A hybrid heuristic for districting problems with routing criteria

Alex Gliesch, Marcus Ritt
Universidade Federal do Rio Grande do Sul
Porto Alegre, Brazil
{alex.gliesch,marcus.ritt}@inf.ufrgs.br

Arthur H. S. Cruz, Mayron C. O. Moreira
Universidade Federal de Lavras
Lavras, Brazil
arthur.cruz@estudante.ufla.br,mayron.moreira@ufla.br

***Abstract*—In this paper we consider districting problems with routing criteria. Such problems arise when clients in a commercial district need to be attended daily and the time to visit them is limited, for example by working hours of retail businesses. Routing costs are modeled by traveling salesman tours starting from predetermined depot locations. Besides having low routing costs, districts must also satisfy balancing (e.g. of total demand or workload) and contiguity constraints, and may also be required to be geographically compact. We propose a general heuristic for solving such problems. It iteratively builds new solutions by a greedy constructive heuristic, and then improves them by a hybrid alternating tabu search strategy. We further propose an efficient scheme for maintaining high-quality routes under small modifications to districts, for example incurred by neighborhood search operators. In extensive computational experiments we study the different components of our method, and show that it is effective in treating a number of problem variants with different constraints.**

***Index Terms*—districting, routing, heuristics**

I. Introduction

Let $G = (V, A)$ be an undirected planar graph of n vertices with distances d_{ij} on edges $a = \{i, j\} \in A$. In districting problems we want to partition the set of vertices or *basic units* into p disjoint districts $D_1, \ldots, D_p$ such that $\bigcup_{i\in[p]} D_i = V$. Districting problems arise in many practical applications, including the planning of public services (e.g. police or fire stations [1]–[3], waste collection [4], health care systems [5]–[7]), commercial [8], [9] and political districts [10], [11], or fair land allocation in agrarian reform projects [12]. It is almost always assumed that districts are connected, i.e. for each $i \in [p]$ the subgraph induced by D_i is connected. Basic units generally represent geographical entities such as city blocks or contiguous land parcels. In most applications the basic units have attributes such as the total population or product demand that must be balanced evenly among the districts. In general we assume that each vertex $v \in V$ has a set of a attributes $w_v^1, \ldots, w_v^a$ and each attribute $i \in [a]$ has a tolerance τ_i. For attribute $i \in [a]$ let $W^i(D) = \sum_{v\in D} w_v^i$ be its total value on subset $D \subseteq V$, and $\bar{w}^i = W^i(V)/p$ its mean value per district. We call a district $D \subseteq V$ *balanced* if $b_i(D) = |W^i(D) - \bar{w}^i|/\bar{w}^i \leq \tau_i$ for each attribute i, i.e., the absolute relative deviations $b_i(D)$ of $W^i(D)$ from the mean $\bar{w}^i$ are not larger than τ_i. In practice the number of attributes a rarely exceeds 3 and typical tolerances are about $1\,\%$ to $5\,\%$.

Besides being connected, geometrically compact district shapes (i.e. not too long, too narrow or having irregular borders) with respect to given vertex positions $p_v \in \mathbb{R}^2, v \in V$ are desirable. Compactness serves several purposes, such as facilitating district traversal or mitigating gerrymandering, in the case of electoral districts. A formal definition of compactness that models human preferences well is difficult, and many metrics have been proposed in the literature [13]. Among the most common are the *diameter* $\overline{d}(D) = \max_{i,j\in D} |p_i - p_v|_2$, i.e. the maximum distance between two vertices of a district, the total sum of distances $\mathrm{PM}(D) = \min_{c\in D} \sum_{i\in D} |p_i - p_c|_2$ to a district center c, also known as *p-median* (or moment of inertia if distances are weighted by attribute values), or the maximum distance $\mathrm{PC}(D) = \min_{c\in D} \max_{i\in D} |p_i - p_c|_2$ to district a center c, also known as *p-center*. In this paper we consider the p-median model. Its main advantage is that it accounts for the compactness of all districts locally, even if one of them is highly dispersed, whereas maximum-based approaches only look at the most dispersed district. The p-median function is also related to connectivity: [14] found that optimal p-median solutions without connectivity constraints are often connected nonetheless. Further, from our earlier experience we have found the p-median function to approximate other compactness measures much better than it can be approximated by them.

Let us now consider routing costs. A route attending district i is modeled by a shortest traveling salesman (TSP) tour which starts and ends at a *depot* $h_i \in V$, and visits all vertices $v \in D_i$ in an order such that the total tour length $\mathrm{R}(D_i)$ is minimal. These tours represent, for example, routes used by delivery trucks in the distribution of commercial products to businesses. Real-world routes are often constrained by time availability, fuel or vehicle storage capacity, and so a budget constraint $\mathrm{R}(D) \leq \overline{R}$ on the length of each tour is given, where the routing budget $\overline{R} > 0$ is part of the problem input. Here, graph G is assumed to be connected such that d_{ij} is defined for all $\{i, j\} \notin A$ as the shortest path distances. This means that, in practice, routes may visit some

Our research has been supported by the funding agencies CNPq (grant 420348/2016-6), by FAPEMIG (grant TEC-APQ-02694-16), by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, by the Google Research Latin America (grant 25111). We also like to thank the support of the Fundação de Desenvolvimento Científico e Cultural (FUNDECC/UFLA).

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

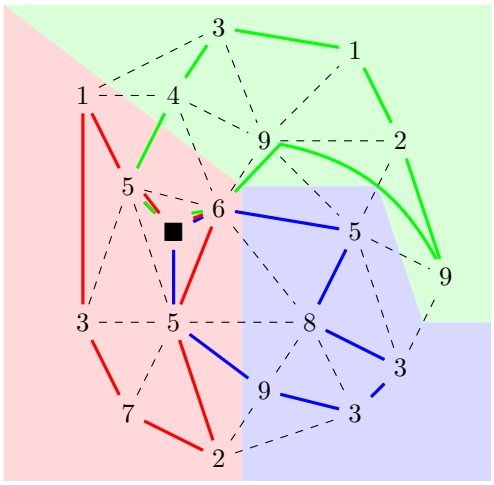

Fig. 1. An example of a districting problem with routes. There are $p = 3$ districts; vertices v are labeled with their single attribute w_v^1. The districts are connected and balanced with a tolerance $\tau_1 = 2.5\,\%$. The central station is marked by a black square, edges are dashed, routes are shown in different colors. Note how the routes from the central station pass over vertices from different districts and the curved connection is the shortest intra-district path going over the green vertex labeled 2.

vertices more than once or even make a detour outside of their assigned district. A problem variant may exclude such detours by requiring routes to stay within district boundaries, in which case distances d vary according to each particular districting plan. This variant is not considered here, however, since maintaining shortest path matrices for each district is computationally too expensive [15].

Depot vertices $h_1, \ldots, h_p$ are fixed to each district. For simplicity we consider depots to be locally unique, i.e., $h_i \neq h_j \ \forall i, j \in [p]$, although some variants may also consider a single global depot for all districts, or ignore depots altogether. The algorithms proposed here can also handle these variants, however, and in Section III-F we briefly examine them in an experiment. Note that not necessarily $h_i \in D_i$, since in our model we allow cross-district routes. An example of a routing plan with a global depot can be seen in Figure 1.

The following mathematical model M summarizes the problem considered in this paper:

$$\begin{aligned} \min_{P \in \left\{ {V \atop p} \right\}} \quad & \mathrm{Z} = \lambda \mathrm{R}(P)/\mathrm{U}_r + (1-\lambda)\mathrm{PM}(P)/\mathrm{U}_c && \text{(M)} \\ \text{subject to} \quad & b_i(D) < \tau_i, && \forall D \in P, i \in [a], \\ & \mathrm{R}(D) \leq \overline{R}, && \forall D \in P, \\ & \text{connected}(D), && \forall D \in P, \end{aligned}$$

where $\left\{ {V \atop p} \right\}$ is the set of all p-partitions of the basic units V. The objective function Z is a convex combination of the total p-median compactness $\mathrm{PM}(P) = \sum_{D \in P} \mathrm{PM}(D)$ and routing costs $\sum_{D \in P} \mathrm{R}(D)$ of the districts, where $\lambda \in [0, 1]$ is a weight. In order to keep $\mathrm{PM}(P)$ and $\mathrm{R}(P)$ in the same order of magnitude, thus allowing a more intuitive choice of λ, both are normalized by upper bounds U_r and U_c, which are explained in Section II-F. The first set of constraints ensures vertex attributes are balanced across districts, and the second enforces budgets on route lengths. To reflect real-world scenarios, we assume routing costs are always either handled as a constraint by setting $\lambda = 0$, or as an objective, by setting $\overline{R} = \infty$. We consider here a single objective rather than a multi-objective model since it is more common in the literature.

In this paper we propose a flexible heuristic solver for addressing districting problems with routing costs. It expands on previous methods that have shown to be effective, and can be adapted to include different constraints and optimization criteria. We further propose dynamic algorithms to maintain good TSP tours efficiently under small modifications to districts, and show that the tours obtained are often close to optimal. We next present some related work, and in Section II explain the heuristic solver proposed. Section III presents an experimental analysis of the proposed methods. We conclude in Section IV.

A. Related work

There exists a rich literature on districting problems, and we focus here on combined districting and routing. A thorough overview of districting problems in general can be found in [16, ch. 23].

Two related problems with an overlap in solution techniques are vehicle routing and location routing. In vehicle routing problems, clients need to be attended from a central depot by vehicles of a limited capacity, minimizing the total travel distance. For a fixed fleet of p vehicles the routes partition the clients into p regions that are attended by the same vehicle. Different from our problem, there are no constraints concerning connectivity or compactness of the resulting regions, and no balancing requirements except from the demand upper bound given by the capacity of the vehicle. In location-routing problems, on the other hand, the focus lies on facility location: one must open a set of facilities and assign clients to them. In addition, routes must be constructed for the clients each facility attends. Vehicles are usually capacitated, and thus several routes may be required to attend the clients assigned to a facility [17], [18]. In contrast, in combined districting and routing we do not consider opening costs and assume the clients of each district can always be visited with a single tour.

For the routes in each district three models are common: the clients in all districts may have to be attended from a single depot (e.g. in commercial districting, [19]), or from a depot assigned to the district (e.g. in salt spreading, [20]), or with open routes (e.g. in waste collection where routes finish at a dump [4]). Usually an upper bound (or budget) on each tour length is imposed, and in rare cases also on the total length of all tours. Some models also consider arc-routing (e.g. in salt spreading and patrolling) instead of vertex-routing, which we do not consider here.

Many authors recognize the importance of combined districting and routing. [21], for example, consider the problem of designing districts for a Meals-On-Wheels service, subject to constraints on capacity, total service time, and maximum

diameter. However, to simplify the model, the total service time does not consider traveling time, and the districts are not required to be connected. Similarly, [4] note that after creating balanced municipal waste districts, routes that satisfy time restrictions must be constructed, but use an approximation by a nearest neighbor tour to simplify the problem.

Several authors consider combined districting and stochastic vehicle routing, usually for a homogeneous fleet of p vehicles [22]–[24]. These problems are modeled as two-stage stochastic problems with recourse actions, where in the first stage customers are partitioned into at most p connected districts, each to be attended by a single vehicle, with the goal to minimize the expected routing cost. Only after districts have been defined the demands of the customers are revealed, and for each district a vehicle routing problem is solved in the second stage. [22] propose a solver using tabu search, which shifts and swaps vertices among districts, while maintaining connectivity. The second-stage routing costs are estimated heuristically by fixing a permutation of the clients, which can be optimally segmented for a known demand, and computing the expected value over all demands. [23] propose a large neighborhood search heuristic, and approximate route length by the Beardwood-Halton-Hammersley formula [25]. [24] extend the problem to unknown customers and optimize number, compactness, similarity to existing plans, and balance of districts with a multi-objective evolutionary algorithm. All these approaches solve the problem from a vehicle routing perspective, and consequently none of them enforce district connectivity.

[19] consider an application from the bottled beverage industry, where p connected districts must be found subject to balancing the number of customers and the total demand, and subject to routing budgets. The objective is to minimize the maximum diameter compactness among all districts. The authors propose a GRASP heuristic for solving the problem. The heuristic first grows districts from seed vertices by randomly selecting adjacent vertices that keep the diameter low, until exceeding $70\,\%$ of the upper bound of an attribute. In a second step, the construction considers also attribute balancing and the frequency of assigning adjacent vertices to the same district. The complete solution is improved by a local search that shifts vertices to neighboring districts, minimizing a weighted sum of diameter, routing cost, and imbalance. In this application, all distances are intra-district shortest path distances.

II. Solution approach

The solution approach we propose builds upon the hybrid heuristic of [26], which is a current state-of-the-art method for solving districting problems with p-median, p-center and diameter objectives and no routing criteria. In the subsections that follow we explain this method in broad terms, and focus on how it has been adapted to include routing criteria, both as objectives and as constraints.

Algorithm 1 Main heuristic method

```
repeat
    S ← selectInitialSolution()
    for i ∈ [A_max] do
        S' ← optimizeObjectiveFunction(S)
        S ← optimizeConstraints(S', S)
        if E(S) > 0 then
            break
until time limit or maximum iterations reached
return best feasible solution encountered
```

A. The hybrid alternating search heuristic

In the following, let a *solution* be a p-partition of V. Further, let $\mathrm{E}(S) = \mathrm{E}_b(S) + \mathrm{E}_r(S)$ be the sum of constraint violations to the balance constraints $\mathrm{E}_b(S) = \sum_{D\in S}\sum_{i\in[a]}\max\{0, b_i(D) - \tau_i\}$ and to the routing budget constraints $\mathrm{E}_r(S) = \sum_{D\in S}\max\{0, (\mathrm{R}(D) - \overline{R})/\mathrm{R}(D)\}$ of solution S, and let $\mathrm{Z}(S)$ be the objective function value of S as given by model M. Here we normalize $\mathrm{E}_r(S)$ by $\mathrm{R}(D)$ in order to maintain both $\mathrm{E}_r(S)$ and $\mathrm{E}_b(S)$ in the same order of magnitude, therefore removing biases towards optimizing one or the other.

Algorithm 1 outlines the main method. It uses a multistart approach akin to a GRASP heuristic [27]. It iteratively obtains a new solution through a greedy randomized constructive heuristic, and then optimizes that solution by an improvement strategy. This is repeated until a termination criterion is reached, either a time limit or a maximum number of iterations, and the best feasible solution encountered is returned.

The improvement strategy alternates between two neighborhood search procedures optimizeObjectiveFunction and optimizeConstraints, that optimize the objective function Z and minimize constraint violations E, respectively. Sections II-C and II-D further detail them. Each alternating iteration except the first starts with a feasible solution, i.e. having $\mathrm{E}(S) = 0$, and optimizes first Z then E. Procedure optimizeConstraints does not allow neighboring moves that cause the objective value $\mathrm{Z}(S')$ to fall below the value $\mathrm{Z}(S)$ of the previous iteration, in order to ensure the progress made to Z is not lost while optimizing constraints. Procedure optimizeObjectiveFunction, however, may work with infeasible solutions by allowing E to increase. Since it already begins at a feasible solution and usually does not modify it much, in practice the additional violations it causes can be easily overcome by optimizeConstraints next. The alternating phase stops whenever constraints cannot be satisfied for a certain Z level, or after a maximum $A_{\max}$ iterations, where $A_{\max}$ is a parameter.

District connectivity is always satisfied, as initial solutions are connected by design and the local optimization procedures operate on neighborhoods allowing only moves that keep connectivity.

B. Generating initial solutions

Initial solutions are obtained using the greedy constructive algorithm of [26], slightly modified to include routing costs.

It has two phases. First, p seed vertices, one per district, are selected through a randomized p-dispersion algorithm. Starting from an initial seed chosen randomly, it iteratively selects the next seed so as to maximize the minimum distance to all others. In the second phase, districts are "grown" from the seeds by iteratively assigning *fringe* vertices until a complete partition is obtained. Fringe vertices to district D are unassigned vertices that share an edge with some vertex of D. At each iteration, the algorithm greedily selects the district $i \in [p]$ with the least sum of attributes $\sum_{j\in[a]} b_j(D_i)$, and chooses from its fringe the vertex whose assignment would lead to the smallest objective function value Z. If routing budget constraints are enabled, i.e. $\overline{R} \neq \infty$ and $\lambda = 0$, we exceptionally use a value $\lambda = 0.5$ when computing Z here. This gives a preference to initial solutions with shorter TSP tours, even if routing costs are only considered as budget constraints, and allows us to maintain the original greedy algorithm unchanged. In this phase we also do not consider depots when computing R, for reasons explained below. A caching mechanism is further used that maintains the best possible assignment of each districts fringe. This mechanism remains valid with the addition of routing criteria.

1) Matching depots to districts: Though the method above considers route costs in the selection of candidate vertices to be included, the selection of initial seeds is agnostic of depot locations, meaning there is no guarantee that a district will be near its depot. This, of course, can have a significant impact in final routing costs, since a far away depot will induce additional travel times, and the later local search-based methods are unlikely to restructure the solution enough to mitigate this.

Note, however, that districts are *anonymous*: they all share the same balancing constraints, routing budgets and do not have location requirements. This means that any assignment of depots to districts is valid, and we can choose one such that total routing costs are minimized. We do this by solving a minimum cost bipartite matching subproblem on a graph of parts $v_1, \ldots, v_p$ and $u_1, \ldots, u_p$, where the cost of an edge (v_i, u_j) is the length of a TSP tour over vertices $D_i \cup \{h_j\}$, where D_i is the i-th district of the greedily constructed solution. The optimal matching can be obtained in $O(p^3)$ time by the algorithm of [28], and each match (v_i^*, u_j^*) corresponds to depot h_j servicing district D_i.

In practice, computing p^2 optimal TSP tours to build the above graph is usually too expensive computationally, and so we use 2-opt local searches, which we denote as simply 2OPT. For district $i \in [p]$ let T_i be a permutation of D_i representing its current TSP tour, and let $|\mathrm{T}_i|$ be its cost. We first optimize each $\mathrm{T}_i, i \in [p]$ by 2OPT. Next, for each pair $i, j \in [p]$ we compute T_i^j as tour T_i with h_j inserted to it by a greedy insertion heuristic, explained further in Section II-E1. We then optimize T_i^j once again with 2OPT, and set the cost of edge (i, j) to $|\mathrm{T}_i^j|$. Since T_i^j is only a slight modification over T_i to include h_j, in practice it is already close to a local minimum and 2OPT takes only a few iterations.

2) Filtering: Following [26] a simple filtering mechanism is used to skip unpromising initial solutions. At each multistart iteration it generates p new solutions and adds them to a pool P, from which the best solution with respect to (Z, E), compared lexicographically, is returned. Then, the worst $\max\{0, |P| - 2p\}$ solutions from P are removed from the pool.

C. Optimizing the objective function

We optimize the objective function by a tabu search [29]. Given a solution, it iteratively applies a neighborhood operator leading to a neighbor of minimal Z. If multiple such neighbors exist, we choose one randomly. The neighborhood consists of two moves: *shifts* $u \to k$ that move a vertex u from its previous district to district k, and *swaps* $u \leftrightarrow v$ that swap the current districts of vertices u and v. We first try all possible shifts, and consider swaps only if no improving shift was found. This is done for performance, as there are roughly $O(\sqrt{n})$ possible shifts and $O(n)$ swaps and, in practice, improving shifts are found in over 80% of cases. This is a change to the original method of [26], which only uses shifts for the p-median objective, since in early experiments we identified that swaps play an important role in minimizing routing costs.

After a move is done, its corresponding vertices are declared *tabu* (i.e., cannot be moved) for t iterations, where the tabu tenure t is a parameter. Note that moves are not necessarily improving. The search stops after $I_{\max}$ iterations without improvement, where $I_{\max}$ is a parameter, and returns the best intermediate solution. Moves which break district connectivity are discarded. We refer the reader to [26] for details on efficient dynamic data structures for connectivity and p-median values, as well as how neighbors can be cached to reduce the number of evaluations by a factor p; we omit these here for lack of space.

D. Optimizing constraint violations

Constraint violations E are optimized by a series of tabu searches of the same structure as above, but guided by E rather than Z, and with an additional stopping criterion when $\mathrm{E} = 0$ (i.e., when a feasible solution is found). To avoid deteriorating the objective value too much, the tabu searches are constrained by a parameter Z_m that defines a maximum allowed objective function value, thus prohibiting moves that would cause Z to exceed it. These tabu searches are embedded into a binary search strategy that looks for the smallest $\mathrm{Z}^* \in [\mathrm{Z}_l, \mathrm{Z}_u]$ such that a tabu search constrained by $\mathrm{Z}_m \leq \mathrm{Z}^*$ can find a feasible solution, where Z_l is the objective value of the solution given to optimizeConstraints, and Z_u is the objective value of the best feasible incumbent.

E. Computing routing costs

Because the TSP is NP-complete [30], maintaining optimal tours $\mathrm{T}_1, \ldots, \mathrm{T}_p$ throughout the entire algorithm is computationally too expensive for practical instance sizes. Therefore, we rely on heuristics. Routing costs are evaluated at several

different stages of the algorithm, each with distinct requirements regarding performance and precision (i.e., gap from the optimal tour value). For example, routing costs of neighbors during the tabu search must be computed very fast, so that they do not become a bottleneck. On the other hand, at the end of each multistart iteration we update the globally best solution, and so a high-precision heuristic is desired, even if it is slower. We have identified five different levels of increasing precision and decreasing performance requirements for the evaluation of routing costs:

1) When evaluating a candidate shift or swap move during tabu search. This action is performance-critical, since it happens at the inner-most loop of the algorithm. All other attributes of solutions, such as balancing constraints, p-median values, or connectivity can be kept dynamically in constant time, and so a TSP heuristic here must ideally also be computed in $O(1)$, else it could dominate the running time of the algorithm.
2) When executing a tabu search move, i.e., applying a shift or swap to a solution, or when assigning fringe vertices during greedy construction. This is executed once every $O(n)$ candidate move evaluations, on average, and so the cost of a more expensive algorithm here is amortized over the evaluations.
3) At the end of each tabu search. This happens after at least $I_{\max}$ moves have been made. A better TSP heuristic here helps mitigate possible deviations from optimality that may have been propagated by successively using a simpler heuristic earlier.
4) At the end of each multistart iteration. In practice this happens after around 25 tabu searches, on average. Besides reducing imprecisions from lower levels, a more precise algorithm here helps to select the best solution among all multistart iterations.
5) When the algorithm halts. Here, we always recompute tours optimally.

In the literature there are several TSP heuristics with varying trade-offs between effectiveness and running times [31], [32]. For our purposes here we consider three: a constant-time greedy insertion heuristic GU, which we propose in Section II-E1, 2-opt local searches (2OPT), which are easy to implement and generally fast if the initial solution is already optimized, and the well-known Lin-Kernighan [33] heuristic (LKH), which is among the most effective and widely used heuristics for the TSP today. For the LKH we use the implementation in the Concorde solver [34] and consider two variants, LK_1 and LK_2, with different parameter configurations: LK_1 uses stallCount $= 1000$ and maxKicks $= n/5$, where stallCount is a stagnation parameter and maxKicks defines the number of perturbation steps, while LK_2 uses stallCount $= 5000$ and maxKicks $= n$. Variant LK_1 is aimed to be faster at the cost of slightly worse tours, while LK_2 invests more time in finding better solutions.

Let $\mathrm{A}_1, \ldots, \mathrm{A}_5$ be parameters that define the algorithms used in levels 1-5 above. We have fixed A_1 to be GU, as it is imperative that evaluations at the lowest level be highly efficient. Because the best choice of A_2 and A_3 was not obvious, we considered $\mathrm{A}_2 \in \{\mathrm{GU}, \mathrm{2OPT}\}$ and $\mathrm{A}_3 \in \{\mathrm{2OPT}, \mathrm{LK}_1\}$, and calibrate them experimentally in Section III-B. For A_4 we use LK_2, since an effective heuristic is needed to select the best multistart solution. For the exact algorithm A_5 we use Concorde. Algorithms $\mathrm{A}_3, \mathrm{A}_4$ and A_5 are warm-started by the existing, already optimized tours, which improves their performance considerably.

Note that, since heuristic solutions for the TSP are upper bounds on the optimal tours, if any heuristic tour satisfies budget constraints then the corresponding optimal tour also does.

1) Dynamic greedy updates to TSP tours: In order to update TSP tours efficiently upon vertex insertions and removals, we propose the following greedy heuristic GU. It assumes the given tours have already been optimized by a more expensive TSP heuristic, and thus tries to maintain their original order as much as possible.

When removing vertex v_r from tour T_i, we simply remove it and maintain the rest of the tour intact, i.e., we link the vertices that come before and after v_r. Because T_i must always include depot h_i, however, if $v_r = h_i$ we simply do not remove it.

When adding vertex v_a to tour T_i, we choose a location $j \in [|D_i| + 1]$ in the tour and insert v_a after the j-th vertex, maintaining the rest of the tour intact. One way to select j would be to test all possible locations and take the one leading to the shortest tour. Since there are $O(n/p)$ possible candidates, however, this is too slow for our purposes, as GU should ideally take constant time. We therefore only consider locations directly before or after neighbor vertices of v_a that are in D_i. The constructive algorithm and tabu searches only allow moves that keep connectivity, thus v_a is guaranteed to have at least one such neighbor. Since instance topologies emulate real-life planar domains, in practice optimal detours to visit v_a tend to have v_a adjacent to one of its neighbors. Because G is planar its average vertex degree is bounded by 6 (see e.g. [35, 4.2.10]), and so examining v_a's neighbors takes amortized constant time. Again, if $v_a = h_i$, we do nothing, since in this case v_a is already present in T_i.

F. Upper bounds

We obtain upper bounds by generating an example solution S_d by a very simple algorithm, and setting $\mathrm{U}_c = \mathrm{PM}(S_d)$ and $\mathrm{U}_r = \mathrm{R}(S_d)$. First, p initial seed vertices, one per district, are selected uniformly at random. Then, while there are unassigned vertices, we iterate cyclically over the districts, each time assigning to the current district the vertex of least index on its fringe, if it exists. Routes are updated by GU at every assignment, and we run 2OPT on each district at the end. Since upper bounds are only used for normalization in the interest of choosing the λ weight, solution S_d does not need to be feasible. In practice, final solution values deviate on average 4.4% from U_c, and 21.3% from U_r.

III. Computational experiments

In this section we report on experiments that assess the effectiveness of the proposed methods. We have implemented our algorithms in C++ and compiled them with GCC 7.2 and maximum optimization. All experiments were executed on a PC with an Intel i7 930 CPU processor and 16GB of main memory. For each test, only one core was used. Following [26], We use parameters $t = 1.5p$, $A_{\max} = 100$ and $I_{\max} = 100$ as recommended by [26], since in practice we have found this setting to work relatively well with routing criteria as well. Each run was limited to 1000 multistart iterations and 10 minutes, and uses a fixed random seed. The time used by A_5 to compute the exact tours at the end is not counted towards the time limit, and is thus omitted from the tables below. The source code, instances and full results of the experiments are available from the authors upon request.

A. Instances

We use four instance sets from the literature, described below. They come from a commercial districting context, where products must be distributed to customers in an urban area. This application of districting has been extensively studied in the recent years, and several variant models have been proposed for it [8], [9], [19], [36], [37]. Instance set RF, proposed by [36], has two instance classes that differ in the distribution of attribute values, each having 20 instances of size $n = 500$ and $p = 10$. Instance set SRC, proposed by [14], contains 20 instances of each size $(n, p) \in \{(60, 4), (80, 5), (100, 6)\}$ and 10 of each size $(n, p) \in \{(150, 8), (200, 11)\}$, that reflect instance sizes treatable by their exact algorithm. Instance set RS, proposed by [19], has two instance classes, with 30 instances of size $n = 1000$ and $p = 10$ and 15 instances of size $n = 1000$ and $p = 40$, respectively. Finally, instance set GRM, proposed by [9], has the largest instances with four of each size $n \in \{1000, 2500, 5000, 10000\}$ and $p \in \{n/200, n/100, n/62.5\}$; we have excluded, however, instances of size $n \geq 5000$ from this set, as they proved to be too difficult when routing is considered.

All instances above have three balancing attributes, namely: the number of customers, the product demand and the expected visitation time of each basic unit. To be consistent with most previous works, we used a tolerance $\tau_i = 0.05$ for $i \in [3]$. The instances provide plane coordinates for each unit, and we compute the graph distances d_{ij} by an all-pairs shortest paths algorithm prior to the execution of our method. The time to compute d was negligible.

Because these instances were originally proposed for models without routing criteria, no district depots are given. In real-world scenarios depots tend to be more or less dispersed over the input graph, and so we defined p depots for each instance by executing the randomized p-dispersion algorithm of Section II-B with a fixed random seed. We define the routing budgets $\overline{R}$ of each instance in a separate experiment, reported in Section III-D.

TABLE I
Calibrating TSP algorithms A_2 and A_3.

Inst.	A_2	A_3	Z [rd.%]	iter.	t.
SRC	2OPT	2OPT	0.01694	1,000.0	70.2
	2OPT	LK_1	0.01231	1,000.0	76.3
	GU	2OPT	0.01682	1,000.0	65.0
	GU	LK_1	0.01382	1,000.0	71.1
RF	2OPT	2OPT	0.24025	373.2	600.0
	2OPT	LK_1	0.20466	390.5	600.0
	GU	2OPT	0.29308	403.3	600.0
	GU	LK_1	0.25944	416.5	600.0
RS	2OPT	2OPT	0.34796	164.8	600.0
	2OPT	LK_1	0.20894	162.5	600.0
	GU	2OPT	0.44179	176.4	600.0
	GU	LK_1	0.40115	171.6	600.0
GRM	2OPT	2OPT	0.34927	19.2	600.0
	2OPT	LK_1	0.25759	24.3	600.0
	GU	2OPT	0.30405	20.1	600.0
	GU	LK_1	0.44132	25.8	600.0

B. Experiment 1: influence of different TSP algorithms

In this experiment we calibrate algorithms A_2 and A_3 used to recompute district routes after neighboring moves and after tabu searches, respectively. We considered the choices $A_2 \in \{GU, 2OPT\}$ and $A_3 \in \{2OPT, LK_1\}$, as described in Section II-E. In the case $A_2 = 2OPT$, when applying a neighboring move we first execute GU to update the given tour, and then optimize it by a 2-opt local search. For this experiment we used $\lambda = 0.5$ and no routing budgets. Table I shows the results. For each instance set and parameter choice we report averages of the final objective value Z, shown as the relative deviation (in %) to the best known values, the number of multistart iterations (iter.) and the total running time in seconds (t.).

For all data sets the configuration $(A_2, A_3) = (2OPT, LK_1)$ achieved the best average Z values, despite being slightly slower when considering running times and iteration counts. This configuration includes the two heuristics of highest time-over-quality ratios, which suggests that investing more effort in finding better TSP routes during optimization pays off. The differences in performance between the four settings were not significant, since all allowed enough multistart iterations for feasible solutions to be found. Among the other three settings there are no clear losers or winners. In the experiments that follow, we have therefore fixed $(A_2, A_3) = (2OPT, LK_1)$.

Table II shows, for configuration $(2OPT, LK_1)$ and each instance class, the average tour size n/p and the ratios between tour lengths obtained at the five stages considered. Here, each column A_i/A_j displays the average difference, in %, between tour lengths found by algorithm A_i when it was executed on a tour previously computed by algorithm A_j.

We see that the proposed greedy update heuristic $A_1 = GU$ generally obtains results that are very close (under 0.08% difference, on average) to the local minima found by $A_2 = 2OPT$. This shows that, despite making some simplifications to be

TABLE II
RELATIVE DEVIATIONS OF TOUR LENGTHS BETWEEN THE DIFFERENT TSP ALGORITHMS USED.

Inst.	n	n/p	A_2/A_1	A_3/A_2	A_4/A_3	A_5/A_4
SRC	60	15.0	0.1098	1.6274	0.0035	0.0000
SRC	80	16.0	0.0999	1.7490	0.0026	0.0000
SRC	100	16.7	0.1085	1.7289	0.0032	0.0000
SRC	120	17.1	0.0998	1.7041	0.0028	0.0000
SRC	150	18.8	0.0983	1.9437	0.0040	0.0000
SRC	200	18.2	0.1066	1.9533	0.0044	0.0000
RF	500	50.0	0.0417	2.0347	0.0223	0.0004
RS	1000	75.0	0.0615	1.6006	0.0241	0.0013
GRM	1000	120.8	0.0224	1.1444	0.0361	0.0044
GRM	2500	123.6	0.0344	1.1705	0.0976	0.0047
Avg.		47.1	0.0783	1.6657	0.0201	0.0011

computable in constant time, GU can be an effective way to maintain tours dynamically. Looking at $n \geq 500$, we also see GU scales well for larger tours. The largest deviations are found between $A_3 = LK_1$ and $A_2 = 2OPT$, and are likely due to the large disparity in effectiveness between these two heuristics. This effect had already been observed earlier by [38]. Column A_4/A_3 shows only a small difference between LK_2 and LK_1, specially for smaller instances, which could indicate that the LKH stagnates quickly and additional restarts do not help. Looking at the last column, we see that LK_2 is always optimal for $n \leq 200$, and very close to optimal (less than 0.005%) for $n \geq 500$, and thus is usually successful in selecting the best multistart iteration.

C. Experiment 2: effect of λ

In this experiment we analyze the effects of using different values of λ when considering routing as objective. We consider values $\lambda \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$, and no routing budgets. At $\lambda = 0.0$ with routing costs disabled, for performance we do not update routes during neighborhood search. The same is done for PM when compactness is disabled at $\lambda = 1.0$. Table III shows the results. For each instance set and value of λ we report the average deviations (in $\%$) of the p-median value PM and the total routing costs R relative to the best known values, as well as the total number of multistart iterations (iter.).

We observe a clear progression in the quality of p-median values with decreasing λ; all best known values for p-median are found at $\lambda = 0$. For routing costs, on the other hand, we see an interesting phenomenon: the best R was consistently found not at $\lambda = 1.0$, but rather between $\lambda = 0.6$ and $\lambda = 0.8$. This can be better seen in Figure 2, which shows the progression of route length deviations as a function of λ, averaged over all instances. We believe this effect is due to p-median values being related to short TSP tours: geometrically compact districts are intuitively faster to traverse, and it is likely that PM has a better neighborhood landscape than R for shifts and swaps.

Looking at the number of iterations we see that variants $\lambda = 0$ and $\lambda = 1.0$ are the fastest, since they need not compute R and PM, respectively. There does not seem to be a clear trend in performance for the other values of λ.

TABLE III
EFFECT OF DIFFERENT VALUES OF λ IN THE OBJECTIVE FUNCTION.

Inst.	λ	PM [rd.%]	R [rd.%]	iter.
SRC	0.0	0.00	5.26	1,000.0
	0.2	0.33	5.32	1,000.0
	0.4	1.28	3.00	1,000.0
	0.6	2.68	1.43	1,000.0
	0.8	4.56	0.49	1,000.0
	1.0	7.07	0.23	1,000.0
RF	0.0	0.00	5.38	619.7
	0.2	0.60	3.51	384.8
	0.4	1.46	1.51	391.8
	0.6	2.68	0.92	383.1
	0.8	5.94	0.28	346.4
	1.0	11.88	0.57	679.8
RS	0.0	0.00	3.61	231.9
	0.2	0.93	3.56	157.1
	0.4	2.07	1.55	159.6
	0.6	4.14	0.58	163.6
	0.8	7.13	0.36	159.4
	1.0	11.14	1.01	262.3
GRM	0.0	0.00	3.45	25.7
	0.2	0.80	2.28	24.7
	0.4	2.22	0.79	25.1
	0.6	4.45	0.41	22.3
	0.8	8.23	0.66	17.4
	1.0	13.38	1.15	63.8

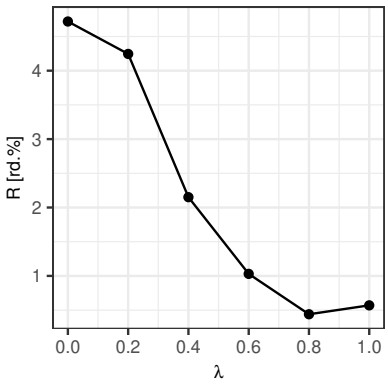

Fig. 2. Effect of λ on routing costs, averaged over all instances.

D. Experiment 3: determining routing budgets for instances

Since the instances considered do not specify routing budgets $\overline{R}$, we define them experimentally. Ideally $\overline{R}$ should not be so low as to make all instances infeasible, nor so high as to make them trivial by allowing even unoptimized tours to satisfy them.

We use the Beardwood-Halton-Hammersley formula [25] as a base for our choice of $\overline{R}$. It states that the length of an optimal TSP tour on n uniformly distributed vertices on surface of area A, with Euclidean distances, converges to $H =$

TABLE IV
CALIBRATING b_d FOR DETERMINING ROUTING BUDGETS PER INSTANCE.

Inst.	n	$b_d{}^*$	feas.	$\mathrm{R}^*/(p\overline{R})$
SRC	60	1.10	4.15	1.19
SRC	80	1.08	4.60	1.19
SRC	100	1.05	3.40	1.21
SRC	120	1.09	3.10	1.23
SRC	150	1.06	3.20	1.24
SRC	200	1.11	2.70	1.30
RF	500	1.01	2.68	1.26
RS	1000	0.89	3.53	1.15
GRM	1000	0.86	5.92	1.12
GRM	2500	0.91	3.25	1.19
Avg.		1.02	3.65	1.21

TABLE V
RESULTS FOR THE VARIANT WITH ROUTING BUDGET CONSTRAINTS.

Inst.	n	PM [rd.%]	R [rd.%]	t.	iter.	feas. [%]
SRC	60	1.64	5.68	30.6	1,000.0	51.3
SRC	80	1.14	5.34	49.5	1,000.0	43.3
SRC	100	0.99	7.03	57.6	1,000.0	35.1
SRC	120	1.25	8.12	70.7	1,000.0	27.3
SRC	150	1.22	8.76	94.5	1,000.0	27.8
SRC	200	0.82	9.45	117.0	1,000.0	23.9
RF	500	0.14	6.87	593.2	730.4	25.0
RS	1000	0.18	3.94	600.0	156.6	30.5
GRM	1000	0.12	3.83	600.0	64.9	38.4
GRM	2500	0.76	5.20	600.0	22.0	35.2
Avg.		0.83	6.42	281.3	697.4	33.8

$b_d/\sqrt{An}$ for large n and some constant b_d. Here we extrapolate the formula to p districts by setting $\overline{R} = H/p$, and use area A as the axis-aligned bounding box of all the vertices. Parameter b_d has been empirically determined to be about 0.714 [32]; however, since input graphs here are usually not complete and thus distances d not always Euclidean, we have found $b_d < 0.8$ to be too tight a constraint in almost all cases. Further, since instance sets use different generators for their topologies, and because instance difficulties also vary according to n and p, it is unlikely that a single b_d which works well for all cases exists.

For each instance, we have therefore considered values $b_d \in \{0.8, 0.85, \ldots, 1.25, 1.3\}$ and executed 10 multistart iterations of our algorithm with no time limit. In order to choose budgets that are challenging, for each instance we then selected the smallest $b_d = b_d{}^*$ for which at least 2 such iterations were feasible. Table IV reports, for each instance set and n, the average $b_d{}^*$ found, as well as the actual number of feasible iterations (feas.) obtained at $b_d = b_d{}^*$. We also report the average ratio between the mean tour length R^*/p of the best known solutions and the calibrated budget $\overline{R} = b_d{}^*/\sqrt{An}$.

We see that $b_d{}^*$ has a mean of 1.02, suggesting that routing costs for the instances considered deviate about 30% from the empirical value 0.714 of the Beardwood-Halton-Hammersley formula. The selected values $b_d{}^*$ found feasible solutions in 3.65 out of 10 iterations, on average, and in at most 5.92 iterations for GRM instances of $n = 1000$, indicating that the generated budget constraints were neither trivial nor impossible to satisfy. The difference between the chosen budgets and the tour lengths of the best known solutions was 20.8%, on average.

E. Experiment 4: routing budgets

In this experiment we solve our model with the routing budget constraints defined in the previous section, and use $\lambda = 0$ to disable routing costs in the objective function. Table V shows, for each instance set and n, the average deviations (in $\%$) of the p-median value PM and the total routing costs R relative to the best known values, as well as the time (t.), in seconds, the number of multistart iterations (iter.), and the percentage of multistart iterations that were feasible.

We see that average PM values stays within 0.83% of the best known values, but nonetheless did not match them. This suggests budget constraints were binding, as procedure optimizeConstraints likely was unable to make highly compact solutions feasible. Relative deviations of total routing costs R were high (6.42%, on average), since after satisfying budget constraints optimizeConstraints stops considering routing costs at all. Feasiblity rates per multistart iteration averaged only 33.8%; however, given the high number of total iterations, all runs found feasible solutions.

F. Experiment 5: different variants regarding routing depots

In this experiment we consider two additional variants regarding routing depots. The first uses a global depot h_G for all districts, which we define in the instances as the optimal 1-center vertex $h_G = \mathrm{argmin}_{i \in V} \max_{j \in V} d_{ij}$. The second does not use any depots. The adaptations to our algorithm required to treat these variants are minor, and omitted here. For each variant we ran our algorithm on all instances with $\lambda = 0.5$. Table VI reports, for each instance set and n, the average p-median value (PM) and routing costs (R) of each variant as relative deviations (in %) from the values of the default configuration, which uses a local depot to each district.

We observe that the variant with no depots finds 1.71% smaller routes, on average, compared to local depots. This was expected, as this variant incurs no additional costs when $h_i \notin D_i$. The single depot variant had significantly higher routing costs: 19% on average, which comes from all district tours having to visit h_G, regardless of the distance. Concerning PM, we find that the variant without any depots was consistently better for most instance sizes. A possible explanation is that, because routes are shorter, R plays a lesser role in Z compared to PM, and thus Z is more biased towards compactness.

IV. CONCLUSIONS

We have proposed a general heuristic solver for districting problems with routing costs modeled by traveling salesman tours. It builds upon existing state-of-the-art methods, and is

TABLE VI
COMPARISON OF DIFFERENT VARIANTS REGARDING DISTRICT DEPOTS.

Inst.	n	no depots		global depot	
		PM [rd.%]	R [rd.%]	PM [rd.%]	R [rd.%]
SRC	60	-0.94	-1.72	-0.67	10.74
SRC	80	0.06	-3.65	0.01	13.49
SRC	100	-0.71	-1.30	-0.45	17.93
SRC	120	-0.37	-2.33	0.40	19.32
SRC	150	0.06	-2.17	-0.18	22.59
SRC	200	-0.62	-2.09	-0.53	29.64
RF	500	-0.83	-1.23	0.36	17.01
RS	1000	-1.66	-1.70	0.12	24.46
GRM	1000	-1.13	-0.07	0.43	12.00
GRM	2500	-2.41	-0.86	1.39	22.90
Avg.		-0.85	-1.71	0.09	19.01

able to handle routing costs both as budget constraints and an objective in instances of up to 2500 basic units. Because the traveling salesman problem is NP-complete, we use a layered strategy for determining which heuristic to use at different stages of the algorithm, and show experimentally that this helps maintain high-quality tours during optimization at low computational costs. Experiments suggest that geometrically compact districts, in particular ones with small p-median values are also related to short tours, and that optimizing compactness can play an important role in minimizing routing costs. The proposed solver can also handle different problem variants regarding routing depots, and provides good approximations for variants that require tours to stay within district boundaries.

REFERENCES

[1] S. J. D'Amico, S.-J. Wang, R. Batta, and C. M. Rump, "A simulated annealing approach to police district design," *Comput. Oper. Res.*, vol. 29, no. 6, pp. 667–684, 2002.

[2] B. Yang, K. Viswanathan, P. Lertworawanich, and S. Kumar, "Fire station districting using simulation: Case study in Centre Region, Pennsylvania," *J. Urban Plan. Devel.*, vol. 130, no. 3, pp. 117–124, 2004.

[3] M. Camacho-Collados, F. Liberatore, and J. M. Angulo, "A multi-criteria police districting problem for the efficient and effective design of patrol sector," *Eur. J. Oper. Res.*, vol. 246, no. 2, pp. 674–684, 2015.

[4] S. Hanafi, A. Freville, and P. Vaca, "Municipal solid waste collection: An effective data structure for solving the sectorization problem with local search methods," *INFOR*, vol. 37, no. 3, pp. 236–254, 1999.

[5] E. Benzarti, E. Sahin, and Y. Dallery, "Operations management applied to home care services: Analysis of the districting problem," *Decision Support Systems*, vol. 55, no. 2, pp. 587–598, 2013.

[6] D. Datta, J. Figueira, A. Gourtani, and A. Morton, "Optimal administrative geographies: an algorithmic approach," *Socio-Economic Planning Sciences*, vol. 47, no. 3, pp. 247–257, 2013.

[7] M. T. A. Steiner, D. Datta, P. J. S. Neto, C. T. Scarpin, and J. R. Figueira, "Multi-objective optimization in partitioning the healthcare system of Parana state in Brazil," *Omega*, vol. 52, pp. 53–64, 2015.

[8] R. Z. Rios-Mercado and H. J. Escalante, "GRASP with path relinking for commercial districting," *Exp. Syst. Appl.*, vol. 44, pp. 102–113, 2016.

[9] A. Gliesch, M. Ritt, and M. C. Moreira, "A multistart alternating tabu search for commercial districting," in *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 2018, pp. 158–173.

[10] F. Ricca, A. Scozzari, and B. Simeone, "Political districting: from classical models to recent approaches," *Ann. Oper. Res.*, vol. 204, no. 1, pp. 271–299, 2013.

[11] D. M. King, S. H. Jacobson, and E. C. Sewell, "The geo-graph in practice: creating united states congressional districts from census blocks," *Computational Optimization and Applications*, vol. 69, no. 1, pp. 25–49, 2018.

[12] A. Gliesch, M. Ritt, and M. C. Moreira, "A genetic algorithm for fair land allocation," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2017, pp. 793–800.

[13] A. Butsch, "Districting problems - new geometrically motivated approaches," PhD thesis, 2016.

[14] M. A. Salazar-Aguilar, R. Z. Ríos-Mercado, and M. Cabrera-Ríos, "New Models for Commercial Territory Design," *Networks and Spatial Economics*, vol. 11, no. 3, pp. 487–507, 2011.

[15] M. Thorup, "Worst-case update times for fully-dynamic all-pairs shortest paths," in *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, 2005, pp. 112–119.

[16] J. Kalcsics, *Location Science*. Springer, 2015, ch. Districting problems.

[17] C. Prodhon and C. Prins, "A survey of recent research on location-routing problems," *Eur. J. Oper. Res.*, vol. 238, 2014.

[18] M. Alberada-Sambola, *Location Science*. Springer, 2015, ch. Location-Routing and Location-Arc Routing.

[19] 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 *Mexican International Conference on Artificial Intelligence*. Springer, 2011, pp. 307–318.

[20] L. Muyldermans, D. Cattrysse, D. van Oudheusden, and T. Lotan, "Districting for salt spreading operations," *Eur. J. Oper. Res.*, vol. 139, pp. 521–532, 2002.

[21] M. Lin, K.-S. Chin, C. Fu, and K.-L. Tsui, "An effective greedy method for the meals-on-wheels service districting problem," *Comput. Ind. Eng.*, vol. 106, 2017.

[22] D. Haugland, S. C. Ho, and G. Laporte, "Designing delivery districts for the vehicle routing problem with stochastic demands," *Eur. J. Oper. Res.*, vol. 180, no. 3, 2007.

[23] L. H, L. G, and G. B, "Districting for routing with stochastic customers," *EURO J. Transp. Logist.*, vol. 1, no. 1, pp. 67–85, 2012.

[24] H. Lei, R. Wang, and G. Laporte, "Solving a multi-objective dynamic stochastic districting and routing problem with a co-evolutionary algorithm," *Comput. Oper. Res.*, vol. 67, pp. 12–24, 2016.

[25] J. Beardwood, J. H. Halton, and J. M. Hammersley, "The shortest path through many points," *Math. Proc. Camb. Phil. Soc.*, vol. 55, no. 4, pp. 299–327, 1959.

[26] A. Gliesch and M. Ritt, "A generic approach to districting with diameter or center-based objectives," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2019, pp. 249–257.

[27] M. G. Resende and C. C. Ribeiro, *Optimization by GRASP*. Springer, 2016.

[28] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[29] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, jan 1986.

[30] M. R. Garey and D. S. Johnson, *Computers and intractability*. freeman San Francisco, 1979, vol. 174.

[31] G. Gutin and A. P. Punnen, Eds., *The Traveling Salesman Problem and Its Variations*. Springer, 2002.

[32] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The traveling salesman problem: a computational study*. Princeton university press, 2006.

[33] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling salesman problems," *Oper. Res.*, vol. 21, pp. 498–516, 1973.

[34] "Concorde TSP solver," 2006. [Online]. Available: http://www.tsp.gatech.edu/concorde

[35] R. Diestel, *Graph theory*, 3rd ed. Springer, 2005.

[36] R. Z. Ríos-Mercado and E. Fernández, "A reactive GRASP for a commercial territory design problem with multiple balancing requirements," *Computers and Operations Research*, vol. 36, no. 3, pp. 755–776, 2009.

[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*, vol. 19, no. 5, pp. 885–895, 2011.

[38] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: A case study in local optimization," *Local search in combinatorial optimization*, vol. 1, no. 1, pp. 215–310, 1997.