# A Column Generation Heuristic for Districting the Price of a Financial Product

Pierre de Fréminville[1], Guy Desaulniers[2], Louis-Martin Rousseau[3], and Sylvain Perron[4]

[1]Artelys, Paris, France
[2]Polytechnique Montréal and GERAD, Department of Mathematics and Industrial Engineering, Montréal, Canada
[3]Polytechnique Montréal and CIRRELT, Department of Mathematics and Industrial Engineering, Montréal, Canada
[4]HEC Montréal and GERAD, Department of Management Sciences, Montréal, Canada

May 1, 2014

**Abstract**

This paper studies a districting problem which arises in the context of financial product pricing. The challenge lies in partitioning a set of small geographical regions into a set of larger territories. In each territory, the customers will share a common price. These territories need to be contiguous, contain enough customers and be as homogeneous as possible in terms of customer value. To address this problem we present a column generation-based heuristic where the subproblem generates contiguous territories taken into account a non linear objective function. Computational results indicate that the territories produced by this heuristic are about 35% more homogenous than those previously used in practice. The developed algorithm has been transferred to a financial firm and is now used to help craft more competitive financial products.

*Keywords*: Districting, financial product pricing, clustering, column generation, heuristic.

## 1 Introduction

The districting problem consists of partitioning a geographical zone into a set of contiguous territories while optimizing a partitioning criterion. In the literature, this problem is also called the districting problem (Bozkaya et al. 2003), the contiguity constrained clustering problem (Hansen et al. 2003), the $p$-regions problem (Duque et al. 2011), or the zone design problem (Bacao et al. 2005). A well-studied application of it is political districting (Ricca and Simeone 2008). Several other applications are stated in the surveys of Duque et al. (2007) and Ricca et al. (2011) on the districting methods developed in the last 40 years. The districting problem is generally difficult to solve mainly because of the contiguity constraint. Variants of the problem have been proven NP-complete.

In this paper, we focus on a variant that has not been studied yet and that arises in companies selling a financial product whose cost price is customer-dependent but unknown a priori for each individual customer **because it is related to random events**. According to the regulations of a governmental authority, the product selling price cannot be set per customer: it must be the same for a relatively large subset of customers. It may, however, vary from one subset to another. The customer subsets must respect certain feasibility rules. First, each subset must contain a sufficiently large number of customers. Second, all customers residing in the same area (a geographical unit or GU that can be defined, e.g., according to the zip codes) must belong to the same subset. Third, each subset is composed of a certain number of GUs that form a contiguous territory. Furthermore, subregion constraints restrict the number of territories that can be used in densely populated area, such as large cities. The goal of the problem is to determine territories such that the expected cost prices of the customers it contains are relatively the same. This allows the company to set the selling price for a territory that seems advantageous to all customers in this territory and, thus, to retain a large market share. There are several ways to measure the cost homogeneity inside a territory. In this paper, the chosen criterion to be minimized is a weighted sum of each territory variance, that is, the *within-territory* variance. **Furthermore, in the present case study (and many other similar contexts), the financial company is already operating and selling products in the studied geographical zone. In such case, it is possible to observe individual cost prices incurred during the past financial years and to estimate the future cost price of a GU as the average of the past cost prices over all the customers it contains.** We call this problem the financial product districting problem (FPDP) and we describe it in details in Section 3.1.

The main contribution of this paper is to present an original column generation heuristic for solving the FPDP. In this heuristic, the column generation subproblem that generates potential territories as needed has a quadratic objective function and is solved by a greedy multi-start heuristic. This methodology allows to solve large-scale instances involving maps with more than 500 GUs. It was recently implemented in a software and tested by a company to design its future districting plans. We report the results of computational experiments performed on some of their data sets.

The paper is organized as follows. In the next section, we review the main approaches developed for solving districting problems. In Section 3 we give a detailed description of the FPDP and model it as a set partitioning problem with side constraints. In Section 4 we present the proposed algorithmic framework. The results of our computational experiments are reported in Section 5. Finally, conclusions are drawn in Section 6.

## 2 Literature review

Various applications of the districting problem have been studied including: political districting (Ricca and Simeone 2008), police zone districting (D'Amico et al. 2002), districting for salt spreading operations (Muyldermans et al. 2002), school districting (Ferland and Guénette 1990), max split clustering (Hansen et al. 2003), health-related analysis (Wise et al. 1997), homecare zone design (Blais et al. 2003), transportation area design (Tavares-Pereira et al. 2007), electric zone design (Bergey et al. 2003ab), and census re-engineering (Openshaw and Rao 1995). These applications involve different objective functions and constraints. For this reason, different heuristics were designed according to the variant of the districting problem studied. In this section, we present a summary of the districting methods that are closely related to the problem or the solution method presented in this paper. For a more detailed survey, we refer the reader to the recent works of Duque et al. (2007) and Ricca et al. (2011).

The most studied districting problem in the literature, the political districting problem (PDP), seeks to redraw political district boundaries. It aims at dividing an electoral map into equally populated and compact districts to prevent favoring a particular political party, thus preventing gerrymandering. To achieve good district design, other constraints can be added to the problem, such as socio-demographic homogeneity and similiarity to the existing plan.

One of the features of the FPDP that sets it apart from many districting problems is that there is no restriction on the shape of the territories. Furthermore, no equity constraint between the territories is imposed but rather a minimum weight constraint on each territory. Last but not least, the subregion constraints add complexity to the problem. These considerations led us to discard some of the methods for the PDP that optimize the compactness of the territories around a territory center (Hess et al. 1965).

In addition, the quadratic objective function of the FPDP, the within-territory variance, is not frequent in the districting literature (except in Wise et al. 1997; Guo 2008) but is widely applied and studied in classical clustering literature: see, e.g., the k-means algorithm of MacQueen (1967), the j-means algorithm of Hansen and Mladenovich (2001), and the exact sum of square deviations algorithm of Aloise (2009).

Below we review the main heuristic clustering methods proposed in the scientific literature. These methods can be grouped into the following classes: conventional clustering, adapted clustering, seeded territories, and local search. Afterwards we survey the few exact methods developed for the districting problem.

Conventional clustering methods are carried out by first using a conventional clustering algorithm and second rearranging the computed territories to make them contiguous. For example, the two-step algorithm proposed by Openshaw and Wymer (1995) involves aggregating the GUs with respect to their expected price cost before dividing the obtained territories by connected components and reassembling them into the required number of territories. The method introduces homogeneity in the first step. However, the first-step solution is often not a good starting point when contiguity is enforced in the second step.

Adapted clustering methods are primarily based on a hierarchical partitioning algorithm where only contiguous clusters of GUs can be merged. The characteristics of these methods are detailed in Margules et al. (1985). The principle is to agglomerate GUs iteratively until a predefined number of territories is reached. This approach is interesting when looking for nested solutions at different scales (that is, with different numbers of territories). However, if a particular scale is sought, it is not the best option. Indeed, the main problem with hierarchical clustering is that the distance between two objects can be higher than the distance between the union of these objects and a third one. Recently, this method was adapted by Guo (2008) and improved later on by Guo and Wang (2011).

The seeded territories method was proposed for the first time by Vickrey (1961) to solve the PDP, and was recently used by Duque et al. (2012). In this method, each territory starts from a "seed" or reference GU and neighboring units are iteratively added to each territory. The performance of the algorithm highly depends on the procedure applied to select the seeds.

Local search methods iteratively modify an initial solution while taking into account an aggregation criterion. At each iteration, the current solution must satisfy the contiguity constraint. The most frequently employed moves in the literature include moving a GU from a donor territory to a neighboring territory (Openshaw and Rao 1995; Ricca and Simeone 2008; Bozkaya et al. 2003), and swapping units between two neighboring territories (Bozkaya et al. 2003). Other proposed moves include: merging two territories and splitting them into two new regions (Openshaw 1978; Horn 1995), selecting a unit to create a new territory (Horn 1995), and combining two feasible solutions using genetic algorithm operators. Using these moves, different local search methods can be applied. For example, Ricca and Simeone (2008) compared four different local search methods and Bozkaya et al. (2003) developed a tabu search method. The main challenge with local search methods is to find an efficient way to determine which GU to move without disconnecting the donor territory. This problem belongs to the general class of subgraph connectivity (or dynamic subgraph) problems (see Duan 2010 2011). Several solution methods have been proposed in the districting literature to tackle it. In the first proposed methods (Nagel 1965; Openshaw 1977), contiguity of a territory is checked by starting from a GU and adding successively all neighboring GUs. Later, Macmillan (2001) introduced an alternative method called "switching points". Lately, Ricca and Simeone (2008) formulated two conditions to ensure that contiguity is preserved when moving a GU from one territory to another. Firstly, the GU to be moved must be adjacent to a GU in the destination territory. Secondly, this GU must not be a cut-node of the origin territory.

Exact solution methods have also been proposed for the districting problem. However they are either computationally expensive and can only solve 50-GU maps (Duque 2004; Duque et al. 2011), or they consider a linear objective function that leads to simplifications in the solution process (Hansen et al. 2003). On the one hand, Duque (2004) developed a computationally expensive edge selection procedure coupled with an exponential number of subtour breaking constraints. Later, Duque et al. (2011) enhanced this procedure with three new ways for enforcing the spatial contiguity constraint. On the other hand, Hansen et al. (2003) proposed to model the problem as a binary linear program with an exponential number of constraints depending on the number of nodes (GUs) and an exact row generator algorithm to solve 600-node instances for a specific graph partitioning problem subject to a contiguity constraint. This model relies, however, on a linear objective function, called the "split", defined as the smallest difference between a class unit and a unit outside of this class.

Another way to solve the problem exactly is to use an optimization model that takes into account all possible feasible territories as an input (Garfinkel and Nemhauser 1970). These enumeration models are rather rare in the districting literature. Mehrotra et al. (1998) adopted such a model and proposed a column generation method to solve PDP instances with 50-GU maps. Feasible territories are generated by solving a linear subproblem, where the cost of a territory represents its compactness. Every district generated within the process satisfies the contiguity constraint and bounds on its population. To obtain an integer solution, they used an exact branching rule that requires model simplifications (e.g., a simplified contiguity constraint).

## 3 Problem statement and mathematical formulation

In this section, we first give a detailed description of the FPDP and then present a set partitioning formulation for the FPDP.

### 3.1 Problem statement

The FPDP can be defined as follows. Given a geographical map divided into a set $G$ of GUs, it consists of clustering the GUs into a maximum of $N$ contiguous territories that must contain at least $N_c$ customers each. The values of $N$ and $N_c$ are usually determined by a governmental authority. This authority may also define a set $S$ of subregions (e.g. cities) and impose that at most $M_s$ territories cover the subregion $s \in S$. These constraints aim at preventing the company from over-segmenting the most populous areas of the map. Note that the constraint on the minimum number of customers per territory must be satisfied only when the territories are determined, that is, there is no need to recompute new territories if the number of customers in a territory drops below $N_c$ later.

The objective of the FPDP is to build territories that are homogeneous as much as possible with regards to the cost price of the customers they contain. To do so, the chosen criterion to be minimized is a weighted sum of each territory variance, that is, the *within-territory* variance. In this sum, the weight of a territory corresponds to the number of customers it contains. Its variance is computed by assuming that each customer in a GU has the same cost price, namely, an estimate of the average cost price of these customers. Minimizing this criterion is equivalent to maximizing the *between-territory* variance, that is, the weighted sum of the square differences between each territory mean and the total mean (Apostol and Mnatsakanian 2003).

In this work, we make the following assumptions:

1. all GUs in $G$ are disjoint and, for each GU $g \in G$, the following data are given:

   $u_g$, its (current) number of customers;

   $p_g$, its expected cost price per customer; and

   $(x_g, y_g)$, the position of its centroid;

2. a territory is connected (contiguous) if every pair of GUs in this territory can be linked through a sequence of adjacent GUs belonging to the territory;
3. a territory is said to cover a subregion if it contains at least one GU of that subregion;
4. an initial solution (a set of territories $T_0$) is generally provided. In this case, a minimum number of territories from this solution, denoted $N_0$, can be required to be part of the solution to compute.

The expected cost price $p_g$ for a GU $g \in G$ is usually obtained based on historical data. Note that using expected cost prices instead of probability distributions of the cost prices is a simplification to the problem that is justified by the complexity of solving it.

Often, the initial solution $T_0$ might be composed of the territories currently in use that we want to reoptimize. In this case, the number of customers in the territories in $T_0$ might have evolved over time and some of these territories might not be feasible anymore with respect to the minimum number of customers in a territory. The proposed algorithm can handled such an initial solution.

## 3.2 Mathematical formulation

Before presenting the proposed mathematical formulation for the FPDP, suppose that the set $T$ of all feasible territories is known, where a territory is deemed feasible if it is contiguous and contains at least $N_c$ customers. Then, the cost price variance $v_t$ for each territory $t \in T$ can be computed as follows. Let $u_t = \sum_{g \in t} u_g$ be the total number of customers in $t$ (where

$g \in t$ means GU $g$ belongs to territory $t$), which also corresponds to its weight. The average cost price $\bar{p}_t$ of $t$ is

$$\bar{p}_t = \frac{\sum_{g \in t} u_g p_g}{u_t} \tag{1}$$

and its cost price variance $v_t$ is, therefore, given by

$$v_t = \frac{\sum_{g \in t} u_g (p_g - \bar{p}_t)^2}{u_t}. \tag{2}$$

For each territory $t \in T$, we define for each GU $g \in G$ a binary parameter $a_{gt}$ that is equal to 1 if territory $t$ contains $g$ and 0 otherwise, and for each subregion $s \in S$, a binary parameter $b_{st}$ that takes value 1 if territory $t$ covers subregion $s$ and 0 otherwise.

Associating with each territory $t \in T$ a binary decision variable $Y_t$ equal to 1 if territory $t$ is selected and 0 otherwise, we can formulate the FPDP as the following set partitioning model with side constraints:

$$\text{Minimize} \quad \sum_{t \in T} u_t v_t Y_t \tag{3}$$

$$\text{subject to:} \quad \sum_{t \in T} a_{gt} Y_t = 1, \qquad \forall\, g \in G \tag{4}$$

$$\sum_{t \in T} Y_t \leq N, \tag{5}$$

$$\sum_{t \in T_0} Y_t \geq N_0, \tag{6}$$

$$\sum_{t \in T} b_{st} Y_t \leq M_s, \qquad \forall\, s \in S \tag{7}$$

$$Y_t \in \{0, 1\}, \qquad \forall\, t \in T. \tag{8}$$

The objective function (3) aims at minimizing the within-territory variance (the weighted sum of the cost price variances of the chosen territories). Constraints (4) ensure that each GU is assigned to one territory. Constraint (5) imposes an upper bound on the number of territories used in the computed solution. Because using additional territories can only reduce the within-territory variance, this bound is always reached in an optimal solution. Constraint (6) enforces a minimum number of territories to keep from the initial solution. If no initial solution is provided or $N_0 = 0$, constraint (6) is omitted. Constraints (7) restrict the number of territories that can be used to cover each subregion.

Let $sq_t$ be the weighted sum of square deviations from the mean in territory $t$, also called the intrinsic second moment:

$$sq_t = \sum_{g \in t} u_g (p_g - \bar{p}_t)^2. \tag{9}$$

Because $sq_t = u_t v_t$ for all $t \in T$, one can remark that the objective function corresponds to minimizing the sum of the intrinsic second moments of the chosen territories.

Note that the set partitioning formulation (3)–(8) has the advantage of being linear (omitting the binary requirements on the $Y_t$ variables). This linearity is possible because we assume that the territories in set $T$ can be enumerated a priori and, consequently, that the values of $u_t$ and $v_t$ for each territory $t \in T$ can also be computed before solving the program. Under this assumption, model (3)–(8) can be solved directly by a mixed integer programming solver. In practice, the complete enumeration of set $T$ is only achievable for small instances because the size of $T$ grows exponentially with the size of $G$. To avoid a complete enumeration, we propose in the next section a column generation heuristic that can be applied to solve large-scale FPDP instances.

## 4 Solution method

This section describes the heuristic that we propose for solving the FPDP. This method is based on the powerful linear programming technique called column generation (Dantzig and Wolfe 1960; Gilmore and Gomory 1961; Desaulniers et al. 2005). Column generation is an iterative algorithm that allows to solve efficiently certain linear programs containing a huge number of variables without enumerating them all a priori. The variables are associated with a set of combinatorial objects (routes, schedules, cutting patterns, territories) that can be represented implicitely in an optimization model. For solving the FPDP, column generation replaces the a priori enumeration of all possible territories (the columns) by a sequence of iterations in which potentially improving territories are generated and added to the model as needed. Our method differs from that of Mehrotra et al. (1998) in several points. In their method, the territories have to respect a compactness constraint. This is why they chose a linear objective function and a simplified contiguity constraint that enable them to generate only compact territories. In our problem, the objective function is quadratic and the contiguity constraint is considered explicitly, as no compactness is imposed. The quadratic form of the objective function together with the contiguity constraint makes the column (territory) generation subproblem difficult to solve. For this reason, we propose to solve the subproblem using a heuristic instead of an exact algorithm (see Section 4.3). In addition, Mehrotra et al. (1998) use an exact branching scheme to obtain an integer solution. In our approach, we rely on a heuristic branching process that does not allow backtracking in the search tree (see Section 4.4).

In the following, we describe first the main scheme of the proposed heuristic before providing details on its components.

## 4.1 Main algorithm

The pseudo-code of the main algorithm is given in Algorithm 1. Starting from an empty set of feasible territories $T'$, the algorithm adds territories to $T'$ using two procedures. Given an initial solution, the procedure *Split&Merge* is called first in Step 2. It splits the territories from the initial solution in various ways and merges the resulting subterritories to create new territories. In Step 3, a column generation procedure is applied to generate new territories. This column generation process exploits the linear relaxation of model (3)–(8) restricted to the current set of territories $T'$ to identify new territories that have the potential to improve the current linear programming (LP) solution. If the column generation process stops with an integer LP solution, the overall algorithm terminates. Otherwise, model (3)–(8) restricted to the current set of territories $T'$ is solved in Step 8 using a commercial mixed integer programming (MIP) solver such as Gurobi. Because this step can be highly time-consuming, we impose a time limit of $pMaxTime$ seconds, where $pMaxTime$ is a parameter. The best integer solution found during this step, if any, is then displayed.

In the hope of finding a first integer solution or an improving one if one was previously found in Step 8, the solution process proceeds to a sequence of branching decisions (Step 13) interspersed by calls to the column generation procedure (Step 15). The branching decisions are imposed to force the respect of the integrality requirements. Column generation is invoked to add territories to $T'$ that can be seen as complementary to the imposed decisions. Note that, to avoid excessive computational times, the decisions are fixed permanently, that is, no backtracking is performed (the heuristic is then termed a diving heuristic). However, to keep some flexibility toward the end of the solution process, we stop imposing branching decisions when the current LP solution $Y^{LP}$ contains less than $pMinFrac$ fractional-valued variables, where $pMinFrac$ is a parameter of predetermined value. At that moment, if the current LP solution is integer, the heuristic stops. Otherwise, model (3)–(8) restricted to the current set of territories $T'$ is solved in Step 19 using a MIP solver (without any time limit). The best integer solution found, if any, is then outputted. Our computational experiments show that this solution is often better than the one produced in Step 8.

Steps 8 to 21 can thus be seen as the application of two different heuristics to derive a good integer solution. The first (Step 8) exploits the power of a commercial MIP solver on a restricted set of potential territories. The second (Steps 12 to 21) is a diving heuristic that generates new territories after imposing each branching decision and can, therefore, yield a better integer solution.

The procedure *Split&Merge*, the column generation process and the types of branching decisions imposed are discussed in Subsections 4.2, 4.3 and 4.4, respectively.

**Algorithm 1** Main algorithm

```
1: $T' := \emptyset$
2: Call procedure *Split&Merge* to create new territories and add them to $T'$
3: Apply column generation to add new territories to $T'$
4: Store the current LP solution, denoted $Y^{LP}$ hereafter
5: **if** $Y^{LP}$ is integer **then**
6:     Output this solution and its cost
7: **else**
8:     Solve integer model (3)–(8) restricted to $T'$ (subject to time limit $pMaxTime$)
9:     **if** a feasible integer solution is found **then**
10:        Output this solution and its cost
11:    Restore solution $Y^{LP}$
12:    **while** the current solution contains at least $pMinFrac$ fractional-valued variables **do**
13:        Impose a branching decision
14:        Remove from $T'$ the territories that are in conflict with this decision
15:        Apply column generation to add new territories to $T'$
16:    **if** the current solution is integer **then**
17:        Output this solution and its cost
18:    **else**
19:        Solve integer model (3)–(8) restricted to $T'$
20:        **if** a feasible integer solution is found **then**
21:            Output this solution and its cost
```

## 4.2 Procedure *Split&Merge*

Column generation heuristics perform better when initialized with a set of "good" columns. To generate an initial set of territories, we propose a procedure called *Split&Merge* that splits multiple times each territory of an initial solution (i.e., the territories in $T_0$) into two subterritories, and merges these subterritories to create new territories containing two or three subterritories. The subterritories themselves are also considered as new territories. If no initial solution is provided, an initial solution can be generated using the seeded territory method described in Section 2 and used by Mehrotra et al. (1998) to initialize their column generation algorithm.

A pseudo-code of the procedure *Split&Merge* is given in Algorithm 2, where $P$ is the set of the pairs of subterritories obtained by splitting in two the territories in $T_0$, whereas $L_1$, $L_2$ and $L_3$ are sets containing the new territories composed of one, two and three subterritories, respectively. In Steps 4 to 8, each initial territory $t \in T_0$ (feasible or not) is split into various pairs of subterritories. Each split is done by considering a line cutting the territory into two subterritories and then associate each GU in $t$ with the subterritory that contains its centroid. A line is defined by a point $(\bar{x}, \bar{y}) \in \mathbb{R}^2$ and an angle $\theta$:

$$\{(x, y) \in \mathbb{R}^2 \mid y = \bar{y} + \tan\theta\,(x - \bar{x})\}.$$

**Algorithm 2** Procedure *Split&Merge*

```
1: $L_1 = L_2 = L_3 := \emptyset$
2: for all $t \in T_0$ do
3:    $P := \emptyset$
4:    for $i = 0$ to $pNoSpc$ do
5:       for $j = 0$ to $pNoSpc$ do
6:          for $k = 1$ to $pNoAng$ do
7:             Create the pair of subterritories $(t_1, t_2)$ obtained from the line defined by the
               triplet $(x_{ti}, y_{tj}, \theta_{tk})$
8:             $P := P \cup \{(t_1, t_2)\}$
9: if $|P| > pMaxNoSub$ then
10:   Sort the pairs $(t_1, t_2)$ in $P$ in increasing order of their score $w_{t_1,t_2}$
11:   Keep the first $pMaxNoSub$ pairs and delete the others from $P$
12: for all $(t_1, t_2) \in P$ do
13:   $L_1 := L_1 \cup \{t_1, t_2\}$
14: for all $t_1 \in L_1$ do
15:   for all $t_2 \in L_1$ such that $t_1 \neq t_2$ do
16:      Create a new territory $t := t_1 \cup t_2$
17:      $L_2 := L_2 \cup \{t\}$
18: for all $t_1 \in L_1$ do
19:   for all $t_2 \in L_2$ such that $t_1 \not\subseteq t_2$ do
20:      Create a new territory $t := t_1 \cup t_2$
21:      $L_3 := L_3 \cup \{t\}$
22: Check the feasibility of all territories in $L_1 \cup L_2 \cup L_3$ and remove all infeasible ones
23: Return $L_1 \cup L_2 \cup L_3$
```

According to such a line, the subterritories $t_1$ and $t_2$ are given by:

$$t_1 = \{g \in t \mid y_g \leq \bar{y} + \tan\theta\,(x_g - \bar{x})\}$$

and

$$t_2 = \{g \in t \mid y_g > \bar{y} + \tan\theta\,(x_g - \bar{x})\}.$$

Figure 1 illustrates the split of a territory $t$ in two subterritories (one in light grey and the other in dark grey) according to a line defined by the indices $i = 1$, $j = 2$ and an angle $\theta$ close to $-\frac{\pi}{4}$. Note that the resulting subterritories $t_1$ and $t_2$ are retained even if they are not feasible territories (due to contiguity or to the number of customers they contain). Indeed, merging them with other subterritories may yield feasible territories.

The set of lines used to generate the pairs of subterritories from a territory $t$ is determined through a discretization of the $xy$-plane and the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$ for the angle $\theta$. This discretization relies on the two parameters, namely, *pNoSpc* the number of subdivisions of the $x$-axis (or the $y$-axis) over the region covered by a territory and *pNoAng* the number of subdivisions of $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

Let $x_t^{min} = \min_{g \in t} x_g$, $y_t^{min} = \min_{g \in t} y_g$, $x_t^{max} = \max_{g \in t} x_g$, and $y_t^{max} = \max_{g \in t} y_g$ be the

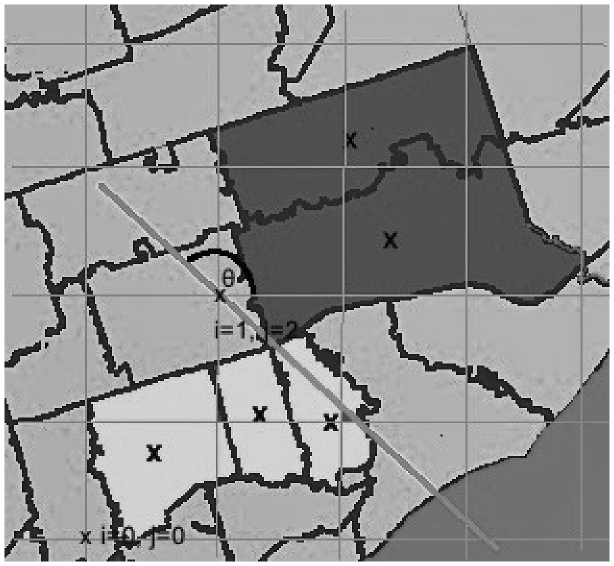

Figure 1: Splitting a territory in two subterritories

minimum and maximum $x$- and $y$-coordinates of the centroids of the GUs in territory $t$. For all $i, j \in \{0, 1, \ldots, pNoSpc\}$ and $k \in \{1, 2, \ldots, pNoAng\}$, a line is defined for the triplet

$$(x_{ti}, y_{tj}, \theta_{tk}) = \left( x_t^{min} + \frac{i}{pNoSpc}(x_t^{max} - x_t^{min}), y_t^{min} + \frac{j}{pNoSpc}(y_t^{max} - y_t^{min}), -\frac{\pi}{2} + \frac{k}{pNoAng}\pi \right).$$

Notice that different lines can yield the same pair of subterritories.

The pairs of subterritories resulting from the splitting process are kept in set $P$. At the end of the splitting process, if $|P|$ exceeds a predetermined value given by parameter $pMaxNoSub$, the elements $(t_1, t_2)$ in $P$ are sorted in Step 10 in increasing order of their score $w_{t_1,t_2} = u_{t_1}v_{t_1} + u_{t_2}v_{t_2}$ (the more homogeneous are the subterritories, the lower the score of the pair). Then, only the first $pMaxNoSub$ elements in $P$ are kept, the others are discarded in Step 11.

Once the best pairs of subterritories are selected, the procedure constructs territories with one, two or three subterritories in Steps 12–13, Steps 14–17, and Steps 18–21, respectively. Finally, in Step 22, all created territories in lists $L_1$, $L_2$ and $L_3$ are checked for feasibility (minimum number of customers and contiguity); infeasible ones are rejected.

Note that when a line is used to split a territory, it can yield more than two connected subterritories (for example, when the territory has a U-shape). In this case, the set $P$ not only contains pairs of subterritories but also subsets of more than two subterritories. These subsets are treated similarly to the pairs.

## 4.3 Column generation

**Algorithm 3** Column generation procedure

```
1: repeat
2:     Solve the RMP considering only the current subset of territories $T'$
3:     Using the dual solution of the RMP, define the objective function of the subproblem
       (see formula (10) below)
4:     Solve the subproblem in the hope of finding negative reduced cost territories
5:     if negative reduced cost territories are found then
6:         Add these territories to $T'$
7: until No territories with a negative reduced is found
```

When applied to the linear relaxation of model (3)–(8) that is then called the master problem, column generation proceeds as follows (see Algorithm 3). Starting from an initial set $T'$ of territories, the algorithm solves at each iteration a restricted master problem (RMP) in Step 2 and a column generation subproblem in Step 4. The RMP corresponds to the master problem restricted to the current set of territories $T'$. It is solved by an LP solver such as the simplex algorithm to provide a primal solution and a dual solution. The subproblem aims at finding feasible territories $t \in T \setminus T'$ that can be added to set $T'$ to possibly improve the solution of the current RMP, that is, territories whose corresponding variables $Y_t$ have a negative reduced cost with respect to the current RMP dual solution (in a more concise way, we will say negative reduced cost territories). This dual solution is, therefore, used in Step 3 to define the subproblem objective function. When such territories are found, they are added to $T'$ in Step 6 before launching a new iteration. Otherwise, the column generation process stops. When the subproblem is solved to optimality, that is, when negative reduced cost territories are identified if some exist, the optimal solution of the last RMP is also optimal for the master problem and provides a lower bound on the optimal value of the integer model.

In the main algorithm of our heuristic for solving the FPDP (see Algorithm 1), column generation is invoked twice, namely, in Steps 3 and 15. In Step 3, column generation starts with the set $T'$ obtained from the procedure *Split&Merge*. In Step 15, it begins with the current set $T'$. Note that in both cases, the initial set of columns does not guarantee the feasibility of the first RMP. Indeed, as mentioned in Section 3.1, the initial solution $T_0$ might not be feasible with respect to the minimum number of customers in a territory and, in this case, the territories provided by the procedure *Split&Merge* might not be sufficient to yield a feasible RMP. Furthermore, in Step 14, territories that do not respect the imposed branching decisions are removed from $T'$ which might also yield an infeasible RMP. To ensure feasibility, we add to every constraint (4) and (6) an artificial variable that has a huge cost. A solution is deemed feasible only if all artificial variables take value 0.

Let $\pi_g$, $g \in G$, $\sigma$, $\beta$, and $\alpha_s$, $s \in S$, be the dual variables associated with the constraints (4)–(7), respectively. Given this notation, the reduced cost $\bar{c}_t$ of variable $Y_t$, $t \in T$, is:

$$\bar{c}_t = u_t v_t - \sigma - \sum_{g \in G} a_{gt} \pi_g - \sum_{s \in S} b_{st} \alpha_s. \tag{10}$$

The subproblem consists of finding territories $t \in T$ with $\bar{c}_t < 0$. Equivalently, it can be stated as finding a territory with the least reduced cost and expressed as:

$$\min_{t \in T} \bar{c}_t.$$

The "best" mathematical programming formulation that we could develop for this subproblem (see de la Poix de Fréminville 2012) is an integer program that involves a quadratic function and an exponential number of constraints (with respect to the number of GUs) to impose contiguity. Because the subproblem seems very hard to solve exactly, we propose a multi-start greedy heuristic for solving it, i.e., for finding negative reduced cost territories.

The greedy component of the heuristic has some similarities with the seeded territories method described in Section 2. It starts from an initial territory (possibly infeasible with respect to the number of customers it contains) and increases it in a greedy fashion by selecting at each iteration a GU to add to it based on a best insertion criterion. More precisely, let $t$ be the current territory at a given iteration. For each GU $g$ that is adjacent to $t$, the reduced cost $\bar{c}_{t \cup \{g\}}$ of the potential territory $t \cup \{g\}$ is computed using formula (10). Among the evaluated territories, the one yielding the lowest reduced cost is selected to pursue the search and is referred hereafter to a *visited* territory. Note that all potential territories must be connected but do not have to satisfy the constraint on the minimum number of customers. All potential territories satisfying this constraint and having a negative reduced cost are added to the current set of territories $T$.

The algorithm is called multi-start algorithm because it starts from various initial territories and even from the same territory several times. Two different sets of initial territories are considered, namely, the set of territories containing each a single GU and the set of territories associated with each basic variable in the current basis of the RMP. The latter set contains territories that have a zero reduced cost; therefore, they are good starting points to search for negative reduced cost territories. They also allow the generation of territories containing a relatively large number of GUs. At the opposite, the former set offers the possibility to create territories that contains a small number of GUs and that can be seen as complementary to the large ones.

The algorithm uses a list of already visited territories to avoid visiting the same territory several times (hence, it is possible to start from the same initial territory several times). In fact, to ensure fast computational times, this list is rather approximated by a set of keys associated with the visited territories. In our case, the key corresponds to the total number of customers in a territory. All keys associated with the visited territories are registered in a set $K$. Any territory, other than an initial territory, whose key belongs to set $K$ cannot be visited anymore. Note that set $K$ is emptied at the beginning of each column generation iteration.

The multi-start greedy algorithm is controlled by the following four parameters:

*pNoTrialsGU*: number of times that each GU is used as an initial territory;

*pNoGreedyItGU*: maximum number of iterations (territory increases) performed by the greedy algorithm for each initial territory corresponding to a GU;

*pNoTrialsBsc*: number of times that each territory associated with a basic RMP variable is used as an initial territory;

*pNoGreedyItBsc*: maximum number of iterations performed by the greedy algorithm for each territory associated with a basic variable.

A pseudo-code describing it is given in Algorithm 4. Steps 2 to 8 apply the greedy procedure *pNoTrialsGU* times on the set of initial territories composed of a single GU, whereas Steps 9 to 15 apply it *pNoTrialsBsc* times on the set $B$ of initial territories associated with the basic RMP variables.

In Steps 6 and 13 of Algorithm 4, the procedure *IncreaseTerritory* is called to find the GU to add to the current territory $t$. A pseudo-code for this procedure is presented in Algorithm 5. In Step 2, $A_t$ indicates the set of GUs that are adjacent to territory $t$. The loop in Steps 2–9 finds the cheapest GU that can be added to the current territory $t$. In this loop, *tmp* denotes a potential territory whose reduced cost is computed in Step 5 if it has not been visited previously according to set $K$ (Step 4). The value of $v_{tmp}$ is computed from $v_t$ as explained in the following paragraph. Given the subregions covered by territory $t$, the contribution of the dual variables to the reduced cost (10) of *tmp* can easily be computed from $t$. In Step 6, the feasibility of *tmp* with regards to the number of customers is checked. If *tmp* is feasible and has a negative reduced cost, it is added to the overall current set $T'$ of territories (those in the RMP or already generated in this column generation iteration) in Step 7. If needed, the best insertion is updated in Steps 8 and 9. The procedure returns the best potential territory, if any, in Steps 10 to 13.

We recall that the average cost price $\bar{p}_t$ of a territory $t \in T$ and its cost price variance $v_t$ are given by formulae (1) and (2). Given the average cost price $\bar{p}_t$ of territory $t$, the average cost price and cost price variance of a territory $t \cup \{g\}$ for $g \in A_t$ can be computed as:

$$\bar{p}_{t\cup\{g\}} = \frac{1}{u_t + u_g}(u_t\bar{p}_t + p_g) \tag{11}$$

$$v_{t\cup\{g\}} = \frac{1}{u_t + u_g}\sum_{h\in(t\cup\{g\})} u_h(p_h - \bar{p}_{t\cup\{g\}})^2. \tag{12}$$

To improve computational efficiency, we rather used the following formulae that were proven in Apostol and Mnatsakanian (2003):

$$\bar{p}_{t_1\cup\{g\}} = \frac{1}{(u_{t_1} + u_g)}\big(u_{t_1}\bar{p}_{t_1} + u_g\bar{p}_g\big) \tag{13}$$

$$v_{t_1\cup\{g\}} = \frac{1}{(u_{t_1} + u_g)}\big(u_{t_1}v_{t_1} + \frac{u_{t_1}u_g}{u_{t_1} + u_g}|\bar{p}_{t_1} - \bar{p}_g|^2\big). \tag{14}$$

These formulae enable computing the reduced cost of a potential territory in Step 5 of Algorithm 5 with a constant time complexity, given that we store the average cost price and the cost price variance of the current territory $t$.

**Algorithm 4** Procedure *MultiStartGreedy*

```
1: $K := \emptyset$
2: for $i := 1 \to pNoTrialsGU$ do
3:     for all $g \in G$ do
4:         $t := \{g\}$, $j := 1$
5:         while $j \leq pNoGreedyItGU$ and $t \neq NULL$ do
6:             $t := IncreaseTerritory(K, t)$
7:             if $t \neq NULL$ then
8:                 $K := K \cup \{t\}$
9: for $i := 1 \to pNoTrialsBsc$ do
10:    for all $t \in B$ do
11:        $j := 1$
12:        while $j \leq pNoGreedyItBsc$ and $t \neq NULL$ do
13:            $t := IncreaseTerritory(K, t)$
14:            if $t \neq NULL$ then
15:                $K := K \cup \{t\}$
```

**Algorithm 5** Procedure $IncreaseTerritory(K, t)$

```
1: $\bar{c}_{min} := \infty$, $g_{min} := NULL$
2: for all $g \in A_t$ do
3:     $tmp := t \cup \{g\}$
4:     if $tmp \notin K$ then
5:         Compute $\bar{c}_{tmp}$
6:         if $\bar{c}_{tmp} < 0$ and $tmp$ is feasible then
7:             $T' := T' \cup \{tmp\}$
8:         if $\bar{c}_{tmp} < \bar{c}_{min}$ then
9:             $\bar{c}_{min} := \bar{c}_{tmp}$, $g_{min} := g$
10: if $g_{min} \neq NULL$ then
11:     Return $t \cup \{g_{min}\}$
12: else
13:     Return $NULL$
```

Finally, note that the proposed multi-start greedy heuristic constructs new territories only by adding GUs to visited territories. It does not allow to remove a GU from a visited territory because this operation can yield a non-contiguous territory and checking contiguity of the resulting territory is rather time-consuming.

## 4.4 Branching decisions

In Step 13 of Algorithm 1, branching decisions are imposed permanently to reduce the total computational time and to guide the solution process toward a good-quality integer solution. Two types of decisions can be imposed that we call *column fixing* and *GU pair fixing*.

**Column fixing**: The variable $Y_t$ with the highest fractional value in the current RMP solution is fixed at one. Let $\tilde{t}$ be the territory associated with this decision. Then all territories $t \in T'$ such that $t \neq \tilde{t}$ and $t \cap \tilde{t} \neq \emptyset$ are removed from $T'$ in Step 14 of Algorithm 1 and their corresponding variables from the RMP.

**GU pair fixing**: For each pair of adjacent GUs $(g_1, g_2) \in G^2$ (with $g_1 \neq g_2$), the following value is computed: $f_{g_1,g_2} = \sum_{t \in T'_{g_1,g_2}} \hat{Y}_t$, where $\hat{Y}_t$ is the value of $Y_t$ in the current RMP solution and $T'_{g_1,g_2} \subseteq T'$ is the subset of territories containing both GUs $g_1$ and $g_2$. The pair $(g_1, g_2)$ with the highest fractional value $f_{g_1,g_2}$ is selected and the decision imposes that these two GUs be covered by the same territory. All territories in $T'$ covering either $g_1$ or $g_2$ but not both are then removed from $T'$ in Step 14 and their corresponding variables from the RMP. Furthermore, the multi-start greedy algorithm is modified to ensure that all new territories to be generated cover either both GUs $g_1$ and $g_2$, or none of them. To do so, an aggregated GU representing $g_1$ and $g_2$ is created. In the greedy algorithm, $g_1$ or $g_2$ cannot be added individually to a territory (nor can they serve as initial territories). Only the aggregated GU can be added (and can serve as an initial territory). Note that further decisions can yield an aggregated GU that represents more than two GUs.

These two types of branching decisions are combined in the following way. If there exists a variable $Y_t$ taking a fractional value greater than or equal to a prespecified threshold $pMinThCol$, column fixing is applied. Otherwise, GU pair fixing is used. Priority is thus given to column fixing unless only a doubtful decision can be imposed. In this case, a less aggressive GU pair fixing decision is made.

# 5 Computational experiments

In this section, we present the computational results obtained by the proposed algorithm on real-life instances. All computational experiments were conducted on a personal computer operating under Linux, equipped with an Intel Core i7 CPU 960 clocked at 3.2GHz and 24GB of RAM. The algorithm was implemented in C++ and relied on the Gurobi commercial MIP solver, version 5.1, for solving the integer programs in Steps 8 and 19 of Algorithm 1.

In the following, instead of reporting the value of the solutions according to the objective function (3), that is, their within-territory variance $v_{within}$, we rather report the *within-cluster correlation coefficient* $r_{within}$ that is defined as the proportion of the total variance due to the within-territory variance:

$$r_{within} = \frac{v_{within}}{v_{tot}}$$

where $v_{tot} = \frac{\sum_{g \in G} u_g (p_g - \bar{p})^2}{\sum_{g \in G} u_g}$ and $\bar{p} = \frac{\sum_{g \in G} u_g p_g}{\sum_{g \in G} u_g}$. Because $v_{tot}$ is a constant, the two measures $v_{within}$ and $r_{within}$ are equivalent from an optimization point of view. However, the coefficient $r_{within}$ ($\in [0, 1]$) allows to better evaluate and compare different solutions for the same instance or for different instances. The lower the value, the higher the impact of the districting proposed by a solution.

## 5.1 Test instances

For our computational experiments, we use two instances of the FPDP that were defined on a single 522-GU map containing a 101-GU subregion. This map, provided by our industrial partner (who wishes to remain anonymous), must be partitioned into a maximum of 55 territories such that at most 10 of them can cover the subregion. Our partner also supplied two real-life datasets (one for each instance) that were collected in two different years of operations. Each dataset indicates the number of customers in each GU and its expected cost price per customer. All input and output information were handled and visualized using the MapInfo Geographical Information System. MapInfo was also used to extract additional geographical information such as the adjacency of GUs. For each instance, we also had access to the territories used by our partner that served as an initial solution for our heuristic.

## 5.2 Results and sensitivity analysis

We performed several test runs of the proposed heuristic to determine empirically a good parameter setting. We opted for the values reported in Table 1.

| | | |
|---|---|---|
| $pMaxTime = 7200s$ | $pNoTrialsGU = 2$ | $pNoGreedyItGU = 30$ |
| $pNoSpc = 20$ | $pNoTrialsBsc = 3$ | $pNoGreedyItBsc = 10$ |
| $pNoAng = 20$ | $pMinFrac = 10$ | $pMinThCol = 0.7$ |

Table 1: Selected parameter values

Parameter $pMaxTime$ was set to a sufficiently large value to allow, most of the times, the computation of the best integer solution in Step 8 of Algorithm 1. Parameters $pNoSpc$ and $pNoAng$ were also set to large values that ensure generating the maximum number of subterritory pairs in the procedure *Split&Merge*. The results of a sensitivity analysis on the value of the other parameters are given below.

The results of our computational experiments are reported in Tables 2 and 3 for the first and the second instance, respectively (assuming that no initial territories must be kept, that is, $N_0 = 0$). In these tables, each line provides the results of a test performed with a specific parameter configuration. This parameter configuration is described in the first six columns (the first four parameters concern the multi-start greedy Algorithm 4, the last two the branching process). In fact, the first line indicates a reference configuration and, for clarity purposes, each subsequent line only identifies the parameter value that differs from the reference configuration. For each parameter configuration, we report nine statistics in the remaining nine columns. The first set of two columns reports the value of $r_{within}$ for the solution computed in Step 8 of Algorithm 1 and for the best solution found overall, respectively. The second set of five columns reports the time spent: (i) in the root node (procedure *Split&Merge* and column generation), (ii) for solving the integer program in Step 8, (iii) for performing the branching (including the column generation reoptimizations), (iv) for solving the reduced integer program in Step 19 of Algorithm 1, and (v) for performing the overall algorithm. The last two columns indicate the number of column fixing decisions and of GU pair fixing decisions applied, respectively. All reported times are in seconds. A dash in the column $r_{within}$ MIP Step 8 means that no integer solution was found within the 7200-second time limit. Furthermore, the best value in each of the $r_{within}$ columns is highlighted in bold. Finally, averages (excluding the entries with a dash) are reported in the last line of each table.

Looking at Tables 2 and 3, one can notice the robustness of the proposed heuristic since the parameters setting changes do not affect too much the solution quality. Both integer programs in Steps 8 and 18 of Algorithm 1 produce solutions with similar $r_{within}$ values, but we observe that in two cases for the first instance, no feasible solutions were found in Step 8 within the 7200-second time limit. Consequently, the branching phase combined with column generation allows to slightly improve solution quality but also brings reliability in the solution process. In terms of computational effort, the difference yielded by varied parameter configurations is much more significant, but in all cases compatible with a planning process that is conducted about once a year. We observe that, for the first instance, solving the integer programs in Step 8 requires the largest proportion of the total computational time on average. This is not the case for the second instance where the branching phase is, on average, the most time consuming. These results show that the time devoted to Step 8 may vary substantially from one instance to another (even of the same size) and that imposing a time limit in Step 8 allows to avoid excessive computational times. Note that instances involving much more than 500 GUs are common in practice. In fact, most of the 512 GUs of the map that we used for our tests correspond to clusters of smaller-sized GUs that were determined by our industrial partner in a preprocessing step to ease their manual solution process.

The solutions provided by our partner exhibit $r_{within}$ values of 22.9 for the first instance and

50.7 for the second one. The proposed heuristic is thus able to reduce the key performance index by about 35% and has been implemented by our partner to help support decisions during future districting exercises.

It is difficult to provide optimality gaps for the computed solutions because no tight lower bounds are available. In fact, to compute lower bounds, one would need to either rely on a nonlinear model (either for a direct solution or as a column generation subproblem) or to enumerate all potential territories and solve the linear relaxation of model (3)–(8). The former option is out of the scope of this paper and the latter is doable only for very small instances. Consequently, to provide insight into the quality of the solutions produced by our heuristic, we performed the following experiment. Using Algorithm 1, we solved the first instance described above but restricted to only the 101 GUs of its subregion that must be partitioned into 10 territories. Then, for each pair of adjacent territories in the computed solution, we checked (by explicit enumeration) if these two territories provide an optimal partition for the GUs contained in these territories. The results indicate that an optimal partition is obtained for 15 of the 16 adjacent pairs. For the other pair, an optimality gap of 5.7% was observed. Consequently, the proposed algorithm seems sufficiently effective to provide solutions that cannot be easily improved by a simple enumeration.

| parameters | | | | | | results | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| column generation | | | | branching | | $r_{within}$ | | CPU time (s) | | | | | No. fixed | |
| $pNoTrialsGU$ | $pNoGreedyItGU$ | $pNoTrialsBsc$ | $pNoGreedyItBsc$ | $pMinFrac$ | $pMinThCol$ | MIP Step 8 | Best | Root | MIP Step 8 | Branching | MIP Step 18 | Total | Columns | GU pairs |
| 2 | 30 | 3 | 10 | 10 | 0.9 | 15.63 | 15.44 | 301 | 1069 | 1014 | 291 | 2675 | 2 | 16 |
| 1 | | | | | | 15.56 | 15.56 | 183 | 1568 | 1116 | 161 | 3028 | 1 | 26 |
| 3 | | | | | | 15.54 | 15.27 | 357 | 2866 | 1813 | 423 | 5459 | 5 | 27 |
| 4 | | | | | | 15.55 | **15.24** | 431 | 1536 | 1840 | 254 | 4061 | 5 | 26 |
| | 20 | | | | | 15.58 | 15.54 | 145 | 448 | 1126 | 130 | 1849 | 8 | 33 |
| | 40 | | | | | 15.62 | 15.52 | 552 | 1846 | 2081 | 493 | 4972 | 1 | 23 |
| | 50 | | | | | 15.55 | 15.55 | 1011 | 1246 | 1942 | 528 | 4727 | 3 | 15 |
| | | 1 | | | | - | 15.41 | 222 | 7200 | 831 | 536 | 8789 | 2 | 12 |
| | | 2 | | | | 15.61 | 15.61 | 246 | 449 | 907 | 199 | 1801 | 3 | 20 |
| | | 4 | | | | 15.56 | 15.55 | 341 | 547 | 2209 | 266 | 3363 | 6 | 33 |
| | | | 20 | | | 15.58 | 15.49 | 441 | 5764 | 1368 | 456 | 8029 | 1 | 17 |
| | | | 30 | | | - | 15.34 | 754 | 7200 | 2244 | 1748 | 11946 | 6 | 12 |
| | | | 40 | | | **15.41** | 15.41 | 1575 | 4828 | 5206 | 825 | 12434 | 1 | 27 |
| | | | | 20 | | 15.63 | 15.44 | 255 | 1069 | 1027 | 268 | 2619 | 2 | 16 |
| | | | | 30 | | 15.63 | 15.44 | 270 | 1108 | 1016 | 261 | 2655 | 2 | 16 |
| | | | | 40 | | 15.63 | 15.44 | 234 | 1092 | 1022 | 264 | 2607 | 2 | 16 |
| | | | | | 0.6 | 15.63 | 15.63 | 187 | 865 | 670 | 36 | 1758 | 17 | 9 |
| | | | | | 0.7 | 15.63 | 15.46 | 312 | 1102 | 979 | 270 | 2663 | 12 | 14 |
| | | | | | 0.8 | 15.63 | 15.49 | 253 | 1111 | 1779 | 155 | 3298 | 9 | 38 |
| | | | | | **Average** | 15.59 | 15.46 | 424.8 | 2258.6 | 1589.0 | 398.1 | 4670.5 | 4.6 | 21.4 |

Table 2: Results for the first instance

| parameters | | | | | | results | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| column generation | | | | branching | | $r_{within}$ | | CPU time (s) | | | | | No. fixed | |
| $pNoTrialsGU$ | $pNoGreedyItGU$ | $pNoTrialsBsc$ | $pNoGreedyItBsc$ | $pMinFrac$ | $pMinThCol$ | MIP Step 8 | Best | Root | MIP Step 8 | Branching | MIP Step 18 | Total | Columns | GU pairs |
| 2 | 30 | 3 | 10 | 10 | 0.9 | 32.65 | 32.55 | 617 | 263 | 914 | 165 | 1959 | 9 | 10 |
| 1 | | | | | | 32.74 | 32.50 | 300 | 173 | 1591 | 100 | 2164 | 4 | 41 |
| 3 | | | | | | 32.83 | 32.51 | 548 | 299 | 1825 | 154 | 2826 | 6 | 26 |
| 4 | | | | | | 32.83 | **32.48** | 608 | 705 | 2708 | 212 | 4233 | 8 | 30 |
| | 20 | | | | | 32.64 | 32.50 | 202 | 122 | 363 | 193 | 880 | 5 | 6 |
| | 40 | | | | | 32.70 | 32.49 | 819 | 366 | 2940 | 252 | 4377 | 7 | 32 |
| | 50 | | | | | 32.82 | **32.48** | 1347 | 1212 | 2800 | 386 | 5745 | 5 | 21 |
| | | 1 | | | | 33.03 | 32.57 | 390 | 272 | 1141 | 187 | 1990 | 7 | 23 |
| | | 2 | | | | 32.78 | 32.55 | 406 | 228 | 2658 | 36 | 3328 | 14 | 50 |
| | | 4 | | | | **32.61** | 32.55 | 456 | 251 | 1773 | 192 | 2672 | 4 | 30 |
| | | | 20 | | | 32.74 | 32.53 | 746 | 414 | 1694 | 320 | 3174 | 2 | 21 |
| | | | 30 | | | 32.73 | 32.56 | 1028 | 633 | 4184 | 365 | 6210 | 6 | 30 |
| | | | 40 | | | 32.72 | 32.56 | 2178 | 2752 | 5595 | 381 | 10906 | 11 | 31 |
| | | | | 20 | | 32.65 | 32.55 | 595 | 258 | 968 | 177 | 1998 | 9 | 10 |
| | | | | 30 | | 32.65 | 32.55 | 598 | 252 | 966 | 196 | 2012 | 9 | 10 |
| | | | | 40 | | 32.65 | 32.55 | 576 | 257 | 919 | 173 | 1925 | 9 | 10 |
| | | | | | 0.6 | 32.65 | 32.59 | 680 | 292 | 741 | 155 | 1868 | 19 | 1 |
| | | | | | 0.7 | 32.65 | 32.52 | 637 | 269 | 475 | 239 | 1620 | 8 | 2 |
| | | | | | 0.8 | 32.65 | 32.51 | 550 | 261 | 693 | 156 | 1660 | 7 | 5 |
| | | | | | **Average** | 32.72 | 32.53 | 699.0 | 488.3 | 1839.3 | 212.5 | 3239.1 | 7.8 | 20.5 |

Table 3: Results for the second instance

## 5.3 Preserving some initial territories

In some cases, a company may want to limit the number of initial territories it has to modify. In fact, re-designing territories often implies additional fixed costs for each modified territory. For this reason, we introduced constraint (6) in the mathematical formulation (3)–(8) of the FPDP. With this functionality, the company can evaluate the gain brought by a modification of an initial territory compared to the cost of modifying its average pricing. Initial territories to be preserved are not selected in advance, but selected during the optimization process. To test this functionality, we conducted a series of experiments on FPDP instances derived from the first instance used in our previous tests. All these instances considered only the 101-GU subregion of the whole map that must be partitioned into a maximum of 10 territories. The instances only vary by the number of initial territories $N_0$ that must be kept. Note that, in the solution provided by our industrial partner for the first instance, these GUs are covered by exactly 10 territories that do not cover GUs outside the subregion. For these tests, the proposed heuristic used the reference parameter configuration given in Section 5.2

Table 4 reports the results obtained from these experiments where the value of $N_0$ varies between 0 and 9 (the columns have the same meaning as in Tables 2 and 3). The case $N_0 = 0$ corresponds to preserving no initial territories (if not advantageous). The case $N_0 = 9$ is equivalent to the case $N_0 = 10$ where all initial territories must be kept. In this case, no optimization is required and the solution has an $r_{within}$ value of 48.66. First, we observe that the solution computed with $N_0 = 0$ yields a 45% reduction of the $r_{within}$ value compared to the initial solution. As expected the $r_{within}$ value of the best solution found increases regularly with the value of $N_0$ except for the case $N_0 = 1$ for which a better solution than with $N_0 = 0$ was computed. This exception is due to the heuristic nature of our solution method. Like for the first instance treated in Section 5.2, most of the time is devoted to solving the integer program in Step 8 of Algorithm 1. Notice that even for a 101-GU instance, the 7200-second time limit was reached for the case $N_0 = 3$. This is the only case where branching was able to improve upon the solution computed in Step 8.

# 6 Conclusion

The FPDP is a complex industrial problem that challenges financial corporations. Although it shares common features with other districting problems, its unique definition prevents one from using an existing methodology to address it. In this paper we introduced a column generation-based heuristic that can compute solutions which are significantly better (by about 35%) than those currently use in practice. Furthermore, the proposed methodology allows to control the level of changes from a current set of territories, for cases where perturbing all territories is not acceptable. The developed methodology has been transferred to a financial corporation and is now used to craft the territories and set the financial product cost prices.

| | $r_{within}$ | | CPU time (s) | | | | | No. fixed | |
|---|---|---|---|---|---|---|---|---|---|
| $N_0$ | MIP Step 8 | Best | Root | MIP Step 8 | Branching | MIP Step 18 | Total | Columns | GU pairs |
| 0 | 26.83 | 26.83 | 55 | 4229 | 78 | 8 | 4370 | 2 | 9 |
| 1 | 26.66 | 26.66 | 68 | 755 | 8 | 170 | 1001 | 0 | 0 |
| 2 | 27.49 | 27.49 | 64 | 459 | 56 | 36 | 615 | 1 | 4 |
| 3 | 34.44 | 28.23 | 84 | 7200 | 116 | 1 | 7401 | 2 | 27 |
| 4 | 29.49 | 29.49 | 70 | 1475 | 76 | 50 | 1671 | 0 | 5 |
| 5 | 32.76 | 32.76 | 104 | 2247 | 193 | 1 | 2545 | 1 | 26 |
| 6 | 35.86 | 35.86 | 117 | 682 | 277 | 3 | 1079 | 0 | 27 |
| 7 | 41.93 | 41.93 | 133 | 630 | 276 | 0 | 1039 | 2 | 37 |
| 8 | 43.87 | 43.87 | 103 | 78 | 284 | 4 | 469 | 3 | 18 |
| 9 | - | 48.66 | - | - | - | - | - | - | - |

Table 4: Results for a 101-GU instance with a varying number of initial territories preserved

## References

D. Aloise. *Exact Minimum Sum of Square Clustering.* PhD thesis, École Polytechnique de Montréal, 2009.

T. M. Apostol and M. A. Mnatsakanian. Sums of squares of distances in m-space. *American Mathematical Monthly*, 110(2):516–526, 2003.

F. Bacao, V. Lobo, and M. Painho. Applying genetic algorithms to zone design. *Soft Computing*, 9:341–348, 2005.

P.K. Bergey, C.T. Ragsdale, and M. Hoskote. A decision support system for the electrical power districting problem. *Decision Support Systems*, 36:1–17, 2003a.

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

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

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

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

G.B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

P. de la Poix de Fréminville. *Partitionnement d'une zone géographique en territoires homogènes et contigus.* PhD thesis, École Polytechnique de Montréal, 2012.

G. Desaulniers, J. Desrosiers, and M. M. Solomon. *Column Generation.* Springer-Verlag, New York, 2005.

R. Duan. New data structures for subgraph connectivity. *ICALP 2010: 37th International Colloquium on Automata, Languages and Programming*, pages 201–212, 2010.

R. Duan. *Algorithms and Dynamic Data Structures for Basic Graph Optimization Problems.* PhD thesis, University of Michigan, 2011.

J. Duque, L. Anselin, and S. Rey. The Max-P-regions Problem. *Journal of Regional Science*, 52(3), 2012.

J.C. Duque. *Design of homogeneous territorial units. A methodological proposal and applications.* PhD thesis, University of Barcelona, 2004.

J.C. Duque, R. Ramos, and J. Surinach. Supervised regionalization methods: a survey. *International Regional Science Review*, 30(3):195–220, 2007.

J.C Duque, R.L. Church, and R. Middleton. The p-regions problem. *Geographical Analysis*, 43(3):104–106, 2011.

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

R.S. Garfinkel and G.L. Nemhauser. Optimal political districting by implicit enumeration techniques. *Management Science Series B-Application*, 16:495–508, 1970.

P.C. Gilmore and R.E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9:849–859, 1961.

D. Guo. Regionalization with dynamically constrained agglomerative clustering and partitioning (redcap). *International Journal of Geographical Information Science*, 22(7):801–823, 2008.

D. Guo and H. Wang. Automatic region building for spatial analysis. *Transactions in GIS*, 15(1):29–45, 2011.

P. Hansen and N. Mladenovich. J-means: a new local search heuristic for minimum sum of squares clustering. *Pattern Recognition*, 34(2):405–413, 2001.

P. Hansen, B. Jaumard, C. Meyer, B. Simeone, and V. Doring. Maximum split clustering under connectivity constraints. *Journal of Classification*, 20:143–180, 2003.

S.W. Hess, J.B. Weaver, H.J. Siegfeld, J.N. Whelan, and P.A. Zitlau. Nonpartisan political redistricting by computer. *Operations Research*, 13:998–1006, 1965.

M.E. Horn. Solution techniques for large regional partitioning problems. *Geographical Analysis*, 27:230–48, 1995.

W. Macmillan. Redistricting in a GIS environment: An optimization algorithm using switching points. *Journal of Geographical Systems*, 3:167–80, 2001.

J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1:281–97, 1967.

C. Margules, D. Faith, and L. Belbin. An adjacency constraint in agglomerative hierarchical classifications of geographic data. *Environment and Planning*, 17:397–412, 1985.

A. Mehrotra, E. Johnson, and G. L. Nemhauser. An optimization based heuristic for political districting. *Management Science*, 44:1100–14, 1998.

L. Muyldermans, D. Cattrysse, V. Oudheusden, and T. Lotan. Districting for salt spreading operations. *European Journal of Operational Research*, 139:521–532, 2002.

S. Nagel. Simplified bipartisan computer redistricting. *Stanford Law Review*, 17:863–899, 1965.

S. Openshaw. A geographical solution to scale and aggregation problems in region-building, partitioning and spatial modeling. *Transactions of the Institute of British Geographers*, 2: 459–72, 1977.

S. Openshaw. An optimal zoning approach to the study of spatially aggregated data. In I. Masser and P.J.B. Brown, editor, *Spatial Representation and Spatial Interaction*, pages 95–113. M. Nijhoff Social Sciences Division, 1978.

S. Openshaw and L. Rao. Algorithms for reengineering 1991 census geography. *Environment and Planning*, 27:425–446, 1995.

S. Openshaw and C. Wymer. Classifying and regionalizing census data. In S. Openshaw, editor, *Census Users Handbook*, pages 239–270. GeoInformation International, 1995.

F. Ricca and B. Simeone. Local search algorithms for political districting. *European Journal of Operations Research*, 189(3):1409–1426, 2008.

F. Ricca, A. Scozzari, and B. Simeone. Political districting: from classical models to recent approaches. *4OR: A Quarterly Journal of Operations Research*, 9:223–254, 2011.

F. Tavares-Pereira, J. Figueira, V. Mousseau, and B. Roy. Multiple criteria districting problems. *Annals of Operations Research*, 154:69–92, 2007.

W. Vickrey. On the prevention of gerrymandering. *Political Science Quarterly*, 76:105–110, 1961.

S.M. Wise, R. P. Haining, and J. Ma. Regionalisation tools for exploratory spatial analysis of health data. In M.M. Fischer and A. Getis, editors, *Recent Developments in Spatial Analysis: Spatial Statistics, Behavioural Modelling, and Computational Intelligence*, pages 83–100. Springer, 1997.