A multistart alternating tabu search for commercial districting

Alex Gliesch[1], Marcus Ritt[1], and Mayron C. O. Moreira[2]

[1] Federal University of Rio Grande do Sul, Porto Alegre, Brazil
{alex.gliesch,marcus.ritt}@inf.ufrgs.br
[2] Federal University of Lavras, Lavras, Brazil
mayron.moreira@dcc.ufla.br

Abstract. In this paper we address a class of commercial districting problems that arises in the context of the distribution of goods. The problem aims at partitioning an area of distribution, which is modeled as an embedded planar graph, into connected components, called districts. Districts are required to be mutually balanced with respect to node attributes, such as number of customers, expected demand, and service cost, and as geometrically-compact as possible, by minimizing their Euclidean diameters. To solve this problem, we propose a multistart algorithm that repeatedly constructs solutions greedily and improves them by two alternating tabu searches, one aiming at achieving feasibility through balancing and the other at maximizing district compactness. Computational experiments confirm the effectiveness of the different components of our method and show that it significantly outperforms the current state of the art, improving known upper bounds in almost all instances.

Keywords: Districting. Territory design. Planar graphs. Tabu search. Multistart. Heuristic algorithm. Compactness. Graph connectivity.

1 Introduction

The goal of districting problems is to group basic geographic units into clusters, called districts, which satisfy given constraints. Typically, units represent geographic entities such as city blocks, and have some attributes, such as population or expected travel cost. An underlying planar graph establishes adjacencies between these units. Common requirements include that districts be equally balanced with respect to attribute values, be geometrically compact by having a more or less regular convex shape, and be connected with respect to unit adjacencies.

In recent years, districting problems have been studied in a wide range of applications, such as the design of electoral districts [1–4], sales territories [5–7], police districts [8], health care districts [9, 10], or agrarian land parcels [11]. [12] provides an extensive overview on applications and solution techniques to many districting problems.

In this paper, we study districting applied to a commercial territory design problem that originates from a real-world context of a bottled beverage distribution company. In this problem, each geographic unit represents a city block and has three attributes, or "activities": the number of customers, the product demand, and the workload. The goal is to divide the city blocks into a fixed number of contiguous, compact districts that are balanced with respect to all three activities.

The problem's domain was first introduced by [5]. They present a core model that optimizes compactness through an objective function based on the dispersion measure of the well-known p-centers problem, while treating contiguity and balance of activities as constraints. They further show that the problem is NP-hard and propose heuristic solutions by a reactive GRASP strategy.

Since then, several models and solution approaches to this problem have been studied. [13] focus on solving it optimally through mixed-integer programming. They also propose a variant model whose objective function is based on the p-median problem, as opposed to a p-center approach (i.e. minimizing maximum and not average distance to the centers). [14] study a multi-objective variant that optimizes both compactness (through a p-median approach) and balance. More recently, [7] proposed a GRASP with a path-relinking strategy to solve a novel variant that needs no district centers for compactness, as the p-center and p-median approaches do, but uses the more flexible concept of polygon diameters.

In this paper, we address the more recent model by [7]. Section 2 presents the problem formally and provides the necessary definitions. In Section 3 we propose a new method for solving this problem, using a multistart tabu search, which alternates between improving balance constraints and maximizing compactness. In Section 4, we report on the results of computational experiments which evaluate the different components of the proposed method, and compare the quality of the solutions to approaches from the literature. We conclude in Section 5.

2 Definitions

We are given an undirected connected planar graph $G = (V, E)$ of $n = |V|$ nodes and $m = |E|$ edges, and a number of districts $p \leq n$. Each node $u \in V$ corresponds to a basic geographic unit and is associated with three activity values w_{u1}, w_{u2}, w_{u3} and coordinates $x_u \in \mathbb{R}^2$. In the following, we use the set notation $[n] = \{1, \ldots, n\}$.

A districting plan (or solution) S for G is a partition of a subset of the vertices $S_1 \dot\cup \cdots \dot\cup S_p \subseteq V$, where each S_i represents a district. We say that a solution S is *complete* if $\bigcup_{i \in [p]} S_i = V$. We use the notation $S(u) = i$ to mean that node u is *assigned to* district i, i.e., $u \in S_i$. If S is incomplete, then some $S(u)$ will be undefined, in which case we say that u is *unassigned*.

We define the *boundary* of a district S_i as $\partial S_i = \{v \mid \{v, u\} \in E, u \in S_i, v \notin S_i\}$, and its free boundary as $\partial_f S_i = \{v \in \partial S_i \mid v \text{ is unassigned}\}$. We say that districts i and j are *neighbors* if $S_i \cap \partial S_j \neq \emptyset$. A *move* is an operation $u \to i$ that

shifts the assignment of node u to district i. We use the notation $S[\nu]$ to refer to S after performing move ν.

Let $w_a(S_i) = \sum_{u \in S_i} w_{ua}$ be the total value of district i with respect to one of the three activities $a \in [3]$, and μ_a be the mean value of w_{ua} over all $u \in V$. We define $b_a(S_i) = |(w_a(S_i) - \mu_a)/\mu_a|$ as the absolute relative deviation of the total value of activity a from the mean activity a among all districts. We say that a district S_i is *balanced* if $b_a(S_i) \leq \tau_a$ for $a \in [3]$, where $\tau_a \geq 0$ is a tolerance parameter. A solution is balanced if all its districts are balanced. The *imbalance* of a district S_i is defined as the excess of the activities over the tolerances $B(S_i) = \sum_{a \in [3]} \max\{0, b_a(S_i) - \tau_a\}$, and the imbalance of a solution S is the total excess of all districts $B(S) = \sum_{i \in [p]} B(S_i)$. A solution S is balanced iff $B(S) = 0$.

Finally, we define the *diameter* of a district S_i as $D(S_i) = \max_{u,v \in S_i} d_{uv}$, where $d_{uv} = |x_u - x_v|_2$ is the Euclidean distance between nodes u and v, and the diameter of a solution S as the maximum diameter $D(S) = \max_{i \in [p]} D(S_i)$. Since the diameter is used here as a measure for compactness, where a small diameter corresponds to a high compactness, we will use both terms interchangeably.

A districting plan S is considered feasible if (1) it is complete, (2) it is balanced, and (3) the subgraph induced by each S_i in G is connected. Our goal is to find a feasible districting plan that minimizes $D(S)$.

3 Proposed methods

The overall structure of our algorithm is similar to a greedy randomized adaptive search procedure (GRASP, [15]): we repeatedly construct a solution using a randomized greedy algorithm and improve it by a heuristic based on local search, returning the best solution found over all repetitions. Our method differs from traditional GRASP in two ways. First, during construction we do not select candidates randomly from a restricted candidate list, as is standard, but select the best overall candidate in two stages by different criteria. Second, instead of improving solutions by local search, we perform two tabu searches alternatedly, one focusing on making solutions feasible by reducing imbalance and the second one focusing on optimizing compactness.

Using different searches for each objective has been applied sucessfully to districting before [16, 6]. The main idea is to focus the effort on optimizing one objective by searching a custom neighborhood that only considers moves with respect to that objective, while (to some extent) ignoring others. A common alternative would be to perform a single search using a composite fitness function that assigns weights to each objective, as is done by [3] and [7]. The main disadvantage of this approach is that it is usually difficult to find weights that work well for all instances.

Algorithm 1 outlines our method. Each iteration starts by computing an initial solution through a greedy constructive algorithm (line 3). Next, the local improvement phase (lines 5–10) optimizes first compactness and then balance, repeating this for a fixed number of iterations $A_{\max}$. After the first iteration,

Algorithm 1 multistart alternating tabu search.

```
1: R ← ∅
2: repeat
3:     S ← greedyRandomizedConstruction()
4:     X = {S}
5:     for i ∈ [A_max] do
6:         S ← optimizeCompactness(S)
7:         S ← optimizeBalance(S, α)
8:         if S ∈ X then
9:             break
10:        X ← X ∪ {S}
11:    S ← arg min{(B(S'), D(S')) | S' ∈ X}
12:    S ← optimizeBalance(S, ∞)
13:    R ← arg min{B(R), B(S)}
14: until stopping criterion satisfied
15: return R
```

subsequent searches are typically much faster, since solutions are already close to a local minimum in both criteria. When optimizing for balance we do not allow the diameter of the solution found in the preceding optimization of compactness to increase by more than a factor α, where $\alpha \geq 0$ is a slack parameter. This maintains a certain progress in the reduction of the diameter. We store intermediate solutions in a lookup table after each iteration and stop if cycling is detected (lines 8–9). This is done with a hash table; since $A_{\max}$is expected to be small (less than 1000), this has is no effect on performance. In the end, we select the intermediate solution with minimum D, or the one with minimum B if no balanced solutions were found (line 11). In the latter case, we make a final attempt at finding a feasible solution by optimizing for balance with $\alpha = \infty$ (line 12). We iterate until a stopping criterion (either a maximum number of iterations or a time limit) is met, and report the best solution found.

A tabu search is used to optimize compactness and balance (procedures optimizeCompactness and optimizeBalance). Tabu search is a non-monotone local search proposed by [17]. Starting from an initial solution, it repeatedly moves it to the best neighbor. To avoid cycling, some neighbors are declared tabu, i.e. they cannot be selected. The two tabu searches in our algorithm explore different neighborhoods, iteratively performing local movements for at most $I_{\max}$ consecutive non-improving iterations, and returning the best intermediate solution. The tabu list contains nodes whose assignment was recently shifted and prohibits to shift them again for t iterations, where the tabu tenure t is a parameter. The tabu list is emptied between consecutive searches.

In the following subsections, we describe each method (greedy construction and tabu searches) in more detail.

3.1 Solution construction

Initial solutions are constructed in two steps: first, we select p initial nodes which will serve as seeds for each district, and then we grow districts by iteratively assigning them a greedily-selected node on their free boundary, until the solution is complete.

In the first step, we compute seed nodes s_i for each district $i \in [p]$. Our method is based on the greedy constructive heuristic proposed by [18] to solve the p-dispersion problem. The p-dispersion problem aims at finding p elements from a set of points on the plane such that the minimum distance between any two elements is maximized. First, we select s_1 randomly from V. Then, for each $i \in [2, p]$ in sequence, we pick a subset $V' \subseteq V \setminus \{s_1, \ldots, s_{i-1}\}$ of size $\sqrt{n}$ uniformly at random, and define the ith center $s_i = \arg\max_{u \in V'} \min_{j \in [i-1]} d_{us_j}$ to be the node in V' that maximizes the minimum distance to the previously chosen centers. [7] use a similar strategy, with the difference that each s_i is chosen from $V' = V \setminus \{s_1, \ldots, s_{i-1}\}$. The main reason we chose to reduce V' to a smaller, random subset is that, with a time complexity of $\Theta(p^2|V'|)$, the former method happens to be too slow for larger instances. Our approach also has the advantage of increasing variability. In practice, we have found the two strategies to have very little difference in average solution quality.

The second step starts from districts $S_i = \{s_i\}$. We maintain for each district i the candidate node $c_i \in \partial_f S_i$ that minimizes $D(S_i \cup \{c_i\})$. If multiple such nodes exist, we select one randomly. At first, we compute c_i for every district by iterating over each $\partial_f S_i$. Then, we repeatedly select the district j with smallest total normalized activity $\sum_{a \in [3]} w_a(S_j)/\mu_a$ and assign $S := S[c_j \to j]$, until S is complete. We recompute c_i for all districts S_i with $c_i = c_j$ after every assignment, because the free boundary $\partial_f S_i$ of these districts changes. There are, in average, only $O(1)$ such districts, since G is planar, and thus has an average degree less than 6. Because only boundary nodes are considered, connectivity is preserved.

This strategy is greedy in the way that it selects a candidate with smallest increase of D for each district. This steers the construction towards initial solutions with low diameters, as opposed to a different strategy that might, for example, try to find solutions that are as balanced as possible by minimizing B. Our rationale is that, in most cases, even highly imbalanced solutions can be made feasible quickly with a local search-based procedure, whereas it is substantially harder to obtain large improvements in compactness. Nonetheless, by selecting the district with minimum activity at each step, we ensure a certain balancing and prevent districts in denser regions from growing too large.

3.2 Optimizing balance

We optimize for balance by shifting boundary nodes between neighboring districts. More specifically, we consider moves in the neighborhood $N_{bal}(S) = \{u \to i \mid i \in [p], u \in \partial S_i\}$. As explained above, we avoid moves which lead to a diameter that exceeds the diameter D_0 of the starting solution (obtained after optimizing compactness) by more than a factor α. Thus, the core search algorithm repeatedly selects a move $u \to i \in N_{bal}$ such that:

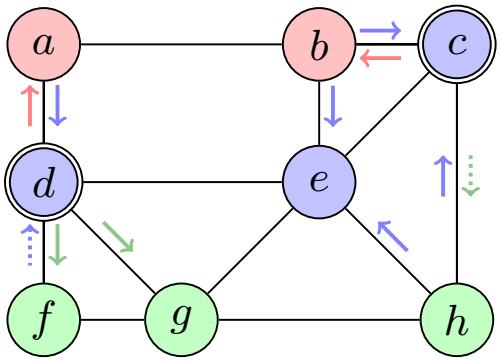


Fig. 1: Neighborhood N_{bal}. Each district is represented by a color. Colored arrows emanating from a node represent a possible shift of the node to the district of that color. We have omitted moves that break connectivity, as is the case of all shifts of e and g. The doubly-circled nodes c and d are the farthest same-district nodes, and make up the solution's diameter. The dotted arrows from $c \to$ ■ and $f \to$ ■ represent moves that would increase the current diameter (the new farthest pair would be (c, f) in both cases): depending on the value of α, they might not be applicable.

1. u is not tabu
2. $D(S[u \to i]) \leq (1 + \alpha)D_0$
3. $S(u)$ remains connected after removing u
4. $B(S[u \to i])$ is minimized

If multiple such moves exist, we select one of them randomly. We then assign $S := S[u \to i]$, and mark u as tabu. We stop either after $I_{\max}$ consecutive steps without improvement, or if there are no more valid moves, or if the solution is balanced ($B(S) = 0$), since there can be no further improvement, and return the intermediate solution with minimum $(B(S), D(S))$, compared lexicographically. Figure 1 illustrates the effective neighborhood seached with a minimalistic example.

Similarly to the constructive algorithm of Section 3.1, we can avoid re-evaluating all of N_{bal} at every iteration by caching, for each district i, a candidate $c_i \in \partial S_i$ that satisfies items 1, 2, 3 and 4 above. Each c_i is initially computed by iterating over ∂S_i. At every iteration, we choose the candidate c_i, $i \in [p]$, with lowest expected imbalance and perform $S := S[c_i \to i]$. Upon assigning $c_i \to i$, we update c only for the districts affected by the move (i.e., i and $S(c_i)$) and their direct neighbors. Again, due to G being planar, there are in average only $O(1)$ such districts (here, we can view districts as nodes and their neighboring relations as edges). Further, at each iteration a node ceases to be tabu, and thus we also update c for neighboring districts of that node.

3.3 Optimizing compactness

Only a small subset of the neighborhood N_{bal} actually reduces the diameter, and so when $B(S) = 0$ or is at a plateau it would be wasteful to continue to search all of N_{bal}. Let the set $L(S) = \{u \mid \exists v \in S, S(u) = S(v),\ d_{uv} = D(S)\}$ contain all nodes incident to some maximum diameter of S. When optimizing compactness, we consider only moves in the neighborhood $N_{cmp}(S) = \{u{\to}i \mid i \in [p],\ u \in L(S) \cap \partial S_i\}$ which have the potential to improve $D(S)$. In the example of Figure 1, we have $L = \{c, d\}$ and $N_{cmp} = \{d{\to}\blacksquare, d{\to}\blacksquare, c{\to}\blacksquare, c{\to}\blacksquare\}$.

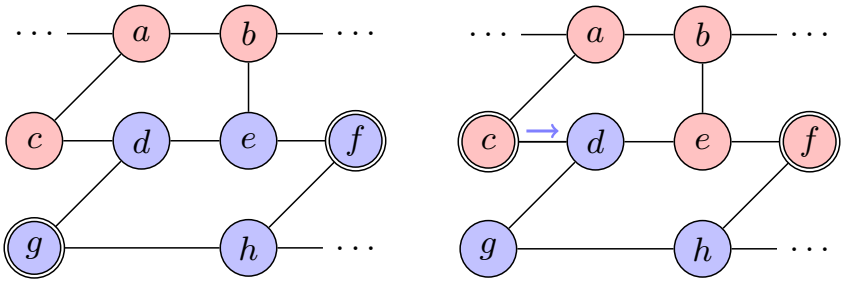
Note that if $|L(S)| > 2$, then even shifting a node in L might not change D, since there still exists another pair of nodes with equal distance (except if a node is incident to multiple diameters). This situation happens frequently when the node locations are regularly distributed, for example, in a grid. It is clear, however, that reducing the cardinality of L is still an improvement. Thus, we rank such moves accordingly by using a modified objective function $D'(S) = D(S) + \epsilon|L(S)|$, where ϵ is some small constant. In our implementation, we use $\epsilon = 5 \times 10^{-7}$.

We then optimize for compactness by repeatedly searching for the move $u{\to}i \in N_{cmp}$ with minimum $D'(S[u{\to}i])$, and assigning $S := S[u{\to}i]$. If multiple such moves exist, we select one at random. As before, we discard moves that break connectivity or are tabu, and stop either after $I_{\max}$ steps without improvement or if there are no more valid moves, and return the best intermediate solution with respect to $(D(S), B(S))$ compared lexicographically.

A common situation occurs when no nodes in L are on the boundary of another district, meaning that $N_{cmp}(S) = \emptyset$ and thus no moves can be made. This represents a plateau that is particularly difficult to escape, since it requires at least as many moves as the length of the shortest path from a node in L to the boundary of some district. In practice, it is very unlikely that this specific set of moves will be performed while searching N_{bal}. Therefore, when this kind of plateau is reached during search, we attempt to escape it as follows. We search G for the path π of smallest Euclidean distance starting from any node $u \in L(S)$ and ending at some node $v \in \partial S_{S(u)}$. We then assign all nodes in π, including u but not v, to district $S(v)$, as long as $S(u)$ remains connected (otherwise we stop and give up). Reassigning node u forces a change in L, and consequently in D' (barring the rare case that u remains in L, paired to the same number of nodes and distances as before). We repeat this process at most sp_{max} times, where sp_{max} is a parameter, or until $|N_{cmp}| > 0$. Using a standard shortest paths algorithm that orders active nodes with a priority queue, finding path π takes, on average, $O(n/p \log n/p)$ steps. Figure 2 illustrates the process, which call shortest path escape, or sp-escape, for short.

3.4 Data structures for efficient operations

The proposed algorithm depends on repeated tests of connectivity and queries of the diameter of districts. Implemented naively, these operations can become the bottleneck of the method. In this section we explain the data structures we have used to implement them efficiently.



(a) Before sp-escape. (b) After sp-escape.

Fig. 2: The sp-escape process. We use "$\cdots$" to mean that the solution continues with other districts at that point. In 2a, the two farthest nodes g and f are not on the boundary of any other district (i.e., $N_{cmp} = \emptyset$). We therefore search for the shortest distance path from either g or f to the boundary of ■, and find feb. We then assign all nodes in feb to ■, obtaining the solution in 2b. Notice that, after this process, the search is able to continue, since $N_{cmp} = \{c \rightarrow ■\}$.

Connectivity queries Testing for loss of connectivity when considering a move $u \rightarrow i$ by performing a graph search on $S(u)$ is very time-consuming, taking $O(n/p)$, on average, and can easily dominate the running time of the algorithm, since a large number of candidates must be evaluated before performing each move.

The size of district boundaries in planar domains in general scales closely to $O(\sqrt{n/p})$. Under this assumption, we can test for loss of connectivity in an amortized time of $O(\sqrt{n/p})$ by maintaining a bit array of size n indicating whether each node's removal will disconnect the district owning it. This array is computed for each district i in time $O(|S_i|)$ using the standard algorithm to find articulation nodes [19]. Before the local improvement phase starts, we compute the articulation nodes of all districts. During search, upon applying a move $u \rightarrow i$, we need only to recompute articulation nodes in two districts: $S(u)$ and i. Although this update takes $O(n/p)$ time, on average, when searching N_{bal} this is only done once for every full evaluation of a district boundary, of average size $O(\sqrt{n/p})$.

This could be improved even further by using the algorithm of [20] for decremental connectivity in planar graphs, as it allows both tests and updates in constant time, or the geo-graph model of [21], which provides an efficient method for maintaining connectivity and hole constraints in districting problems.

Maintaining diameters A brute-force method computes the diameter of a district i in $O(|S_i|^2)$ by checking the distance between all node pairs. Similarly, the expected diameter of a move $u \rightarrow i$ can be computed in $O(|S_i|)$ by checking the distance from u to all nodes in S_i.

We can do better by using the fact that the most distant pair of points of a point set must lie on that point set's convex hull [22]. Knowing that the convex hull of a set of n uniformly-distributed points on the plane has expected size $O(\log n)$ [23] and assuming we maintain the convex hull of every district, we can compute the diameter of district i in expected time $O(\log |S_i|)$ by executing the so-called "rotating calipers" algorithm [22] on the convex hull of S_i. (Here, we assume that node coordinates are more or less uniformly-distributed.)

We compute convex hulls with the monotone chain algorithm [24], which runs in $O(n \log n)$ time for any set of n points or in $O(n)$ time if the points are already sorted lexicographically by their coordinates. Thus, if we sort the list of nodes of every district by their coordinates before the local improvement phase and keep them sorted after each move using a simple linear update, we can maintain the convex hulls in time $O(|S_i|)$ per update. This could be further improved using the algorithm of [25], which maintains convex hulls of n points in $O(\log^2 n)$ time per update.

4 Computational experiments

4.1 Test instances

[7] report results on two data sets called DS and DT. Both contain 20 randomly generated instances, with each instance having $n = 500$ nodes, $p = 10$ districts and m ranging from 917 to 986 edges. These data sets were originally proposed by [5] and were generated to resemble real-world scenarios. Data set DS selects node activities uniformly from the intervals $[4, 20], [15, 400]$, and $[15, 100]$, and data set DT selects activities from a non-uniform symmetric distribution. We refer to [5] for more details on these data sets.

Because all the instances in these data sets have the same size, for a broader evaluation we have generated an additional data set "DL" with 4 instances of each combination $n \in \{1000, 2500, 5000, 10000\}$ and $p \in \{n/200, n/100, n/62.5\}$, 48 in total. The three levels of p roughly represent difficulties "easy", "medium" and "hard", with respect to achieving feasibility. We have chosen these size ranges because no existing methods were able to consistently find feasible solutions for larger instances, whereas smaller instances tended to be trivial. For DL, we have decided to use the same uniform activity generation used for generating data set DS, as we have found it significantly more challenging than the one used for DT. As in [7], we use balancing tolerances $\tau_1 = \tau_2 = \tau_3 = 0.05$. The coordinates of each node were drawn uniformly at random from $[0, 1000]^2$. The graph topology was obtained by computing a Delaunay triangulation on the node coordinates, which ensures connectivity and planarity.

4.2 Experimental setup

We have implemented our algorithms in C++ and compiled them with GCC 5.4.0 and maximum optimization. The code, data sets, instance generators and

Table 1: Parameter calibration: initial ranges and best setting found by iterative racing.

Param.	Description	Optimization range	Best
t	Tabu tenure	$\{0.5, 1, 1.5, 2, 2.5, 3\}p$	$0.5p$
$I_{\max}$	Max. consecutive non-improving tabu iter.	$10^2 \times \{1, 5, 10, 25, 50\}$	500
$A_{\max}$	Max. alternations between tabu searches	$\{1, 5, 10, 25, 50\}$	10
sp_{max}	Max. sp-escape iterations	$\{1, 5, 10, 25, 50\}$	25
α	Slack factor allowed to D when optimizing B	$[0.0, 0.2]$	0.0

detailed results of the experiments are available to the community upon request. All experiments were performed on a PC with an 8-core AMD FX-8150 processor and 32 GB of main memory running Ubuntu Linux 16.04. For each experiment, only one core was used.

The main parameters of our algorithms were calibrated with the irace package in GNU R [26], with a budget of 2000 runs and a time limit of 5 minutes per run. This time limit was chosen due to time availability reasons. To avoid overfitting, for the calibration we have generated an additional data set of 48 instances, with the same characteristics as DL. The parameter t was set relative to the number of districts p, as recommended by [6]. Table 1 shows the parameter ranges used for calibration, and the best values obtained by racing. Unexpectedly, the best value for α was found to be 0, which suggests that allowing the compactness to increase when optimizing balance is unhelpful.

In the following, we report experiments with instance set DL for several configurations. For each instance, we have executed 10 replications of the algorithm with 30 minutes of running time. For each instance size (n, p), we report averages over all replications of instances of that size. In each table, we report the average relative deviation of the diameters from the best known value "$D(\%)$", over replications that achieved feasibility (i.e., $B = 0$), and the empirical probability "p. f." of achieving feasibility on each multistart iteration, averaged over all replications.

4.3 Experiment 1: constructive algorithm

In Section 3.1 we argued that a construction strategy biased towards minimizing D is more effective, since optimizing balance during the local improvement phase is much easier than optimizing compactness. This experiment aims to support that argument by comparing our standard constructive approach (*greedy-D*) to two other strategies. The first one, *greedy-B*, greedily selects the best candidate c_i of each district with respect to balance B instead of diameter D. The second one, *BFS*, constructs solutions non-greedily by expanding free boundary-nodes in breadth-first order, each time assigning the expanded node to a neighboring district, with ties broken lexicographically. Both approaches generate initial seeds from the p-dispersion-based heuristic.

Table 2: Comparison of constructive algorithms.

n	p	*greedy-D*				*greedy-B*				*BFS*			
		B_C	$D_C(\%)$	$D(\%)$	p. f.	B_C	$D_C(\%)$	$D(\%)$	p. f.	B_C	$D_C(\%)$	$D(\%)$	p. f.
1,000	5	0.19	48.97	**0.13**	0.99	0.22	51.51	0.52	1.00	3.09	64.07	0.18	1.00
1,000	10	0.97	65.08	**0.03**	0.89	0.86	105.63	2.45	0.93	6.71	122.12	0.10	0.92
1,000	16	2.67	104.91	**1.21**	0.43	1.90	134.61	6.46	0.58	12.13	160.74	1.96	0.56
2,500	12	1.27	63.33	**0.47**	0.99	1.05	99.32	11.12	0.99	8.39	118.86	1.28	1.00
2,500	25	3.14	102.58	**1.15**	0.77	3.42	137.02	16.21	0.83	17.48	218.61	2.62	0.86
2,500	40	7.97	115.68	**6.41**	0.21	6.17	187.62	21.06	0.30	28.61	246.94	9.07	0.32
5,000	25	2.25	88.98	**1.63**	0.97	2.46	136.12	28.40	0.98	15.73	164.05	3.33	0.99
5,000	50	8.17	112.37	**1.93**	0.58	7.62	236.50	34.81	0.67	32.48	266.82	5.89	0.74
5,000	80	16.46	95.33	**7.50**	0.05	12.90	240.87	43.48	0.09	56.92	316.57	12.53	0.12
10,000	50	6.89	106.42	**2.01**	0.92	8.77	146.55	48.14	0.94	29.83	249.46	7.40	0.94
10,000	100	17.73	109.97	**5.29**	0.34	14.81	264.57	51.79	0.46	60.90	307.61	11.64	0.49
10,000	160	34.86	120.40	**46.54**	0.01	28.28	220.67	93.79	0.01	107.97	301.49	69.26	0.02
Average		8.55	94.50	**6.19**	0.60	7.37	163.42	29.85	0.65	31.69	211.45	10.44	0.66

Table 2 shows the results of the three approaches. For each approach, columns B_C and $D_C(\%)$ report, for the iteration with smallest final diameter, the average imbalance and diameter (relative to the best known value) of the constructive part, whereas columns $D(\%)$ report the final average diameter (relative to the best known value) after local improvement.

We can see that *greedy-D* leads to smallest diameters, on average, followed by *BFS* and *greedy-B*. As expected, *greedy-D* had the advantage in initial compactness and *greedy-B* in initial balance (though a small one), both yielding significantly better initial solutions than *BFS*. Yet, all three methods had a similar empirical probability (p. f.) of finding a feasible solution after local improvement, at each multistart iteration. This supports our claim that the difficulty to make near-balanced and highly imbalanced initial solutions feasible during local improvement is similar.

Although *BFS* has worse initial solutions, after local improvement they become competitive in both compactness and probability of finding a feasible solution. A possible explanation for this unexpected success is that, because instances in DL have very regular topologies (a Delaunay triangulation of uniform random points), a breadth-first strategy which, by design, typically yields compact districts in graph space will also yield somewhat compact districts in Euclidean space.

4.4 Experiment 2: search strategies

This experiment evaluates the effectiveness of the alternating tabu search strategy (A-TS), alternating local searches (A-LS) and a single tabu search with an objective function which is a weighted sum of balance and compactness (W-TS). For the weighted approach, we use the same objective function as [7]:

Table 3: Comparison of search strategies.

n	p	A-TS			A-LS			W-TS		
		D(%)	Iter.	p. f.	D(%)	Iter.	p. f.	D(%)	Iter.	p. f.
1,000	5	0.13	24,429	0.99	**0.08**	118,903	0.99	0.41	12,904	0.99
1,000	10	**0.03**	20,775	0.89	0.90	118,291	0.76	3.35	10,819	0.80
1,000	16	**1.21**	10,787	0.43	4.25	112,855	0.18	10.30	12,162	0.34
2,500	12	**0.47**	13,085	0.99	3.54	33,536	0.93	7.09	5,265	0.99
2,500	25	**1.15**	10,168	0.77	7.39	34,733	0.32	19.63	5,436	0.72
2,500	40	**6.41**	6,276	0.21	72.62	34,186	0.01	34.21	6,772	0.15
5,000	25	**1.63**	7,767	0.97	9.02	13,178	0.51	21.63	2,980	0.96
5,000	50	**1.93**	5,200	0.58	15.76	13,602	0.03	37.73	3,333	0.53
5,000	80	**7.50**	3,697	0.05	101.86	12,941	0.00	51.54	3,767	0.04
10,000	50	**2.01**	3,406	0.92	16.27	4,932	0.11	41.19	1,597	0.90
10,000	100	**5.29**	2,240	0.34	213.95	4,914	0.00	59.17	1,737	0.29
10,000	160	**46.54**	1,786	0.01	—	4,545	0.00	97.47	1,530	0.00
Average		**6.19**	9,135	0.60	40.51	42,218	0.32	31.98	5,692	0.56

$W(S) = 0.3\,B(S) + 0.7\,D(S)/d_{max}$, where $d_{max} = \max_{u,v \in V} d_{uv}$ is the maximum distance of two nodes. The weighted approach repeatedly performs a single tabu search in the neighborhood N_{bal} with the goal of minimizing $W(S)$, stopping after $I_{\max}$ consecutive non-improving iterations. As long as D does not change, we can use the candidate caching explained in Section 3.2 to avoid re-evaluating N_{bal} fully every time. For the local search approach, we stop as soon as no candidates improve the incumbent solution, but still allow a fixed $A_{\max}$ alternating searches.

Table 3 shows the results. Columns "Iter." show the average number of multistart iterations for each approach. All approaches use constructive heuristics *greedy-D*, and thus column "D" of A-TS is the same as for *greedy-D* in Table 2.

The experiments show that the alternating tabu search A-TS leads to the best results, with the smallest diameters, on average, and highest probability of finding a feasible solution. The alternating local search A-LS leads to a similar average diameter as the weighted tabu search W-TS, but is significantly worse than A-TS in both diameter and achieving feasibility, failing to solve the largest instance with $n = 10000$ and $p = 160$. This confirms that the tabu search is effective. Both W-TS and A-TS have a similar chance of finding a feasible solution, but A-TS finds significantly better diameters, on average. This indicates that, while a weighted approach does well at improving balance, an alternating strategy and techniques like the sp-escape play an important part in reducing the diameter. The number of iterations per second of A-LS is as expected the highest, since the search stops at the first local minimum, followed by A-TS and W-TS. The number of iterations of A-TS is higher for small p but decreases faster as p grows, while the number of iterations for A-LS and W-TS appears to be independent of p.

Table 4: Comparison to existing methods.

	n	p	Our method			RME-16		
			D(%)	Iter.	p. it.	D(%)	Phases	p. ph.
DS	500	10	**0.96**	23,112	0.29	7.46	375	0.98
DT	500	10	**0.03**	99,520	1.00	4.43	433	1.00
	1,000	5	**0.13**	24,429	0.99	1.08	96	1.00
	1,000	10	**0.03**	20,775	0.89	11.41	100	1.00
	1,000	16	**1.21**	10,787	0.43	22.07	87	1.00
	2,500	12	**0.47**	13,085	0.99	16.86	28	1.00
	2,500	25	**1.15**	10,168	0.77	34.00	23	1.00
DL	2,500	40	**6.41**	6,276	0.21	40.56	17	0.84
	5,000	25	**1.63**	7,767	0.97	38.91	7	1.00
	5,000	50	**1.93**	5,200	0.58	48.44	3	1.00
	5,000	80	**7.50**	3,697	0.05	109.27	2	0.14
	10,000	50	**2.01**	3,406	0.92	50.47	1	1.00
	10,000	100	**5.29**	2,240	0.34	78.57	1	0.50
	10,000	160	**46.54**	1,786	0.01	—	0	0.00
	Average		**5.38**	16,589	0.60	35.66	84	0.82

4.5 Experiment 3: comparison with existing methods

This experiment compares our methods to the GRASP with path relinking (PR) proposed by [7] to solve the same problem. We refer the reader to their original paper for a more detailed description of their algorithms.

[7] have kindly made available to us all data sets and source code developed in their research. Because their implementation was done in MATLAB, we chose to reimplement their algorithms in C++ and compile them under the same environment as our own. In the reimplementation, we have used the "static" PR variant proposed by them, as it reportedly produces better results, on average. For a fair comparison, in the reimplementation we have used the optimization techniques we proposed in Section 3.4 to maintain diameters and connectivity dynamically. We have also adapted their PR algorithm, as we have found that it does not always maintain connectivity during the path relinking process: we simply stop the process once no more moves maintain connectivity. Further, we have found that ordering PR moves by expected increase to the objective function was more effective than the original approach, which orders moves lexicographically by node index. Both modifications have improved the overall performance. Finally, because the termination criterion of their algorithm is not a maximum time limit, but rather a fixed number of iterations, in our implementation we repeat their full algorithm while there is still time and, in the end, report the best intermediate solution.

Table 4 shows the results. Columns under "RME-16" refer to our implementation of [7]'s algorithm. As before, for each instance we report averages over 10 replications of 30 minutes of running time each. The first two rows show av-

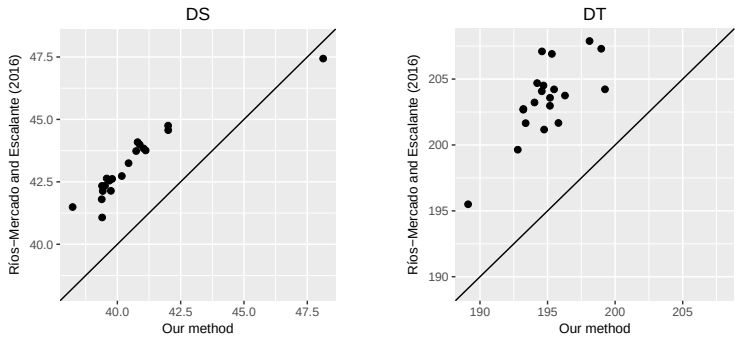


Fig. 3: Comparison of the final diameters achieved by our method and our implementation of [7]'s, for data set DS (left) and DT (right).

erage results for data sets DS and DT, respectively, and the following for data set DS. We report the average number of multistart iterations of our method ("Iter."), and the number of times [7]'s method was repeated under the time limit ("Phases"), as well as the probability of the methods finding a feasible solution in a single iteration ("p. it.") or phase ("p. ph."). Note that these values are not directly comparable, since one refers to multistart iterations and the other to replications of the full algorithm, and the former are two orders of magnitude more than the latter in the same time.

We can see that our method yields significantly more compact solutions in every instance size of data set DL, with better overall results in data sets DS and DT as well. For DL, the algorithm of [7] usually had a high probability of finding a feasible solution in a single phase, only failing to solve the largest instances with $n = 10000$ and $p = 160$. Our method had somewhat low probabilities of feasibility for DS as well as instances in DL with high p, but, given the sheer total number of iterations, it was always able to solve each instance at least once.

Figure 3 compares average final diameters obtained by both methods on each instance of data sets DS and DT. We can see that our method achieved lower values for all instances of data set DT, and for all but one instance (d500-03) of data set DS. In fact, this was the only case where their method consistently achieved better results in all replications. This could indicate that our method has a hard time exploring some parts of the search space, even in small instances.

5 Concluding remarks

We have proposed a multistart heuristic that uses an alternating tabu search strategy to solve a commercial districting problem. Our method differs from previous approaches because it improves each optimization criterion separately through a customized tabu search, as opposed to using a composite fitness function that assigns weights to each objective. We have performed experiments on two data sets from the literature, and one proposed in this paper, with a wider

range of instance sizes. The results confirmed the effectiveness of the different components of our method and showed that it has a high probability of finding feasible solutions of good quality. Our method is also competitive when compared to existing approaches in the literature, significantly improving state of the art upper bounds in almost all cases.

As a final note, we believe the proposed alternating search strategy can be applied without much change to other districting problems that model compactness as diameters, as [27], as well as related grouping problems such as the maximum dispersion problem [28], whose objective has a dual correspondence to the diameter.

Acknowledgments

This research was supported by the Brazilian funding agencies CNPq (grant 420348/2016-6), FAPEMIG (grant TEC-APQ-02694-16) and by Google Research Latin America (grant 25111). We would also like to thank to support of the Fundação de Desenvolvimento Científico e Cultural (FUNDECC/UFLA).

References

1. Ricca, F., Scozzari, A., Simeone, B.: Political Districting: From classical models to recent approaches. Annals of Operations Research 204(1), 271–299 (2013)
2. Ricca, F., Simeone, B.: Local search algorithms for political districting. European Journal of Operational Research 189(3), 1409–1426 (2008)
3. Bozkaya, B., Erkut, E., Haight, D., Laporte, G.: Designing new electoral districts for the city of Edmonton. Interfaces 41(6), 534–547 (2011)
4. Bação, F., Lobo, V., Painho, M.: Applying genetic algorithms to zone design. Soft Computing 9(5), 341–348 (2005)
5. Ríos-Mercado, R.Z., Fernández, E.: A reactive GRASP for a commercial territory design problem with multiple balancing requirements. Computers & Operations Research 36(3), 755–776 (2009)
6. Lei, H., Laporte, G., Liu, Y., Zhang, T.: Dynamic design of sales territories. Computers & Operations Research 56, 84–92 (2015)
7. Ríos-Mercado, R.Z., Escalante, H.J.: GRASP with path relinking for commercial districting. Expert Systems with Applications 44(September 2015), 102–113 (2016)
8. Camacho-Collados, M., Liberatore, F., Angulo, J.M.: A multi-criteria Police Districting Problem for the efficient and effective design of patrol sector. European Journal of Operational Research 246(2), 674–684 (2015)
9. Steiner, M.T.A., Datta, D., Steiner Neto, P.J., Scarpin, C.T., Rui Figueira, J.: Multi-objective optimization in partitioning the healthcare system of Parana state in Brazil. Omega 52, 53–64 (2015)
10. Blais, M., Lapierre, S.D., Laporte, G.: Solving a home-care districting problem in an urban setting. Journal of the Operational Research Society 54(11), 1141–1147 (2003)
11. Gliesch, A., Ritt, M., Moreira, M.C.O.: A genetic algorithm for fair land allocation. In: Genetic and Evolutionary Computation Conference. pp. 793–800. ACM Press (2017)

12. Kalcsics, J.: Districting Problems. In: Location Science, pp. 595–622. Springer (2015)
13. Salazar-Aguilar, M.A., Ríos-Mercado, R.Z., Cabrera-Ríos, M.: New Models for Commercial Territory Design. Networks and Spatial Economics 11(3), 487–507 (2011)
14. Salazar-Aguilar, M.A., Ríos-Mercado, R.Z., González-Velarde, J.L.: GRASP strategies for a bi-objective commercial territory design problem. Journal of Heuristics 19(2), 179–200 (2013)
15. Feo, T.A., Resende, M.G.C.: A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters 8(2), 67–71 (1989)
16. Butsch, A., Kalcsics, J., Laporte, G.: Districting for Arc Routing. INFORMS Journal on Computing 26(October), 809–824 (2014)
17. Glover, F.: Future paths for integer programming and links to artificial intelligence. Computers & Operations Research 13, 533–549 (1986)
18. Erkut, E., Ülküsal, Y., Yeniçerioğlu, O.: A comparison of p-dispersion heuristics. Computers & Operations Research 21(10), 1103–1113 (1994)
19. Tarjan, R.E.: A note on finding the bridges of a graph. Information Processing Letters 2(6), 160–161 (1974)
20. Łącki, J., Sankowski, P.: Optimal Decremental Connectivity in Planar Graphs. Theory of Computing Systems 61(4), 1037–1053 (2016)
21. King, D.M., Jacobson, S.H., Sewell, E.C., Cho, W.K.T.: Geo-Graphs: An Efficient Model for Enforcing Contiguity and Hole Constraints in Planar Graph Partitioning. Operations Research 60(5), 1213–1228 (2012)
22. Shamos, M.I.: Computational Geometry. Ph.D. thesis (1978)
23. Har-Peled, S.: On the Expected Complexity of Random Convex Hulls pp. 1–20 (nov 2011), `http://arxiv.org/abs/1111.5340`
24. Andrew, A.M.: Another efficient algorithm for convex hulls in two dimensions. Information Processing Letters 9(5), 216–219 (1979)
25. Overmars, M.H., van Leeuwen, J.: Maintenance of configurations in the plane. Journal of Computer and System Sciences 23(2), 166–204 (1981)
26. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 3, 43–58 (2016)
27. Chou, C., Kimbrough, S.O., Sullivan-Fedock, J., Woodard, C.J., Murphy, F.H.: Using Interactive Evolutionary Computation (IEC) with Validated Surrogate Fitness Functions for Redistricting. In: Genetic and Evolutionary Computation Conference. pp. 1071–1078 (2012)
28. Fernández, E., Kalcsics, J., Nickel, S.: The maximum dispersion problem. Omega 41(4), 721–730 (2013)