One interesting observation is that the construction and adjustment phases produce solutions with a very high degree of infeasibility. This infeasibility is very nicely repaired by the local search at a very high computational cost, in any case. The reason for this is that attempting to merge two territories into one in the adjustment phase may result in a high violation of the upper bound of the balancing constraints.

Salazar-Aguilar et al. [36] developed an exact optimization scheme for solving several CTDP models with double balancing and connectivity constraints. They used their framework for solving two linear models with both types of dispersion functions: the one based on the p-center problem (CPTDP) and the one based on the p-median problem (MPTDP). Models with an objective function from the p-median problem could be solved faster than the others, and solutions obtained from the relaxation of the median-based models had a very high degree of connectivity. They can successfully solve instances of up to 100 BUs for the CPTDP and up to 150 BUs for the MPTDP. They conclude that p-center-based dispersion measures yield more difficult models as they have weaker LP relaxations than the median-based models. Later, Díaz et al. [7] present exact optimization algorithms and lower bounding schemes for the MPTDP. Their two proposed exact algorithms are based on different problem relaxations. The first one uses the relaxation of an integer quadratic programming formulation. The second methodology obtains feasible solutions using a primal heuristic within the framework of a subgradient optimization algorithm to solve a Lagrangian dual that also provides lower bounds for the optimal solution. Sandoval et al. [41] develop an enhanced exact optimization algorithm for the CPTDP. All these exact methods [7, 36, 41] seem adequate for solving small- to medium-size instances of CPTDP and MPTDP.

Ríos-Mercado and Salazar-Acosta [34] present a heuristic based on GRASP and adaptive memory programming for a CTDP that considers the minimization of a p-center problem function subject to additional budget routing constraints.

López-Pérez and Ríos-Mercado [21] and Ríos-Mercado and López-Pérez [33] extend the CTDP model by incorporating additional planning criteria such as joint and disjoint assignment requirements and similarity with the existing plan. Joint (disjoint) assignment means that a given set of customers must be assigned to the same (different) territory. Similarity with the existing plan means that the new plan must be similar to the previous plan by allowing only a small portion of the basic units to be assigned to different territories. In this work, the authors use a p-median problem objective function for measuring dispersion. The authors present a new mixed integer linear model and a solution framework based on a cut generation strategy within a branch-and-bound algorithm for solving the allocation level for a fixed set of territory centers. The proposed method could efficiently handle instances up to 10,000 basic units obtaining high-quality solutions.

Ríos-Mercado and Escalante [31] present a metaheuristic based on GRASP and path relinking for the CTDP where a diameter based objective function is minimized subject to connectivity and multiple-activity balancing constraints. Static and dynamic strategies enhance the path relinking component of their method. The diameter dispersion function measures the maximum distance between any two basic units within a territory; that is, it does not depend on centers.

CTDP has also been addressed from a multiobjective optimization perspective. Salazar-Aguilar et al. [37] present an exact optimization method for obtaining Pareto fronts for relatively small instances for the problem where both territory compactness and balance are simultaneously opti-

mized. Salazar-Aguilar et al. [39] and Salazar-Aguilar et al. [40] develop heuristic methods based on scatter search and GRASP, respectively, for addressing larger instances. Their heuristics find good quality approximations to the Pareto fronts; however, in each of these multiobjective optimization approaches, center-based functions are used for measuring territory dispersion.

In terms of developing lower bounds, Elizondo-Amaya et al. [10] present a dual bounding scheme based on Lagrangian relaxation for the CTDP with p-center-based objective function and no connectivity constraints.

To the best of our knowledge, the only works addressing the CTDP with the same features present in our problem, that is, minimization of a p-median objective function subject to connectivity and multiple-activity balancing constraints are due to Salazar-Aguilar et al. [38] and Gliesch and Ritt [15]. In the former, the authors propose a divide-and-conquer approach to this problem with limited success. Their results indicate that their procedure is unreliable as it struggles to find feasible solutions for large-scale instances due to the difficulty of producing balanced territories in the last iterations of their heuristic. In the latter, the authors present a tabu search metaheuristic that reports interesting results. They were able to solve very large instances with high degree of feasibility success. One of our work's main motivations is to develop a robust and reliable heuristic for this problem and illustrate the usefulness of the location-allocation framework in this regard.

3.2 Location-Allocation Approaches

In a location-allocation heuristic, the simultaneous location and allocation decisions are decomposed into two independent phases, which are iteratively performed until a satisfactory result is obtained. In the first (location) phase, the centers of the territories are chosen, while in the second (allocation) phase, the basic units are assigned to these centers. In Section 4 the location-allocation method is explained in detail. Location-allocation approaches to districting problems are particularly useful for problems with single-activity balancing constraints under the minimization of a p-median objective function due to some important theoretical properties.

The first location-allocation algorithm was proposed by Hess et al. [17] for political districting. In that work, the authors modeled a political districting problem with single-activity balancing constraints as a p-median problem. They applied the method to solve instances of up to 299 basic units and 35 districts. One limitation of that work is that they handle the connectivity issue by ignoring the disconnected solutions obtained during the iterative procedure.

In subsequent studies using this location-allocation idea, the allocation phase is solved as a transportation problem by relaxing the integer variables. Optimal solutions are obtained to the allocation subproblem but at the cost of obtaining a few fractional values meaning there may be some basic units assigned to more than one territory. A theoretical result due to Hojati [18] establishes that at most $p - 1$ variables are fractional, with p being the number of territories.

These fractional basic units are also called *split units or areas*. Therefore, an additional step for assigning these split areas to a single territory is followed. This problem is called the split resolution subproblem, and it is also NP-hard. Therefore usually heuristics are applied to solve the split area subproblem. The different studies differ mainly on how they develop heuristics to address this issue.

Hess and Samuels [16] propose a simple tie-breaking heuristic that assigns a unit to the territory that owns the largest share of the split area. Marlin [22] indicates that split units may be kept in particular cases, mainly when the split unit contains a large amount of activity. He suggests assigning the smaller areas without significantly violating the constraints and then solving the entire problem again. However, this procedure does not reduce the number of splits in general.

Fleischmann and Paraschis [12] present a sophisticated local assignment heuristic with relatively good results. They apply their method in a real/world case from a German company for consumer goods with 1400 basic units and 168 districts.

George et al. [14] show an alternative method for resolving the issue of split geographic units. They set up this split area problem as a network flow problem by defining some particular cost functions. They applied their method to a political districting problem in New Zealand with 35,000 basic units and 93 electoral districts.

It is important to note that in all these works, the location-allocation scheme has been applied to districting problems with single-activity balancing constraints. In this regard, our work's main contribution is to use this location-allocation idea to handle problems with both connectivity and multiple-activity balancing constraints.

4 Proposed Heuristic

The proposed heuristic is based on a location-allocation procedure. This scheme typically consists of two iterative phases. In the first phase, territory centers are placed systematically. Once centers are fixed, the number of binary variables reduces considerably such that the remaining model, called the allocation model, is easier to solve. In the second phase, the allocation model is solved, assigning the remaining basic units to territory centers. When solving the allocation phase, a related model, where the integrality restriction of the binary variables is relaxed, is considered. This allocation subproblem resembles a transportation problem. The solution to this related allocation model may have fractional values. However, there is an interesting result that bounds this number of fractional variables. Each of these fractional variables is called a split variable. The split resolution problem consists of finding integer values for each of these fractional variables. Once this is done, territory centers are re-computed, and the procedure goes on to the next iteration. These two phases are iteratively performed until a given number of iterations without solution improvement is reached.

Procedure 1 Location-Allocation( )

Input: A TDP instance; m:= iteration limit

Output: $X = (X_1, \ldots, X_p)$ A p-partition solution to TDP

1: $X^{\text{best}} \leftarrow \emptyset$, $X \leftarrow \emptyset$, $iter \leftarrow 1$, $Q \leftarrow \emptyset$
2: **while** ($iter \leq m$) **do**
3: $V_c \leftarrow location(X)$
4: **if** ($V_c \in Q$) **then**
5: break
6: **end if**
7: $Q \leftarrow Q \cup V_c$
8: $X \leftarrow allocation(V_c)$
9: **if** (X is connected) **then**
10: $X \leftarrow localsearch(X)$
11: **end if**
12: **if** (X is better than X^{best}) **then**
13: $X^{\text{best}} \leftarrow X$
14: $iter \leftarrow 1$
15: **else**
16: $iter \leftarrow iter + 1$
17: **end if**
18: **end while**
19: **return** X^{best}

Now, our research contribution is as follows. First, rather than facing single-activity balancing constraints, which is the case in previous applications of the location-allocation scheme, we deal with multiple-activity balancing constraints. Thus, some of the theoretical results bounding the number of split variables no longer hold. In addition, we also have very complex connectivity constraints. Therefore our proposed location-allocation scheme incorporates a split resolution heuristic to handle both connectivity and multiple-activity balancing constraints, and a third phase, called improvement phase, where a local search procedure is applied attempting to improve both feasibility and solution quality. The pseudo-code of the proposed location/allocation scheme is shown in Procedure 1. The individual components are described next. Note that Steps 4-7 reflect an efficiency check that stops the algorithm if a given set of centers V_c is repeated because the algorithm is deterministic. Q is a collection of subsets $Q = \{V_{c_1}, V_{c_2}, \ldots\}$ such that V_c is an element of Q. Also, note that the local search in Step 10 is carried out only if solution X found in the allocation phase meets the connectivity constraints. This step will be further explained in Section 4.3.

4.1 Location Phase

In this phase a new configuration of territory centroids (or medians) is found from the solution found in the previous iteration. This can be done simply by solving a 1-median problem for each territory. It is basically a sorting procedure. For each territory, a new median c_k is chosen by minimizing the sum of the distances to every other node in the territory, that is, $c_k = \arg\min_{i \in V_k} \sum_{j \in V_k} d_{ij}$, where V_k is the set of nodes in territory k.

At the start of the very first iteration, we do not have a configuration of territory medians yet. One way to obtain this initial configuration is by solving the MILP until a feasible solution is found. However, this procedure may not be appropriate for large instances. Another way is to simply use any p-partition-based heuristic. In our case, we use the following procedure based on the implementation of the well-known p-means heuristic. First, p basic units are randomly chosen as initial centroids. Second, each remaining unassigned basic unit is assigned to its closest centroid. This assignment defines p territories. Once territories are formed, new centroids for each territory are updated, and the process is repeated. This iterative procedure takes place until no new set of centroids is found. This procedure is a very quick heuristic that returns, as output, an initial set of centroids.

4.2 Allocation Phase

This is the heart of the algorithm. The configuration of centroids (or medians in this case) located in the previous phase is used as an input to this phase. Let $V_c \subset V$ be the set of p pre-determined territory centroids. By fixing the corresponding binary variables (that is, setting $x_{ii} = 1$ for each $i \in V_c$), we could reformulate problem P_0 with a smaller number of variables and constraints. The resulting model is as follows. Note that the number of variables has been reduced from n^2 to np and constraints (3) have been removed.

$$(P_1) \qquad \text{Minimize} \quad f(x) = \sum_{i \in V_c} \sum_{j \in V} d_{ij} x_{ij} \tag{8}$$

$$\text{subject to} \quad \sum_{i \in V_c} x_{ij} = 1 \qquad j \in V \tag{9}$$

$$\sum_{j \in V} w_j^a x_{ij} \leq (1 + \tau^a)\mu^a \qquad i \in V_c,\ a \in A \tag{10}$$

$$\sum_{j \in V} w_j^a x_{ij} \geq (1 - \tau^a)\mu^a \qquad i \in V_c,\ a \in A \tag{11}$$

$$\sum_{j \in \cup_{v \in S} N^v \setminus S} x_{ij} - \sum_{j \in S} x_{ij} \geq 1 - |S| \qquad i \in V_c,\ S \subset V \setminus (N^i \cup \{i\}) \tag{12}$$

$$x_{ij} \in \{0, 1\} \qquad i \in V_c,\ j \in V \tag{13}$$

Now, let us considered a related problem where binary variables x_{ij} are relaxed and the tolerance parameters τ^a is set to zero. The resulting model, parameterized by $a \in A$ is given by:

$$
\begin{aligned}
(P_1(a)) \qquad \text{Minimize} \quad & f(x) = \sum_{i \in V_c} \sum_{j \in V} d_{ij} x_{ij} && && (14)\\
\text{subject to} \quad & \sum_{i \in V_c} x_{ij} = 1 && j \in V && (15)\\
& \sum_{j \in V} w_j^a x_{ij} = \mu^a && i \in V_c && (16)\\
& x_{ij} \geq 0 && i \in V_c,\ j \in V && (17)
\end{aligned}
$$

Although it is not quite unimodular, the problem has a similar structure as that of a transportation problem and can therefore be solved efficiently by the simplex method. It is necessary to emphasize that there are two problems, one for each activity. In the optimal solution of each problem $P_1(a)$ the territories are perfectly balanced. However, it is possible that the unique assignment constraint is not satisfied due to the relaxation of the binary variables. After solving the problem $P_1(a)$, every node is assigned to a centroid. However, we can find some basic units for which more than one variable x_{ij}, $i \in V_c$, have positive values. These areas are called *split units* or just *splits*. Note that this means that these basic units have been assigned to two or more territories. An important theoretical result [18] establishes that the number of split units is bounded by $p - 1$. This result is very meaningful because it implies that regardless the size of V, we will always obtain at most $p - 1$ split variables when solving $P_1(a)$.

There is is another issue related to our problem, particularly when considering the fact that we are dealing with multiple-activity balancing constraints. While it is true that each problem $P_1(a)$ may have at most $p-1$ fractional salutions, this does not necessarily mean that we will have $2(p-1)$ splits when considering both problems. There may be integer variables that are associated to different centroids in each model, such that these are also split variables. Unfortunately, there is no theoretical bound for these new split variables. This issue is further investigated in the experimental work.

Now it is necessary to assign the split variables to only one centroid. This has to be done in such a way that the original problem remains feasible (that is the original balancing and connectivity constraints are met) while minimizing the objective function. This problem is called the *split resolution problem* and it is presented next.

Split Resolution Problem

Let V^a the set of split units in the optimal solution of problem $P_1(a)$, $a \in A$, and let V^3 be the set of basic units that are uniquely assigned to different territories in the optimal solution to problems $P_1(1)$ and $P_1(2)$. Then the set of split units is given by $Y^{\text{split}} = V^1 \cup V^2 \cup V^3$.

The split resolution problem consists of deciding to which territory or centroid each split unit must be assigned. In previous works, that is, districting problems dealing with single-activity balancing constraints only, this is achieved by aiming at the goal of keeping the territories as balanced as possible. To do this task, a merit function that penalizes the deviation from the activity target is defined.

In our case, the choice of this merit function is different since we are dealing with the multiple-activity balancing constraints and the connectivity constraints. Given the connectivity constraints are harder to meet, the following criteria are taken into account for iteratively assigning each split node.

1. Repair connectivity: We give priority to split nodes whose insertion into a given disconnected territory repairs it.

2. Balancing: If only unfeasible choices remain, assign the node to territory with the lowest violation of the balancing constraints.

3. Dispersion minimization: Among all possible feasible decisions, that is, assignment choices where adding a split node to a territory renders a feasible solution with respect to the balancing constraints, assign the node to the territory with the smallest increment in the objective function.

The split resolution heuristic is displayed in Procedure 2. The algorithm takes as an input Y^{split}, the set of split BUs of the associated problem P_1. A grouping of the BUs given by $X = (X_1, \ldots, X_p)$ is also known. Clearly, if $Y^{\text{split}} \neq \emptyset$, this grouping X is not a valid p-partition because it contains the multiple territory assignments of the split units in Y^{split}. Naturally, if $Y^{\text{split}} = \emptyset$, X is a valid p-partition. Let T_i be the set of (multiple) territories associated with unit $i \in Y^{\text{split}}$. The idea of the algorithm is to take each node in Y^{split}, and somehow decide a unique territory assignment for it by an appropriate choice of a territory in T_i. In the first part of the algorithm, we first attempt to repair connectivity, if possible. Procedure *repairConnectivity(i, T_i)* takes i and T_i as input and returns a territory $k \in T_i$ only if assigning i to k repairs connectivity. Otherwise, it returns NULL. Function *assign(i, k)* assigns unit i to territory k, including the corresponding update of solution X. Procedure *findBest()* finds the "best" remaining node in Y^{split} as follows. First it chooses the node whose best territory assignment causes the lowest increment in infeasibility with respect to the balancing constraints. In case of ties, it chooses the node whose best assignment causes the

lowest increment in the objective function. If it still tied, it chooses a node by lexicographic order. This function also returns the best corresponding territory assignment.

Procedure 2 Split-Resolution( )

Input: A TDP instance; Y^{split}:= Set of split BUs

Output: $X = (X_1, \ldots, X_p)$ A p-partition solution to TDP

```
for ( i ∈ Y^split ) do
    k ← repairConnectivity(i, T_i)
    if ( k ≠ NULL ) then
        assign(i, k)
        Y^split ← Y^split \ {i}
    end if
end for
while ( Y^split ≠ ∅ ) do
    (i, k) ← findBest()
    assign(i, k)
    Y^split ← Y^split \ {i}
end while
return X
```

Naturally, being a heuristic, there is no theoretical warranty that a feasible solution is obtained after the split resolution has taken place. When the infeasibility is due to disconnectivity, the algorithm does not update and proceeds to the start of the following iteration. In practice, this does not occur too often. When the infeasibility is due to a violation of the balancing constraints, our procedure is enhanced by a local search phase that attempts to improve the solution in terms of its quality or recover feasibility if it applies. This is described next.

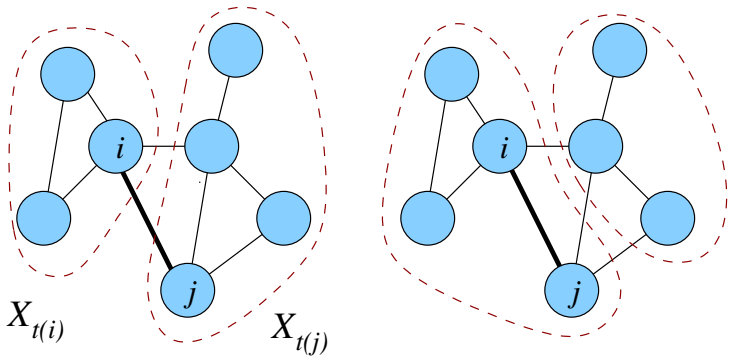
Of course, being a heuristic, there is no theoretical warranty that a feasible solution is obtained after the split resolution has taken place. When the infeasibility is due to disconnectivity, the algorithm does not do any updating, and proceeds to the start of the following iteration. In practice this does not occur too often. When the infeasibility is due to violation of the balancing constraints, our procedure is enhanced by a local search phase that attempts either to improve the solution in terms of its quality or to recover feasibility if it applies. This is described next.

4.3 Local Search Phase

A solution obtained by the mechanism described in the previous section may not be feasible in terms of the balancing constraints. Therefore a local search phase is developed for attempting to improve the solutions taking as an input a solution that satisfies the connectivity constraints.

This improvement can be made in two ways: (i) improve the objective function value of a feasible solution or (ii) improve the feasibility violation of an infeasible solution. We now describe the main components of the local search procedure.

Neighborhood topology: A neighborhood $N(X)$ consists of all solutions reachable from p-partition X by moving a basic unit j from its current territory $t(j)$ to a neighbor territory $t(i)$, where i is the corresponding basic unit in territory $t(i)$ adjacent to j, without creating a non-contiguous solution. Such a move is denoted by $move(j, i)$ and is illustrated in Figure 1, where *move(j, i)* is represented by arc (j, i) (depicted in bold). Note that $move(j, i)$ is allowed only if $X_{t(i)} \cup \{j\}$ is connected (which is always the case if arc (j, i) exists), and $X_{t(j)} \setminus \{j\}$ remains connected. In practice an additional stopping criterion, such as *limit_moves*, is added to avoid performing the search for a relatively large amount of time. Thus the procedure stops as soon as a local optimal is found or the number of moves exceeds *limit_moves*. An aspiration criterion that allows to override *limit_moves* if the current solution is within ϵ of the best known solution is into effect. This ϵ is set to 0.01. For the purposes of our work, we have set this parameter to a very large number (10,000) such that our local search always stops with a local optimal solution, that is, it never stops by reaching *limit_moves* number of moves.



(a) Before move (b) After move

Figure 1: Move definition.

Merit function: Now, the next step is to have a merit function that measures the attractiveness of a given neighbor solution, helping decide whether to make a move. Typically, this is done employing a merit function that takes into account the objective function. However, in this particular case, since a solution may not be feasible, we define a merit function as a convex combination of two terms, one measuring objective function quality and the other measuring the degree of constraint violation.

Specifically, for a given p-partition $X = (X_1, \ldots, X_p)$, with corresponding medians $(c(1), \ldots, c(k))$,

its merit function $\varphi(X)$ is given by

$$\varphi(X) = \lambda F(X) + (1-\lambda)G(X) \tag{18}$$

where

$$F(X) = \left(\frac{1}{d_{\max}}\right) \sum_{k=1,\ldots,p} \left\{ \sum_{j \in X_k} d_{c(k),j} \right\} \quad \text{and}$$

$$G(X) = \sum_{k=1}^{p} \sum_{a \in A} g^a(X_k),$$

with $d_{\max} = \max_{i,j \in V}\{d_{ij}\}$ being a normalization factor, and $g^a(X_k) = (1/\mu^a)\max\{w^a(X_k) - (1+\tau^a)\mu^a, (1-\tau^a)\mu^a - w^a(X_k), 0\}$, being the sum of the relative infeasibilities of the balancing constraints. In case of ties, a lexicographic rule is in effect, that is, making the move with the lowest node index first, and if moves have the same lowest node index, choosing the move with the lowest second node index first.

5 Computational experiments

This section reports experimental results obtained with the proposed location-allocation-improvement heuristic. The proposed method was implemented in the C# programming language and compiled with the Mono JIT compiler version 6.8.0. The code and data sets are publicly available for research purposes from the authors upon request. All of the experiments were run in a workstation with an Intel Core i7–4790K processor at 4.00 GHz and 32 GB DDR3 1600 MHz RAM, running MS Windows 10 Pro. The ILOG CPLEX 12.5 callable library was used for solving the linear programming subproblem.

5.1 Experimental Setting

For the experiments we used the data base from Ríos-Mercado and Fernández [32]. These are randomly generated instances based on real-world data. Data set DS is considered for experimentation. We now describe next how these instances were generated. This set contains BU weights generated from a uniform distribution. This data set has been used as reference in several works on commercial districting. In summary, each instance is a planar graph with the BUs (nodes) distributed in the $[1,500] \times [1,500]$ square, where the weights w^a are uniformly distributed in the [1,4] and [1,12] range for number of customers and product demand, respectively. Unless otherwise noted, for all of the instances in the DS data set, we use a tolerance level $\tau^a = 0.05$, $a \in A$. Recall that τ^a measures the allowable relative deviation from the target average size μ^a for activity $a \in A$.

Hence, a value of $\tau^a = 0.05$ implies that instances are tightly constrained in all activities and therefore the problem is more difficult to solve than instances that use a larger value of τ^a. For each combination of $(n, p) \in \{500, 1000, 2000\} \times \{20, 40, 60\}$, 20 different instances were generated. Thus, considering the number of basic units, the number of territories and the tolerance factor, there is a total of 9 combinations and for each one of those combinations 20 instances which gives a total of 180 generated instances.

Throughout the evaluation, the algorithm is run with the iteration limit set to $m = 40$ and the merit function weight parameter on the local search set to $\lambda = 0.9, 0.8, 0.7$ for the 20-, 40-, and 60-territory instances, respectively. These values were observed to deliver better results on preliminary full experimentation for fine-tuning these algorithmic parameters. Showing the full fine-tuning of these parameters is out of the scope of this paper. Neverthess, it is interesting noting that as we increase p, best results are obtained when λ decreases. This stems form the fact that larger values of p imply more balancing constraints (4)-(5) to satisfy, and therefore more weight is needed in the infeasibility component of the merit function (18). This is achieved by lowering the value of λ. Furthermore, given that λ does not depend on n, a good rule of thumb for choosing λ as a function of p would be $\lambda = 1 - p/200$ for p values between 10 and 100.

5.2 Evaluating Algorithm Running Time

In this experiment, we evaluate the algorithm running time as a function of instance size dimension parameters, namely, number of BUs (n) and number of territories (p).

Figure 2 shows the average results for each combination of n and p over 20-instance groups. The horizontal axis indicates the value of p, while the vertical axis displays the running time (CPU seconds). The different p values are indicated by different colors, as indicated in the small box, where Tn means the instances of size n BUs. Table 1 displays a full summary of worst, average, and best running times for each combination of n and p. As can be seen from these results, the algorithm running time is well behaved as it exhibits an almost linear increase as p gets large. Another interesting observation is that the growth of p affects more the running time than the growth of n. This observation indicates that the number of split nodes plays an important role in the time complexity. This is further analyzed in Section 5.4. As expected, fo each value of n, the more time-consuming instances solved were the ones with $p = 60$. Nevertheless, the 2000-node instances take around 20 minutes in the worst case; thus, these are very reasonable running times considering the districting decisions are taken every 3-4 months.

To gain more insight into the algorithm's computational effort, Table 2 shows the average proportion of time spent in each of three phases, for each combination of n and p. As can be seen, the allocation phase, which is the heart of the algorithm is the more time-consuming phase. This phase involves solving the LP subproblem and then applying the split resolution heuristic at every

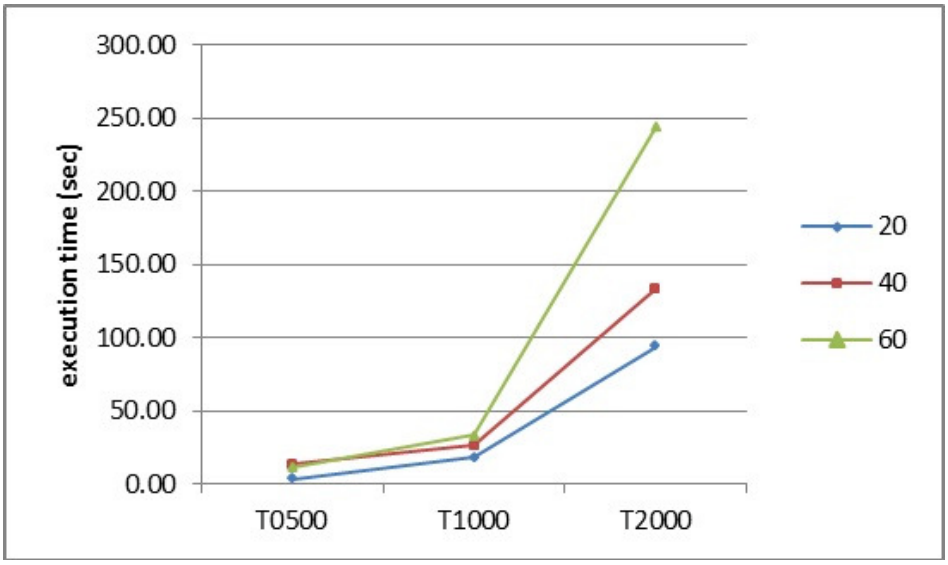


Figure 2: Average running time as a function of n and p.

Table 1: Time performance of heuristic.

		Running times (CPU seconds)		
		$n = 500$	$n = 1000$	$n = 2000$
$p = 20$	Best	1.300	8.202	26.382
	Average	3.464	18.393	93.836
	Worst	7.455	43.430	219.566
$p = 40$	Best	2.100	9.163	23.896
	Average	13.220	26.065	133.244
	Worst	49.706	106.298	443.953
$p = 60$	Best	3.174	10.529	35.670
	Average	11.651	33.164	244.371
	Worst	28.105	86.473	790.553

iteration . The location phase, as expected, takes less time as it is basically a sorting procedure. Note the results are very consistent regardless the dimension of the problem.

5.3 Assessing of Impact of Local Search Phase

One of the important contributions of our heuristic is the incorporation of the local search phase. As stated before, this phase aims at helping to improve both the feasibility and objective function value of the solution found from the previous allocation phase. Thus, in this experiment, we investigate the impact of the local search phase on solution quality. To this end, we compute, in each iteration, the relative improvement of the local search phase with respect to the solution from the allocation phase, and obtain the average over all iterations needed to converge.

Table 2: Proportion of running time employed by each phase.

		Phase		
	n	Location	Allocation	Local Search
p=20	500	7.959%	76.952%	15.089%
	1000	7.148%	77.630%	15.222%
	2000	7.109%	77.663%	15.228%
p=40	500	11.628%	73.884%	14.487%
	1000	10.815%	74.564%	14.620%
	2000	11.440%	74.042%	14.518%
p=60	500	13.929%	71.961%	14.110%
	1000	13.441%	72.369%	14.190%
	2000	13.236%	72.540%	14.224%

Table 3: Assessment of local search phase.

		Average relative improvement by LS (%)		
		$n = 500$	$n = 1000$	$n = 2000$
p=20	Minimum	45.60	47.99	50.63
	Average	47.87	51.58	57.21
	Maximum	49.70	55.92	62.76
p=40	Minimum	46.09	51.85	58.73
	Average	48.72	55.77	64.10
	Maximum	50.23	58.81	71.33
p=60	Minimum	48.70	54.85	63.44
	Average	50.18	60.19	70.38
	Maximum	51.77	63.98	76.07

Table 3 shows the average results of this experiment for each combination of n and p. The tables displays the average relative improvement obtained by the local search (%). As we can see from the table, this is indeed significant. For the 500-node instances, we observe improvements ranging from 45 to 51%. These averages get better as the size of the instance grows. For the 1000-node instances, the improvements range from 48 to 63%. For the 2000-node instances, the observe improvements of up to 76%. This clearly shows the huge positive impact of the local search phase.

5.4 Assessing the Effect of the Split Nodes

A critical issue to investigate is the effect and impact that the growth of the number of split units has in algorithmic performance. Clearly, the number of split units is related to the computational effort in the allocation phase. Recall, that for TDPs with single-activity balancing constraints, there is a theoretical result that bounds the number of split units by $p - 1$. However, for multiple-activity balancing constraints, such as the one addressed in this paper, this result no longer holds. Therefore, it becomes very important to investigate and empirically assess the growth of the number of split units and its effect on algorithm running time. Now, the number of split BUs comes from

three possible sources, the splits from each of the subproblems associated to the linear programming subproblems parameterized on each activity $a \in A$, two in this case, and the integer variables that link each node to different centroids in each of the subproblems. We know, that for each of the former, the number of split units is bound by $p - 1$; however, for the latter, there is no known bound. This is precisely the motivation for empirically study this growth.

Table 4: Number of splits units.

		$n = 500$	$n = 1000$	$n = 2000$
p=20	Best	23.00	28.00	44.00
	Average	27.70	37.45	56.10
	Worst	36.00	50.00	85.00
p=40	Best	45.00	56.00	75.00
	Average	49.05	60.90	90.25
	Worst	53.00	68.00	114.00
p=60	Best	62.00	79.00	102.00
	Average	67.35	83.45	116.70
	Worst	73.00	90.00	142.00

Table 4 displays the results for each combination of n and p. The three rows for each fixed value of p show the best, average, and worst number of splits observed. The last three columns indicate the number of nodes associated to the test instances. First, we observe, for the 500-node instances, that in all cases, the average number of split units found is well below the $2(p - 1)$ threshold. Furthermore, the worst possible value (largest one) is also below this threshold. Now, for the 1000-node instances, this number grows but not by much. The average number of split units is still observed below this threshold, and the worst value is also below this threshold. Finally, for the 2000-node instances, the number of split units grows over the threshold, but the growth is still well behaved as the growth is relatively modest. For instance, the largest group of instances (2000×60) has an average numebr of split units below the threshold. We conclude that, the number of split units observed is kept relatively low keeping the problem tractable.

5.5 Feasibility Analysis

Another important issue to investigate is the feasibility issue. Recall that the split resolution heuristic, in the allocation phase, attempts to resolve the splits by considering the satisfaction of both connectivity and multiple-activity balancing constraints. However, due to its heuristic nature, feasibility is not guaranteed. Furthermore, the ensuing local search phase does not guarantee feasibility, although this phase is expected to reduce or eliminate infeasibility.

To this end, in this experiment, when running the algorithm, we measure the degree of infeasibility by computing the number of feasible solutions found after every iteration and dividing this by the total number of iterations needed by the algorithm to converge. In other words, we compute

the fraction of the proportion of times per iteration that a feasible solution was found.

Table 5 shows the results for each combination of n and p. The figure in each cell can be seen as an empirical estimation that a solution found in a single iteration is feasible.

Table 5: Feasibility success per iteration.

	$n = 500$	$n = 1000$	$n = 2000$
$p = 20$	94.81%	91.66%	89.33%
$p = 40$	94.93%	91.67%	89.28%
$p = 60$	95.38%	91.67%	89.27%

Now, given that it was found that the overall algorithm takes in the order of 50 to 60 iterations at least to stop, the probability of success of the overall execution of the algorithm is practically 1. In other words, the probability of the algorithm delivering an infeasible solution is equal to the probability of finding an infeasible solution in each iteration. For example, let q be the success rate per iteration and l be the number of total iterations employed by the algorithm. Thus the probability that the algorithm fails in finding a feasible solution is given by $(1 - q)^l$. Taking into account that it was empirically found that $q > 0.89$ and $l \geq 40$, the failure probability is practically zero. This can be confirmed in Table 6, that shows the percentage of times (success rate) that the overall algorithm delivered a feasible solution at termination. As we can see, of all instances tested under all different combinations, none were found infeasible.

Table 6: Feasibility success per total execution.

	$n = 500$	$n = 1000$	$n = 2000$
$p = 20$	100%	100%	100%
$p = 40$	100%	100%	100%
$p = 60$	100%	100%	100%

5.6 Solving Harder Instances

To further test the algorithm on more challenging scenarios, additional instances were generated as follows. For the first experiment, with the aim of observing the behavior as a function of n, additional instances with 3000, 4000, and 5000 BUs were generated for each combination of $p \in \{20, 40, 60\}$. For the second experiment, additional instances with 10, 30, 50, 70, 80, 90, and 100 territories were generated for each combination of $n \in \{500, 1000, 2000\}$.

Figure 3 and Figure 4 display the results of running time as a function of n and p, respectively In the former, the running time is plotted as a function of n (shown in the horizontal axis). In the latter, the same curves are plotted but as a function of p.

As we can see from Figure 3, running time increases as the number of nodes grows beyond 2000. Nevertheless, even for 5000 nodes instances, running time is under 20 minutes, very manageable.

From Figure 4, we can see a sharp increase after 60-70 territories, particularly for the largest subset ($n = 2000$). Still, the worst case is on the order of 700 seconds. In real-world, typical instances usually have less than 60 territories.

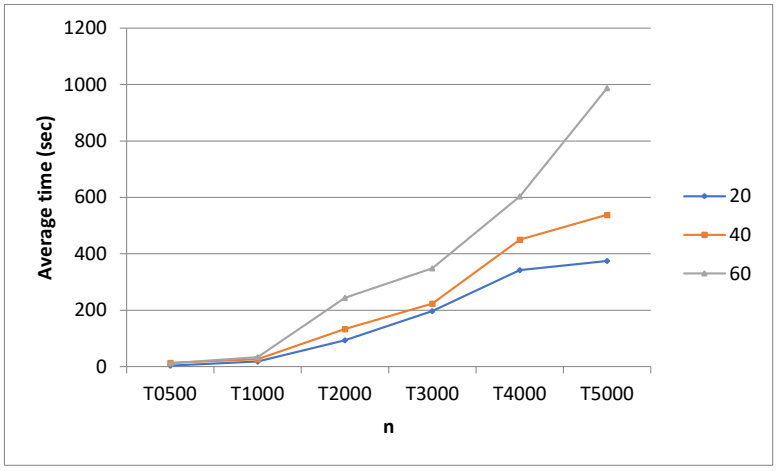


Figure 3: Running time versus number of BUs for challenging instances.

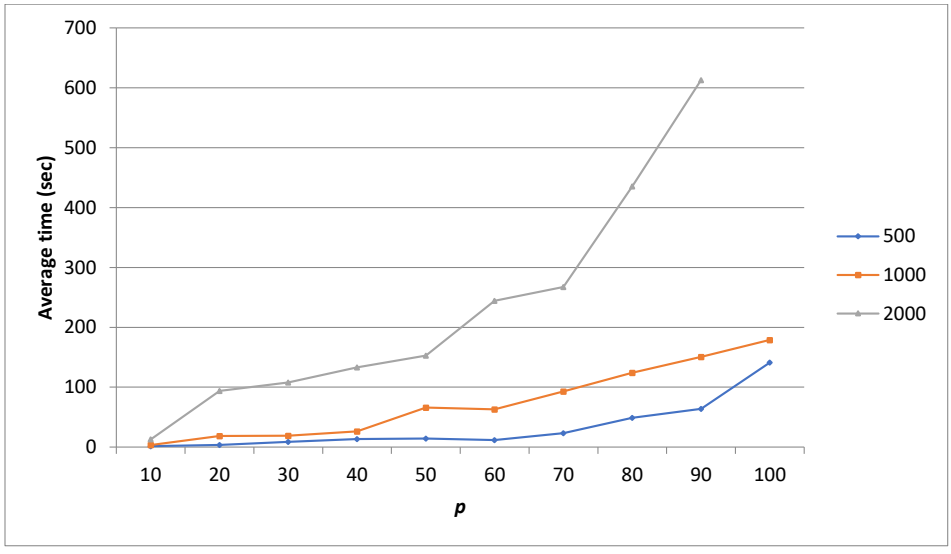


Figure 4: Execution time versus number of territories for challenging instances.

As we have seen from the previous subsection, the algorithm has a feasibility success rate of 100% in all instance tested. In each instance, we are assuming a tolerance paramater of $\tau^a = 0.05$, which is a very acceptable tolerance value by any means.

Table 7: Feasibility success rate as a function of the tolerance parameter.

		$n = 500$	$n = 1000$	$n = 2000$
$\tau^a = 0.05$	$p = 20$	100.0	100.0	100.0
	$p = 40$	100.0	100.0	100.0
	$p = 60$	100.0	100.0	100.0
$\tau^a = 0.04$	$p = 20$	100.0	100.0	100.0
	$p = 40$	100.0	100.0	100.0
	$p = 60$	100.0	100.0	100.0
$\tau^a = 0.03$	$p = 20$	100.0	100.0	100.0
	$p = 40$	100.0	100.0	100.0
	$p = 60$	100.0	100.0	100.0
$\tau^a = 0.02$	$p = 20$	93.3	90.0	40.0
	$p = 40$	100.0	93.3	53.3
	$p = 60$	100.0	100.0	83.3
$\tau^a = 0.01$	$p = 20$	0.0	0.0	0.0
	$p = 40$	3.3	6.7	6.7
	$p = 60$	6.7	10.0	10.0

In previous works with related models, tolerance levels of 0.10, 0.20 and even 0.30 were assumed. Thus, in the next experiment, we push the feasibility issue further and setup a new set of challenging instances by solving the same sets for tolerance values of 0.04, 0.03, 0.02, and 0.01.

Table 7 shows the results for each combination of n, p and τ^a. The cells indicate the average feasibility success rate over groups of 20 instances per combination.

As can be seen from the table, the algorithm is very successful for τ^a values of 0.04 and 0.03. When crossing to the 0.02 level, the algorithm still manages to solve most of the instances, but not all. However, when going to $\tau^a = 0.01$, the algorithm struggles in finding feasible solutions. We may conclude that our algorithm is highly reliable for τ^a levels as low as 0.03, which is remarkable.

5.7 Assessing the Merit of the Location-Allocation Construction

Existing construction heuristics for variants of districting problems that are subject to balance constraints struggle to find feasible designs as they cannot meet all of the balancing constraints. This issue has been observed in many works [11, 32, 31]. In those works, the local search plays a fundamental role in their success. Now, the attractive feature of the location-allocation idea in districting is that the relaxed allocation subproblem $P_1(a)$ is always feasible with respect to the balance constraints. Then, the theoretical result that bounds the number of fractional variables by a relatively small value that depends on p (not on n) can be exploited for delivering feasible

solutions almost all the time. Thus, the local search is an enhancement to the location-allocation heuristic. When the local search phase is applied to traditional constructive heuristics, a significant effort is made to bring the solution to the feasible space and then attempt to improve it. In contrast, under the location-allocation idea, when the local search is applied to a feasible or almost feasible solution, less computational effort is needed.

To explore and evaluate this effect, the following experiment is carried out. We implemented a simple construction heuristic based on the idea from Ríos-Mercado and Escalante [31] that builds the territories simultaneously. It has been documented that construction heuristics that build territories simultaneously work better than those that build territories one at a time. For these tests, we generated a total of 168 instances distributed as follows: 2 different values of n (4,000 and 5,000), 3 different values of p (15, 30, and 40), and 2 different values of tau (0.03 and 0.05), and, for each of these set combinations, 14 instances. A time limit of 3600 seconds was used, but in every run, neither heuristic reached the time limit but stopped by local optimality of the local search. We run the proposed heuristic and the simple construction heuristic under both with and without the local search. The comparison of the heuristics with no local search has the goal of assessing the feasibility issue referred above. In the following figures, the term "ORIG" and "SIMP" refer to the proposed location-allocation and simple construction heuristic, respectively. The suffixes "_LS" and "_noLS" indicate whether the local search is applied or not, respectively.

We first show the results in terms of feasibility. Figures 5 and 6 show the number of feasible solutions found by each method for data sets under $\tau = 0.05$ and 0.03, respectively.

First, the comparison of methods with no local search can bee seen in the second and fourth columns in each chart. Under the location-allocation scheme, 47, 21, and 4 feasible solutions were obtained (out of 56 for the $p = 10$, $p = 30$, and $p = 40$ instances, respectively). This contrasts significantly with the simple heuristic that obtained zero feasible instances in each case. This makes the case for the location-allocation scheme. When comparing the methods incorporating the local search phase in both methods (first and third columns in each chart), the proposed method and the simple heuristic obtained all feasible solutions in the $p = 10$ and $p = 30$ instances. For the p=40 instances, the proposed method obtained more feasible solutions than the simple heuristic.

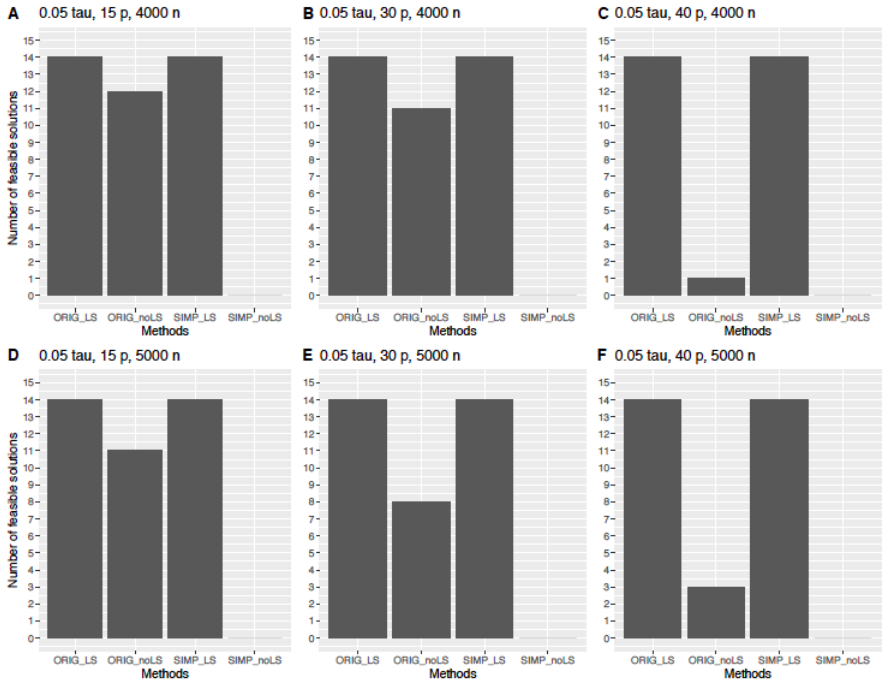


Figure 5: Evaluation of feasibility for instances with $\tau = 0.05$.

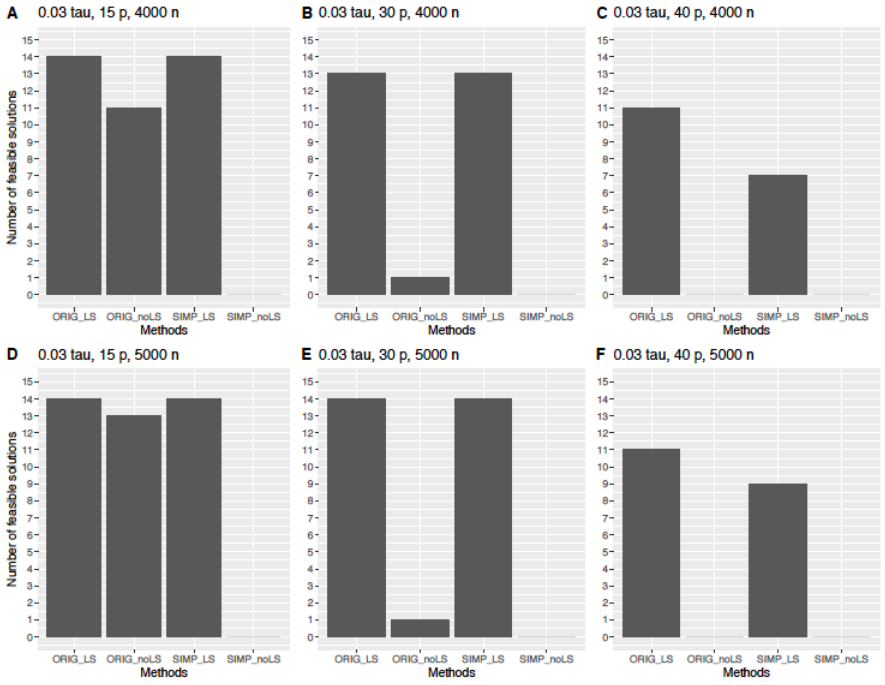


Figure 6: Evaluation of feasibility for instances with $\tau = 0.03$.

We now present the results in terms of solution quality. Figures 7 and 8 below show the average objective function found by each method for data sets under $\tau = 0.05$ and 0.03, respectively.

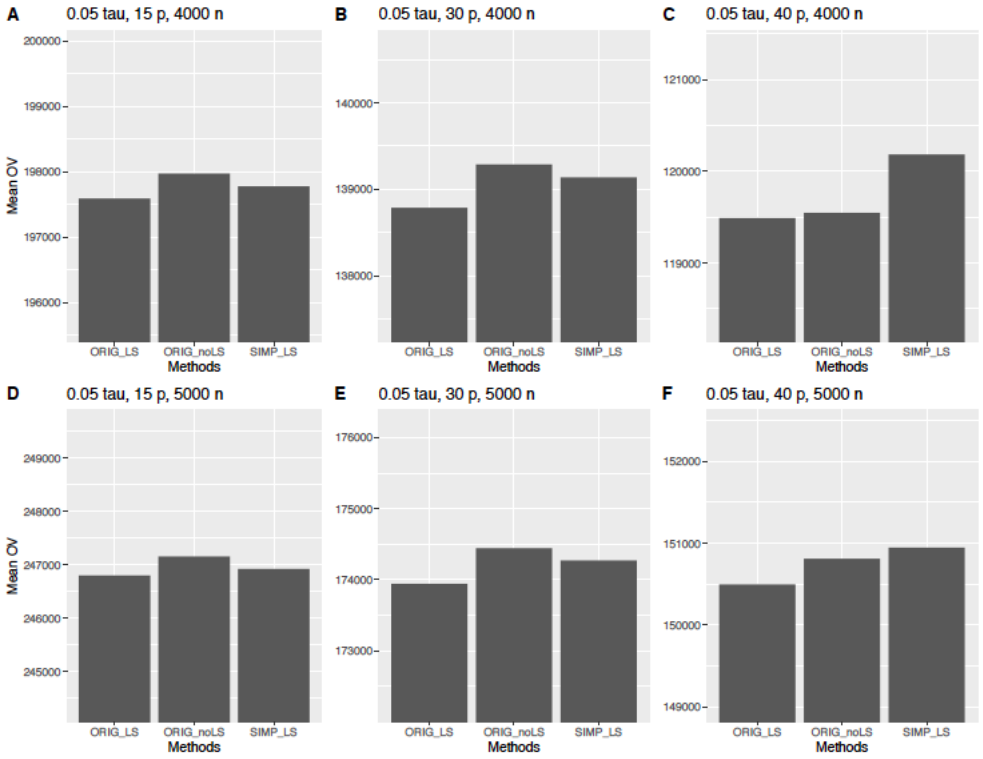


Figure 7: Evaluation of objective function value for instances with $\tau = 0.05$.

When we look at the complete methods, it can be clearly seen that the proposed method obtains solutions of better quality than the other heuristic. This difference is even more dramatic if one considers that all of the values obtained from the proposed method come from feasible instances, whereas not all instances are feasible when the other heuristic was used. In summary, the particular properties and features of the location-allocation idea are intelligently exploited such that the heuristic finds feasible designs most of the times. When this construction is followed by the local search, those few infeasible instances are repaired, and those feasible instances are enhanced in terms of solution quality. Other ways of constructing feasible solutions are not as successful, such that the local search spends more effort trying to achieve optimality, and still fails in many cases.

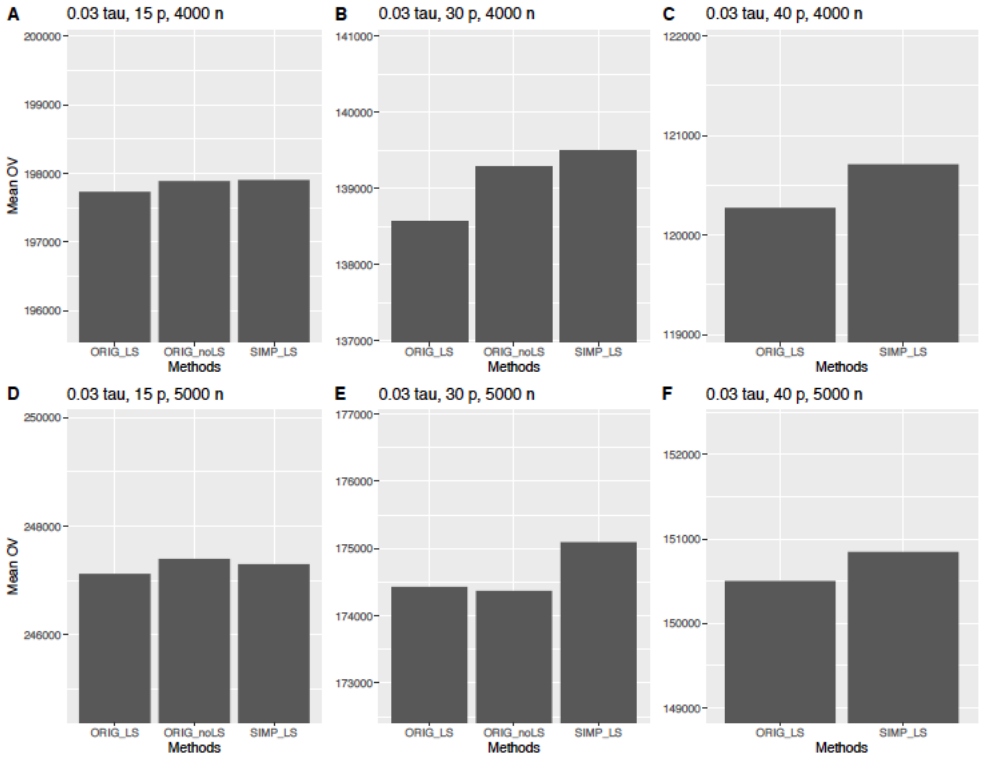


Figure 8: Evaluation of objective function value for instances with $\tau = 0.03$.

5.8 Comparisons with Existing Work

As stated before, to the best of our knowledge, the only existing heuristic methods for a districting problem with these features (e.g., multiple-activity balancing constraints, contiguity constraints and dispersion minimization measured by a p-median type of function) are due to Salazar-Aguilar et al. [38] and Gliesch and Ritt [15]. In this section we present a comparison of our method to each of these methods.

Comparison with the Divide-and-Conquer Heuristic [38]

As noted by Salazar-Aguilar et al. [38], their method, based on a divide-and-conquer approach struggles in finding feasible solutions to many problem instances. The reason behind that is that, when dividing the problem into smaller problems, an adjustment on the tolerance levels must be made and this is not a straigthforward matter. Setting the value too small makes the subproblems unfeasible, leaving it too large, makes the main problem unfeasible. In addition, that heuristic is also time consuming.

Nevertheless, we obtain the code of that heuristic from the authors and test it in the same

platform as ours on some instances. We use instances with 1000 and 2000 BUs and 50 territories with tolerance level of 0.10 (10 replicates per combination). Setting the tolerance to 0.05 makes the existing algorithm look worse in terms of feasibility.

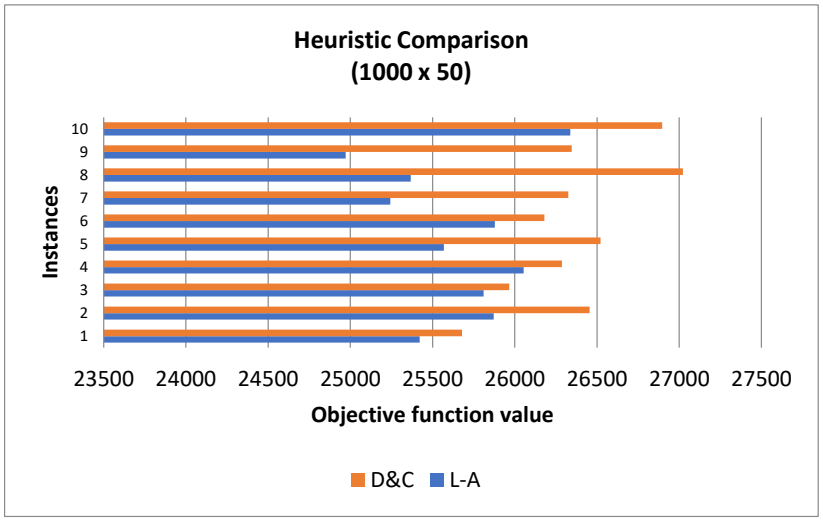


Figure 9: Comparison of D&C and L-A on 1000-node instances.

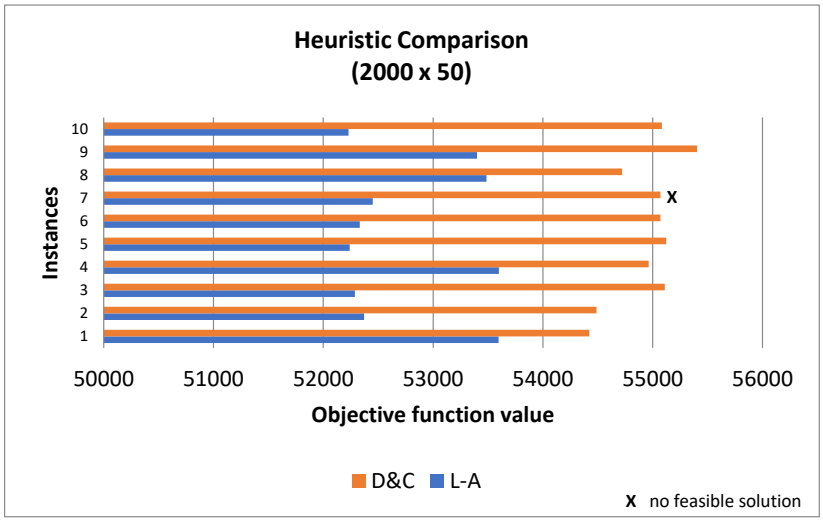


Figure 10: Comparison of D&C and L-A on 2000-node instances.

Figure 9 and Figure 10 show the comparison between the two heuristics on the 1000- and 2000-node instances, respectively. The horizontal axis indicates the individual instance tested, and the vertical axis shows the objective function value. The existing algorithm is labeled as D&C and our proposed location-allocation heuristic is labeled as L-A. An infeasible solution is indicated by an "X" (as found by D&C on instance 7 of the 2000-node set).

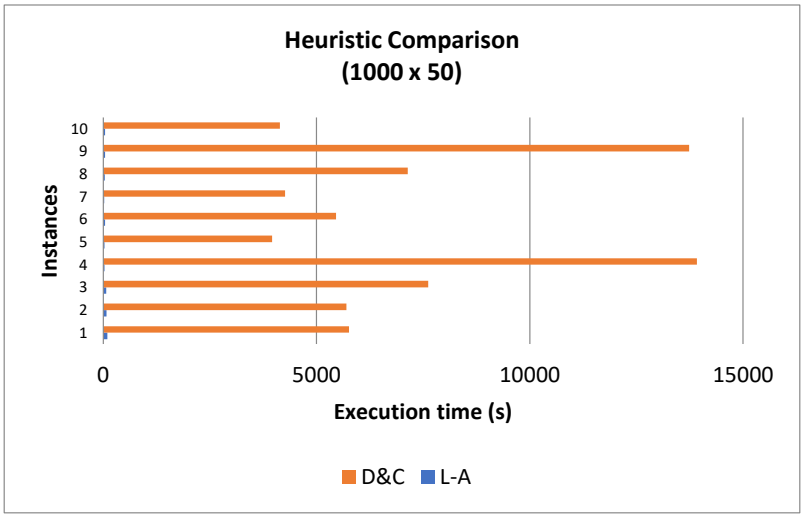


Figure 11: Time comparison of heuristics on 1000-node instances.

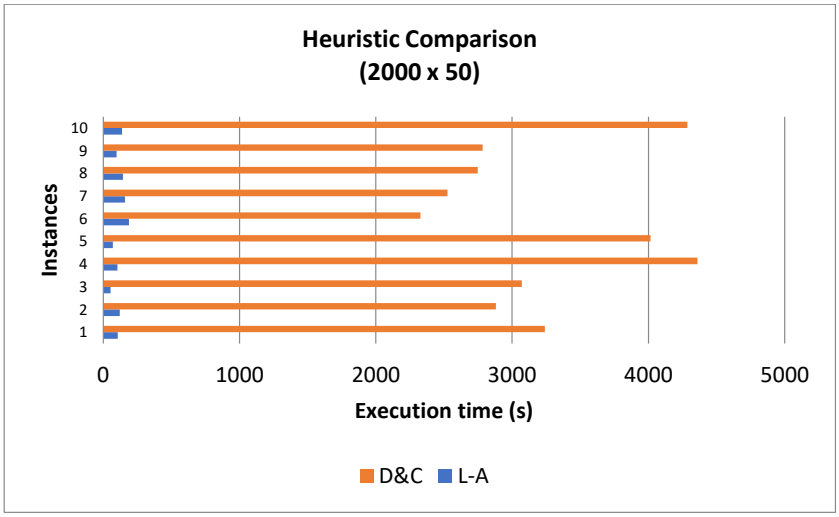


Figure 12: Time comparison of heuristics on 2000-node instances.

In addition, Table 8 summarizes the results by presenting a comparison in terms of the relative improvement shown by L-A over D&C. The first and and third columns indicate the instance number for the 1000×50 and 2000×50 data sets, respectively. The second and fourth columns display the relative improvement found by L-A with respecto to D&C for the 1000×50 and 2000×50 data sets, respectively. The final row shows the average relative improvements.

As can be seen, our heuristic clearly outperforms the existing approach in all instances tested. This superiority is exhibited even at a larger magnitude in the larger instances. In top of that, the existing method failed to find feasibility in one instance (of the total of 20 tested).

Table 8: Relative improvement of L-A over D&C.

1000×50 Set		2000×50 Set	
Instance	RI (%)	Instance	RI (%)
1	1.00	1	1.52
2	2.21	2	3.88
3	0.60	3	5.12
4	0.88	4	2.48
5	3.59	5	5.23
6	1.15	6	4.97
7	4.11	7	4.76
8	6.12	8	2.26
9	5.22	9	3.62
10	2.08	10	5.18
Average	2.70	Average	3.90

The superiority of our method is even more remarkable when we look at the computational effort. Figure 11 and Figure 12 show the comparison between the two heuristics on the 1000- and 2000-node instances, respectively, in terms of CPU time (seconds). We had seen before, that for this size of instances, our method takes on average less than 100 seconds. The existing approach takes average running times 7000 and 3000 seconds.

Comparison with Tabu Search [15]

We now make a comparison with the tabu search method of Gliesch and Ritt [15], referred to as TS-GR. To establish a fair comparison, it is important to note the following. The first point we would like to point out is the different nature of both heuristics. First, our heuristic is very fast, falling into the category of quick heuristics. The location and allocation phases are solved rather quickly. The number of iterations needed to converge is relatively small. Perhaps the most time consuming is the local search, but even then, because it is a straight hill-climbing strategy stopping at the first local optima, and because it starts from a feasible (or very close to) solution, its computational effort is rather modest. On the other hand, the TS-GR heuristic is a more time-consuming heuristic that employs rather sophisticated techniques to attempt to escape from local optimality, spending as much effort as needed.

With this in mind, we carry out the following experiments. First, we apply both heuristics independently to several subsets of instances, allowing as stopping criteria the same amount of time (30 minutes). We use a data set comprised of instances of 2500, 5,000, and 10,000 basic units with combination values of $p \in \{12, 48\}$ and $\tau \in \{0.05, 0.10\}$ as displayed in the table below. For each tested combination, 12 instances were solved, so averages are taken over 12 instances.

Table 9 displays the results. The first and second columns show the combination of p and τ used. The third column indicates the value of n. The fourth and fifth columns show the number of

feasible solutions found (out of 12) by the TS-GR and our proposed L-A heuristic, respectively. The sixth and seventh columns display the average CPU time employed by each heuristic. Naturally, the TS reached the limit of 1800 seconds every time. The eighth column shows the average relative improvement in solution quality of L-A over TS-GR. A negative value indicates TS-GR was better.

Table 9: Comparison of TS-GR and Location-Allocation.

			Feasibility		Time		
p	τ	n	TS-GR	L-A	TS-GR	L-A	ARI (%)
12	0.05	2500	12	12	1800.00	130.61	-1.82
		5000	12	12	1800.00	392.80	-0.99
		10000	12	12	1800.00	474.37	-0.30
48	0.10	2500	12	12	1800.00	121.64	-0.27
		5000	12	12	1800.00	409.57	-0.12
		10000	12	12	1800.00	541.42	-0.08

In summary, both heuristics turned out to be very competitive. All found feasible instances with very similar objective function quality. The first observation is that, on average, the solution quality delivered by TS-GR was better than the LA heuristic. However, the difference in almost all cases is minimal. For the case with $p = 12$, this difference in solution quality was less than 1% for the 5000- and 10000-node instances. For the case with $p = 48$, which resembles more practical settings, the average solution quality difference is less than 0.3%. In terms of solution time, the existing tabu search employed all of the 1800 seconds set forth in each run. The existing heuristic employed significantly less time, that is, found a solution of around the same quality faster.

Now, in the second experiment, we try to make a case for the location-allocation idea. If we recognize that the local search phase in our procedure can be seen as an improvement phase, and the tabu search algorithm can be seen as a local search heuristic, both ideas can be used collaboratively. To this end, we apply our algorithm and then the obtained solution was fed into the TS-GR local search phase of the other heuristic (omitting their construction phase). We refer to this as LA-TS. We use a data set comprised of instances of 2500, 5,000, and 10,000 basic units with combination values of $p \in \{12, 48\}$ and $\tau \in \{0.05, 0.10\}$ as displayed in the table below. For each tested combination, 12 instances were solved, so averages are taken over 12 instances.

Table 10 displays the results. The first and second columns show the combination of p and τ used. The third column indicates the value of n. The fourth and fifth columns show the number of feasible solutions found (out of 12) by the TS-GR and the LA-TS heuristics, respectively. The sixth and seventh columns display the number of times each heuristic found the best solution. The eighth column shows the average relative improvement in solution quality of LA-TS over TS-GR. A negative value indicates TS-GR was better than LA-TS.

The first observation is that both procedures find feasible instances in 100% of the cases. The solution quality is very similar; both approaches remain very competitive. In the $p = 48$ case,

Table 10: Comparison of TS-GR and Location-Allocation plus Tabu Search.

			Feasibility		Number of best		
p	τ	n	TS-GR	LA-TS	TS-GR	LA-TS	ARI (%)
12	0.05	2500	12	12	3	9	0.00
		5000	12	12	5	7	0.03
		10000	12	12	5	7	0.03
48	0.10	2500	12	12	7	5	-0.29
		5000	12	12	6	6	-0.21
		10000	12	12	6	6	0.07

the number of best solutions found is about the same, 19-17 in favor of TS-GR. However, when analyzing the $p = 12$ case, the combined LA-GR finds more best solutions than its counterpart by a 23-13 score.

What we found is that in most of the cases the solution obtained this way was better than the solution delivered by the other heuristic running as a standalone version. This makes a case for the location-allocation idea proposed in our work.

6 Conclusions

In this paper, a heuristic methodology based on a location-allocation scheme was developed to address a real-world territory design problem from a beverage distribution firm. The TDP includes both connectivity and multiple-activity balancing constraints, intending to minimize a p-median objective function. The algorithm is enhanced by incorporating a local search phase that improves solution quality as observed in the computational testings. The proposed heuristic is very efficient and reliable, by intelligently exploiting the mathematical structure of the problem to yield feasible designs all the time. When compared to existing work, the proposed method clearly outperforms the divide-and-conquer heuristic and is very competitive with respect to the tabu search heuristic. A closer look at the empirical work reveals that the location-allocation phase is highly successful in delivering solutions that satisfy or almost satisfy the balancing constraints. This property is a real advantage over traditional construction heuristics that struggle in finding feasible designs. Consequently, the local search phase converges faster to a local optimal than traditional methods.

The algorithm was proven to handle very well the multiple-activity balancing constraints and the connectivity constraints. The methodology can be extended to handle more complex models. For any TDP with p-median objective function subject to additional side constraints, we could relax the additional constraints and then solve a linear programming subproblem with the balance constraints only very efficiently. Then a split resolution heuristic that intelligently puts back the previously relaxed constraints can be developed by adequately exploiting problem structure. Naturally, local search is strongly encouraged.

As shown in the empirical tests, the local search implemented in our work is a simple hill-climbing approach that converges relatively quickly when compared with actual work. However, under this framework, if one uses a more sophisticated local search phase that attempts to escape from local optima, as the tabu search local phase of Gliesch and Ritt [15], for example, solution quality can be significantly improved. Naturally, this comes at the expense of more computational effort, but it is an idea worthwhile pursuing to tackle problems with similar structure, which is having a p-median type of objective function, subject to multiple balancing constraints.

Acknowledgments: The presentation of this paper was improved thanks to the remarks and criticism of the anonymous reviewers and the area editor. We are very grateful for their input. The first author was supported by the Mexican National Council for Science and Technology (CONACYT grants CB05-1-48499Y and CB11-1-166397) and by UANL through its Scientific and Technological Research Support Program (grants UANL-PAICYT CE012-09, IT511-10, CE728-11, and CE331-15.) The fourth author was supported by CONACYT's Postdoctoral Research Program under grant 2019-000019-01NACV-00239.

References

[1] P. K. Bergey, C. T. Ragsdale, and M. Hoskote. A simulated annealing genetic algorithm for the electrical power districting problem. *Annals of Operations Research*, 121(1):33–55, 2003.

[2] M. Blais, S. D. Lapierre, and G. Laporte. Solving a home-care districting problem in an urban setting. *Journal of the Operational Research Society*, 54(11):1141–1147, 2003.

[3] B. Bozkaya, E. Erkut, and G. Laporte. A tabu search heuristic and adaptive memory procedure for political districting. *European Journal of Operational Research*, 144(1):12–26, 2003.

[4] F. Caro, T. Shirabe, M. Guignard, and A. Weintraub. School redistricting: Embedding GIS tools with integer programming. *Journal of the Operational Research Society*, 55(8):836–849, 2004.

[5] S. J. D'Amico, S.-J. Wang, R. Batta, and C. M. Rump. A simulated annealing approach to police district design. *Computers & Operations Research*, 29(6):667–684, 2002.

[6] L. S. de Assis, P. M. Franca, and F. L. Usberti. A redistricting problem applied to meter reading in power distribution networks. *Computers & Operations Research*, 41:65–75, 2014.

[7] J. A. Díaz, D. E. Luna, and M. G. Sandoval. Bounding procedures and exact solutions for a class of territory design problems. In R. Z. Ríos-Mercado, editor, *Optimal Districting and Territory Design*, volume 284 of *International Series in Operations Research and Management Science*, chapter 5, pages 77–106. Springer, Cham, Switzerland, 2020.

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

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

[10] M. G. Elizondo-Amaya, R. Z. Ríos-Mercado, and J. A. Díaz. A dual bounding scheme for a territory design problem. *Annals of Operations Research*, 44:193–205, 2014.

[11] E. Fernández, J. Kalcsics, S. Nickel, and R. Z. Ríos-Mercado. A novel maximum dispersion territory design model arising in the implementation of the WEEE-directive. *Journal of the Operational Research Society*, 61(3):503–514, 2010.

[12] B. Fleischmann and J. N. Paraschis. Solving a large scale districting problem: A case report. *Computers & Operations Research*, 15(6):521–533, 1988.

[13] S. Gentry, E. Chow, A. Massie, and D. Segev. Gerrymandering for justice: Redistricting U.S. liver allocation. *Interfaces*, 45(5):462–480, 2015.

[14] J. A. George, B. W. Lamar, and C. A. Wallace. Political district determination using large-scale network optimization. *Socio-Economic Planning Sciences*, 31(1):11–28, 1997.

[15] A. Gliesch and M. Ritt. A generic approach to districting with diameter or center-based objectives. In M. López-Ibáñez, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'19)*, pages 249–257, New York, 2019. ACM. ISBN 978-1-4503-6111-8.

[16] S. W. Hess and S. A. Samuels. Experiences with a sales districting model: Criteria and implementation. *Management Science*, 18(4):P41–P54, 1971.

[17] 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.

[18] M. Hojati. Optimal political districting. *Computers & Operations Research*, 23(12):1147–1161, 1996.

[19] J. Kalcsics and R. Z. Ríos-Mercado. Districting problems. In G. Laporte, S. Nickel, and F. Saldanha da Gama, editors, *Location Science*, chapter 25, pages 703–741. Springer, Cham, Switzerland, 2nd edition, 2019.

[20] F. Liberatore, M. Camacho-Collados, and B. Vitoriano. Police districting problem: Literature review and annotated bibliography. In R. Z. Ríos-Mercado, editor, *Optimal Districting and Territory Design*, volume 284 of *International Series in Operations Research and Management Science*, chapter 2, pages 9–28. Springer, Cham, Switzerland, 2020.

[21] J. F. López-Pérez and R. Z. Ríos-Mercado. Embotelladoras ARCA uses operations research to improve territory design plans. *Interfaces*, 43(3):209–220, 2013.

[22] P. G. Marlin. Application of the transportation model to a large scale "districting" problem. *Computers & Operations Research*, 8(2):83–96, 1981.

[23] M. E. Mayorga, D. Bandara, and L. A. McLay. Districting and dispatching policies to improve the efficiency of emergency medical service (EMS) systems. *IIE Transactions in Healthcare Systems Engineering*, 3(1):39–56, 2013.

[24] A. Mehrotra, E. L. Johnson, and G. L. Nemhauser. An optimization based heuristic for political districting. *Management Science*, 44(8):1100–1114, 1998.

[25] 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.

[26] F. Pukelsheim, F. Ricca, B. Simeone, A. Scozzari, and P. Serafini. Network flow methods for electoral systems. *Networks*, 59(1):73–88, 2012.

[27] F. Ricca and A. Scozzari. Mathematical programming formulations for practical political districting. In R. Z. Ríos-Mercado, editor, *Optimal Districting and Territory Design*, volume 284 of *International Series in Operations Research and Management Science*, chapter 6, pages 105–128. Springer, Cham, Switzerland, 2020.

[28] F. Ricca, A. Scozzari, and B. Simeone. Political districting: From classical models to recent approaches. *Annals of Operations Research*, 204(1):271–299, 2013.

[29] R. Z. Ríos-Mercado, editor. *Optimal Districting and Territory Design*, volume 284 of *International Series in Operations Research and Management Science*. Springer, Cham, Switzerland, 2020.

[30] R. Z. Ríos-Mercado and J. F. Bard. An exact algorithm for designing optimal districts in the collection of waste electric and electronic equipment through an improved reformulation. *European Journal of Operational Research*, 276(1):259–271, 2019.

[31] R. Z. Ríos-Mercado and H. J. Escalante. GRASP with path relinking for commercial districting. *Expert Systems with Applications*, 44:102–113, 2016.

[32] 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.

[33] R. Z. Ríos-Mercado and J. F. López-Pérez. Commercial territory design planning with realignment and disjoint assignment requirements. *Omega*, 41(3):525–535, 2013.

[34] R. Z. Ríos-Mercado and J. C. Salazar-Acosta. A GRASP with strategic oscillation for a commercial territory design problem with a routing budget constraint. In I. Batyrshin and G. Sidorov, editors, *Advances in Soft Computing*, volume 7095 of *Lecture Notes in Artificial Intelligence*, pages 307–318. Springer, Heidelberg, Germany, 2011.

[35] R. Z. Ríos-Mercado, J. R. Maldonado-Flores, and J. L. González-Velarde. Tabu search with strategic oscillation for improving recollection assignment plans of waste electric and electronic equipment. Technical Report PISIS–2017–01, Graduate Program in Systems Engineering, Universidad Autónoma de Nuevo León, San Nicolás de los Garza, Mexico, 2017.

[36] 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.

[37] M. A. Salazar-Aguilar, R. Z. Ríos-Mercado, and J. L. González-Velarde. A bi-objective programming model for designing compact and balanced territories in commercial districting. *Transportation Research Part C: Emerging Technologies*, 19(5):885–895, 2011.

[38] M. A. Salazar-Aguilar, J. L. González-Velarde, and R. Z. Ríos-Mercado. A divide-and-conquer approach to commercial territory design. *Computación y Sistemas*, 16(3):309–320, 2012.

[39] M. A. Salazar-Aguilar, R. Z. Ríos-Mercado, J. L. González-Velarde, and J. Molina. Multiobjective scatter search for a commercial territory design problem. *Annals of Operations Reseach*, 199(1):343–360, 2012.

[40] M. A. Salazar-Aguilar, R. Z. Ríos-Mercado, and J. L. González-Velarde. GRASP strategies for a bi-objective commercial territory design problem. *Journal of Heuristics*, 19(2):179–200, 2013.

[41] M. G. Sandoval, J. A. Díaz, and R. Z. Ríos-Mercado. An improved exact algorithm for a territory design problem with p-center-based dispersion minimization. *Expert Systems with Applications*, 146:113150, 2020.

[42] T. Shirabe. Classification of spatial properties for spatial allocation modeling. *Geoinformatica*, 9(3):269–287, 2005.

[43] S. Yanık and B. Bozkaya. A review of districting problems in health care. In R. Z. Ríos-Mercado, editor, *Optimal Districting and Territory Design*, volume 284 of *International Series in Operations Research and Management Science*, chapter 3, pages 29–53. Springer, Cham, Switzerland, 2020.

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