UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ALEX GLIESCH

A genetic algorithm for fair land allocation

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. Marcus Ritt

Porto Alegre
February 2018

CIP — CATALOGING-IN-PUBLICATION

Gliesch, Alex

A genetic algorithm for fair land allocation / Alex Gliesch. – Porto Alegre: PPGC da UFRGS, 2018.

46 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2018. Advisor: Marcus Ritt.

1. Land allocation. 2. Genetic algorithm. 3. Combinatorial optimization. 4. Districting. 5. Agrarian reform. I. Ritt, Marcus. II. Título.

“We should not judge people by their peak of excellence, but by the distance they have traveled from the point where they started.”

— HENRY WARD BEECHER

ABSTRACT

The goal of agrarian reform projects is the redistribution of farmland from large latifundia to smaller, often family farmers. One of the main problems the Brazilian National Institute of Colonization and Agrarian Reform (INCRA) has to solve is to subdivide a large parcel of land into smaller lots that are balanced with respect to certain attributes. This problem is difficult since it considers several constraints originating from legislation as well as ethical considerations. Current solutions are computer-assisted, but manual, time-consuming and error-prone, leading to rectangular lots of similar areas which are unfair with respect to soil aptitude and access to hydric resources. In this thesis, we propose a genetic algorithm to produce fair land subdivisions automatically. We present a greedy randomized constructive heuristic based on location-allocation to generate initial solutions, as well as mutation and recombination operators that consider specifics of the problem. Experiments on 5 real-world and 25 artificial instances confirm the effectiveness of the different components of our method, and show that it leads to fairer solutions than those currently applied in practice.

Keywords: Land allocation. Genetic algorithm. Combinatorial optimization. Districting. Agrarian reform.

Um algoritmo genético para a alocação justa de terras.

RESUMO

O objetivo de projetos de reforma agrária é redistribuir terras de grandes latifúndios para terrenos menores, com destino à agricultura familiar. Um dos principais problemas do Instituto Nacional de Colonização e Reforma Agrária (INCRA) é subdividir uma parcela grande de terra em lotes menores que são balanceados com relação a certos atributos. Este problema é difícil por que precisa considerar diversas restrições legais e éticas. As soluções atuais são auxiliadas por computador, mas manuais, demoradas e suscetíveis a erros, tipicamente produzindo lotes retangulares de áreas similares mas que são injustos com relação a critérios como aptidão do solo ou acesso a recursos hidrográficos. Nesta dissertação, nós propomos um algoritmo genético para gerar subdivisões justas de forma automática. Nós apresentamos um algoritmo construtivo guloso randomizado baseado em locação-alocação para gerar soluções iniciais, assim como operadores de mutação e recombinação que consideram especificidades do problema. Experimentos com 5 instâncias reais e 25 instâncias geradas artificialmente confirmam a efetividade dos diferentes componentes do método proposto, e mostram que ele gera soluções mais balanceadas que as atualmente usadas na prática.

Palavras-chave: Alocação de terras. Algoritmo genético. Otimização combinatória. Distritamento. Reforma agrária.

LIST OF ABBREVIATIONS AND ACRONYMS

GA	Genetic algorithm
BFS	Breadth-first search
SA	Simulated annealing
TS	Tabu search
GRASP	Greedy randomized adaptive search procedure
PR	Path relinking
MIP	Mixed integer programming
CPT	Chinese postman tour

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

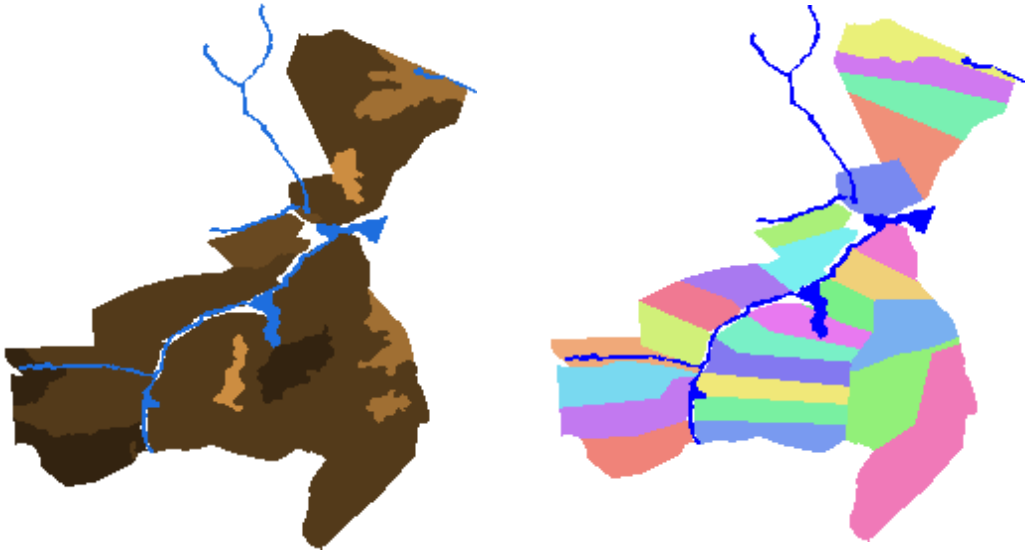
1.1 Motivation

The Territorial Organization in Agrarian Reform Projects and Environmental Planning Problem (PROTERRA, acronym for the name in Portuguese: *Problema de Organização Territorial em Projetos de Reforma Agrária e Planejamento Ambiental*) consists of subdividing a large parcel of agrarian land with a single owner into several smaller lots which are to be designated to family farmers, under a set of constraints. PROTERRA is directly associated with the agrarian reform process, which is one of the main factors in the sustainable development of countries. Successful implementations of agricultural settlements have shown to have direct impacts on land use, food production, job creation and eradication of hunger (CLEMENTS, 2012; MOURA et al., 2011; REIS, 2012). In Brazil, the technical regulations for the design of agrarian settlements are controlled by the Brazilian National Institute of Colonization and Agrarian Reform (INCRA). However, the current land allocation process is mainly manual, time-consuming and often flawed, usually leading to rectangular lots of similar areas but which fail to consider intrinsic attributes such as soil aptitude or access to hydric resources, and have difficulties handling regions with irregularly-shaped boundaries.

Problem instances are given in the form of ESRI shapefiles (ESRI; PAPER, 1998) describing a rectangular area of farmland where a given number of lots must be allocated. The topology of an instance is defined by land areas representing farmable soil and which are allowed to be allocated to a lot, water areas representing hydric resources such as rivers or lakes, and natural preservation areas representing regions of environmental importance and which must not be part of the allocation, so as to avoid undue interventions that may affect the feasibility of settlement. Lots are required to be contiguous regions of farmable land: water and natural reserves act as obstacles and may not divide a lot. For simplicity, other man-made entities such as existing architecture or road network are represented in the instances as natural reserves, since they also serve the purpose of obstacles.

Besides its size and topology, an instance is characterized by its soil: the usable land is divided into regions with different classes of aptitude with respect to their potential to grow crops. According to Ferreira (2015), these aptitude values are obtained by an analysis of physical, agronomic and social factors. The number of classes of aptitude is fixed between 5 and 10 per instance. Regions with the same aptitude are contiguous and

Figure 1.1: Left: the topology of the real-world instance Veredas. White areas represent outer boundaries or nature preservation areas, blue represents water areas, and the different tints of brown represent the different soil aptitudes, darker tints being higher aptitudes. Right: the current manual subdivision for this instance. Each color corresponds to a lot.



have smooth transitions between them. Typically areas nearest to water resources have an easier access to irrigation, and thus have the highest soil aptitude. Figure 1.1 shows the real-world instance Veredas, located in the Brazilian state of Minas Gerais, as well as its current manual allocation used by INCRA. One can observe that the manual allocation is somewhat unfair to a few lots, seemingly not considering soil aptitude at all, and does not always respect river boundaries.

In addition to lot contiguity, a fair allocation must also consider geographical, political and fairness constraints. Current requirements are that lots be accessible (i.e. no enclaves) and balanced fairly with respect to area, average soil aptitude and access to water. Other variant criteria, which are not considered in this thesis, include satisfying a minimum lot area while maximizing the number of lots allocated, balancing soil capacity and access to road networks, or generating lots that meet some geometric compactness measure.

Our main motive in this thesis is to propose efficient, automatic methods for generating land subdivisions that abide to the specified requirements. To this end, we represent land descriptions as discrete 2-dimensional grids, which allows us to model the problem as a classical districting problem on a planar graph. Districting problems have been widely studied in the literature and are usually NP-hard, and thus typically solved by heuristic approaches. Section 1.2 provides the necessary background on these problems, as well as relevant applications in land allocation. In Section 1.3 we present a mathematical defini-

tion for PROTERRA which is based on the current requirements by INCRA. In Chapter 2 we propose a genetic algorithm that finds suitable heuristic solutions to PROTERRA, and present experimental evidence to confirm its effectiveness. We conclude in Chapter 3.

1.2 Background

1.2.1 Land allocation

Fair land allocation has been a topic of research in the recent years in both developed and developing countries. Typically, real-world implementations of land allocation have specific sets of requirements, usually stemming from national legislation, and thus problem representations and solution methods tend to be quite diverse. In the following, we overview some recent studies that are relevant to our own.

Borgwardt, Brieden and Gritzmann (2014) propose a computational system based on geometric clustering algorithms to solve a farmland consolidation problem in the state of Bavaria, Germany. The study considers heterogeneous parcels whose shapes and sizes are given by the instance. The current holdings of each farmer are usually disconnected and dispersed over the region, which results in increased overheads in transportation and management, and the goal is to reassign the parcels in a way that each farmer receives a connected, compact set of parcels whose area and total soil aptitude are similar to their current holdings.

Bui, Pham and Deville (2013) develop a constraint-based local search algorithm to solve an agricultural land allocation problem in Vietnam. The main motivation is similar to the one of Borgwardt, Brieden and Gritzmann (2014): current allocations are scattered over the region, which increases overhead. The problem considers several fields of different soil aptitudes, and each farmer must receive a land parcel of predefined area in each field. Although they can have arbitrary shapes, the paper only considers rectangular fields and parcels.

Demetriou, See and Stillwell (2013b) propose a genetic algorithm based on geometric modeling to solve a land partitioning problem Cyprus. A weighted objective function considers four optimization criteria: balancing lots with respect to area and soil aptitude, ensuring access to a road network and compactness. For the latter, a specialized index proposed by Demetriou, See and Stillwell (2013a) is used. Although the problem is reasonably similar to our own, the approach they propose relies on a purely geometric

representation of solutions, rather than a discrete one. The method generates initial solutions by Voronoi diagrams, and the crossover and mutation operators act by adding or removing centroids and recomputing these diagrams.

In Brazil, computational methods for land subdivision in agrarian reform projects have previously been studied by Moreira et al. (2011) and Neto et al. (2011), where a tabu search procedure and a genetic algorithm are proposed, respectively. These studies target a simplified version of PROTERRA which only considers the balancing of soil aptitudes and areas by a weighted objective function, and are a previous work to our own. As our approach, both use a grid discretization of the map to treat the map as a planar graph. Recently, part of the results on PROTERRA presented in this thesis have been published at the Genetic and Evolutionary Computation Conference (GLIESCH; RITT; MOREIRA, 2017).

1.2.2 Districting problems

Because we use a discrete model and represent instances as weighted undirected planar graphs, we can model PROTERRA as a classical districting problem. A districting (or territory design) problem aims at grouping several small geographical areas, usually called basic units, into larger areas, called districts (or territories). Typically units represent real-world entities, such as a city block, street, or, in our case, a parcel of land. Districting problems appear in a wide range of applications, including the design of electoral districts (BRIEDEN; GRITZMANN; KLEMM, 2017; KING; JACOBSON; SEWELL, 2017; RICCA; SCOZZARI; SIMEONE, 2013; RICCA; SIMEONE, 2008; BOZKAYA et al., 2011; BAÇÃO; LOBO; PAINHO, 2005; HESS et al., 1965), sales territories (RÍOS-MERCADO; FERNÁNDEZ, 2009; MORENO-REGIDOR; García López de Lacalle; MANSO-CALLEJO, 2012; LEI et al., 2015; RÍOS-MERCADO; ESCALANTE, 2016; SALAZAR-AGUILAR; RÍOS-MERCADO; CABRERA-RÍOS, 2011), police districts (CAMACHO-COLLADOS; LIBERATORE; ANGULO, 2015), health care districts (STEINER et al., 2015; BLAIS; LAPIERRE; LAPORTE, 2003), and, in our case, agrarian lots (GLIESCH; RITT; MOREIRA, 2017).

Although the range of applications is diverse, most districting problems share three core optimization criteria: districts are required to be equally balanced with respect to attributes associated to basic units (e.g. total population, expected travel cost, soil aptitude), be connected with respect to an underlying planar graph over the units, and be geometri-

cally compact. The former concept is typically hard to define, as shape compactness is a subjective measure. Other domain-specific requirements include maximizing similarity to a previous plan, conforming to administrative boundaries, ensuring district accessibility (i.e., enclaves), and enforcing a minimum or maximum district cardinality.

Districting problems are usually NP-hard (KALCSICS; NICKEL; SCHRÖDER, 2005) and have been solved primarily by metaheuristic approaches, such as genetic algorithms (GLIESCH; RITT; MOREIRA, 2017; FORMAN; YUE, 2003; BAÇÃO; LOBO; PAINHO, 2005), GRASP (RÍOS-MERCADO; ESCALANTE, 2016; SALAZAR-AGUILAR; RÍOS-MERCADO; GONZÁLEZ-VELARDE, 2013; GUO; WANG, 2011), tabu search (BUI; PHAM; DEVILLE, 2013; MOREIRA et al., 2011; BUTSCH; KALCSICS; LAPORTE, 2014; BOZKAYA et al., 2011), and local search (RICCA; SIMEONE, 2008). A popular optimization model is location-allocation (KALCSICS; NICKEL; SCHRÖDER, 2005), which consists of greedily selecting initial seed nodes, one for each district (location), and then growing (or allocating) these districts out of the remaining unassigned nodes. Other, less common approaches are geometrically-inspired (KALCSICS; NICKEL; SCHRÖDER, 2009; RICCA; SCOZZARI; SIMEONE, 2008; BRIEDEN; GRITZMANN; KLEMM, 2017), or exact, through mixed-integer programming (SALAZAR-AGUILAR; RÍOS-MERCADO; CABRERA-RÍOS, 2011; BENZARTI; SAHIN; DALLERY, 2013; GARCÍA-AYALA et al., 2016).

Table 1.1 gives an overview of relevant studies on a number of different districting problems, the constraints and objective functions they consider, and the solution methods proposed. One can see that the columns representing compactness, balance and connectivity are a constant through most domains. For further details on these (and more) applications and respective solution techniques, we refer the reader to the extensive overviews done by Kalcsics, Nickel and Schröder (2005) and Kalcsics (2015).

Table 1.1: Overview of some relevant studies of districting problems. Columns Comp., Bal., Size, Conf., Conn, and Enc. denote whether the corresponding study considers compactness, balance, size, conformity, connectivity and accessibility constraints, respectively. GA, TS, SA, PR and OBA correspond to genetic algorithm, tabu search, simulated annealing, path relinking and old bachelor acceptance, respectively.

Source	Application	Obj. function	Cmp.	Bal.	Size	Conf.	Conn.	Acc.	Method
Kalcsics, Nickel and Schröder (2009)	generic		✓	✓			✓		geometric
Duque, Anselin and Rey (2012)	generic	(k, homogeneity)	maybe		✓		✓	✓	TS, SA
Li, Church and Goodchild (2014)	generic	compactness	✓	✓	✓	✓	✓		TS, GRASP
Forman and Yue (2003)	political	bal.+cmp., weighted	✓	✓			✓		GA
Bozkaya et al. (2011)	political	5-way weighted	✓	✓		✓	✓		TS
Bação, Lobo and Painho (2005)	political	balance + comp.	✓	✓			✓		GA
Ricca and Simeone (2008)	political	bal., cmp., conf.	✓	✓		✓	✓		TS, SA, OBA
Brieden, Gritzmann and Klemm (2017)	political	compactness	✓	✓		✓	✓		geometric
Butsch, Kalcsics and Laporte (2014)	routing	district CPT	✓	✓			✓		TS
Assis, Franca and Usberti (2014)	routing	bal., cmp.	✓	✓		✓	✓		GRASP
Ríos-Mercado and Escalante (2016)	commercial	compactness	✓	✓			✓		GRASP+PR
Salazar-Aguilar, Ríos-Mercado and González-Velarde (2013)	commercial	bal., cmp.	✓	✓			✓		GRASP
Salazar-Aguilar, Ríos-Mercado and Cabrera-Ríos (2011)	commercial	compactness	✓	✓			✓		MIP
Borgwardt, Brieden and Gritzmann (2014)	land alloc.			✓					geometric
Gliesch, Ritt and Moreira (2017)	land alloc.	balance		✓	✓		✓	✓	GA

1.3 Problem definition

We represent input areas as regular rectangular two-dimensional grids, whose cells are hereafter referred to as *spatial units*. The precision with which the shapefiles are discretized is determined by experts and may differ among instances, but usually corresponds to 300m^2 to 1000m^2 per cell. Based on this discretization, in this section we provide a mathematical description of the problem. We use the set notation $[n] = \{1, \ldots, n\}$.

Let k be the number of lots to allocate, and $U = R \dot\cup P \dot\cup L$ be the set of spatial units located on a regular $n \times m$ matrix, where R comprises the units representing rivers or water resources, P the units representing nature preservation areas, and L the units representing usable land areas. For each $u \in L$ we are given a land aptitude value q_u. For the instances originating from real-world scenarios we have used the aptitude class-value mapping proposed by Ferreira (2015), and for artificial instances these values were chosen uniformly from the range $[30, 100]$.

We define a *lot* as a subset of land units $C \subseteq L$, and a *solution* $S = (C_1, \ldots, C_k)$ as a k-partition of the usable land units into lots. A solution S can also be seen as a surjective mapping $S : L \to [k]$ that assigns a lot to each land unit, and so we use the notation $S(u)$ to represent the lot that unit $u \in L$ is assigned to. If a unit u is not currently allocated to any lot, $S(u)$ is undefined and we say that u is *unassigned*. If a solution has no unassigned units we say that it is *complete* (i.e., $\bigcup_{i \in [k]} C_i = L$), otherwise it is *incomplete* or *partial*.

We use a standard 4-neighborhood on a grid (top, bottom, left, right) to define the neighborhood $N(u)$ of a unit $u \in U$. This notion is extended to lots in the straightforward way $N(C) = (\cup_{u \in C} N(u)) \setminus C$. With N we define the land unit graph $G = (L, E)$ where $E = \{\{u, v\} \mid u \in L, v \in N(u) \cap L\}$. We say that a land unit $u \in L$ is *next to water* if $N(u) \cap R \neq \emptyset$, and that u is *on the border* of lot i to lot j if $u \in C_i$ and $N(u) \cap C_j \neq \emptyset$. Finally, we define $B_{ij} = \{u \mid u \text{ is on the border of } i \text{ to } j\}$, and the total border of lot i as $B_i = \bigcup_{j \in [k]} B_{ij}$.

A solution is said to be *valid* if it satisfies four constraints: *connectivity*, *accessibility*, *balance* and *equality*. The connectivity constraint requires lots to be connected in G. The accessibility constraint requires that there be no enclaves, i.e. lots are not allowed to completely enclose one another. The balance constraint states that lots with direct access to a water resource must not be larger in area than lots without direct access to water. Here, the area of a lot i is defined simply as the number $|C_i|$ of spatial units assigned to

it. Finally, the equality constraint requires that the ratio λ of the area of the largest lot and the area of the smallest lot be smaller than a given threshold $\bar{\lambda} = 3$.

The goal of PROTERRA is to find a valid solution that is as fair as possible by minimizing the standard deviation σ of lot aptitude values. The total aptitude value of a lot i is defined by $v_i = \sum_{u \in C_i} q_u$, and the standard deviation of lot qualities as

$$\sigma = \sqrt{1/k \sum_{i \in [k]} (v_i - \bar{v})^2},$$

where $\bar{v}$ is the average v_i. When comparing two solutions, we can omit the square root and the division by k, so in our algorithms we use the sum of squares

$$SS = \sum_{i \in [k]} (v_i - \bar{v})^2$$

as a surrogate objective function.

2 A GENETIC ALGORITHM FOR FAIR LAND ALLOCATION

2.1 Overview

We propose a genetic algorithm (GA) to find heuristic solutions to PROTERRA. Our algorithm iteratively evolves a population of solutions by generating offspring solutions from a set of individuals, which are selected from the current population based on their fitness. The offspring solutions are obtained through recombination and mutation operators inspired by the natural evolution process. Individuals in the initial population are generated by a randomized greedy heuristic, and each next generation is composed of some of the best individuals from the previous generation, some offspring solutions, and some newly-generated random solutions. For a thorough, detailed introduction to genetic algorithms we refer the reader to the excellent literature available (e.g. Gendreau and Potvin (2010), Goldberg (1989), Holland (1975)).

Algorithm 1 A genetic algorithm for PROTERRA

```
1:  P ← {greedyConstructiveHeuristic() | i ∈ [p]}
2:  b ← best solution in P
3:  repeat
4:      P' ← best e solutions in P
5:      for i ∈ [o] do
6:          (S1, S2) ← selectParents(P)
7:          S3 ← crossover(S1, S2)
8:          S3 ← mutate(S3)
9:          P' ← P' ∪ {S3}
10:     P ← P' ∪ {greedyConstructiveHeuristic() | i ∈ [r]}
11:     b ← best solution in P
12: until stopping criterion satisfied
13: return b
```

Algorithm 1 outlines our main method in detail. The initial population P is created by generating p solutions using a greedy constructive algorithm (line 1), which will be presented further. The main loop of the algorithm (lines 3–12) iteratively updates the current population. The new population P' consists of the best e solutions of the current population (line 4), o solutions obtained by the recombination process (lines 5–9), and r solutions generated using the same randomized greedy process used for the initial population. The elite rate e, the offspring rate o and the random rate r are parameters, and $e + o + r = p$. By adding r randomized solutions at each iteration, we ensure that the population stays varied and avoid that all individuals become related or very similar. The

parent solutions S_1 and S_2 of each new offspring solution S_3 are uniquely selected by a 3-tournament in P (line 6). A 3-tournament selects 3 elements uniformly at random from P and returns the best one. It is ensured that $S_1 \neq S_2$, but for simplicity we allow for the same pair of parents to be selected more than once. The crossover and mutation operators (lines 7 and 8) are stochastic, and will be detailed in a following section. The algorithm ends when a stopping criterion is reached, which may be a maximum time limit, a minimal objective function threshold, or a maximum number of generations. At each iteration we update the best overall solution b and return it at the end (lines 2, 11 and 13).

In the following sections, we present the components of the algorithm in detail.

2.2 Solution representation

We represent solutions in memory as integer vectors of size $|L|$ which map a land unit in $[L]$ to a lot in $[k]$ (or an appropriate flag if the unit is unassigned). This representation is straightforward given the definition of S as a surjective map of land units into lots. On the other hand, it can be inefficient for tasks which operate solely on the lot borders B_i, such as the construction of new solutions or local search procedures that swap units between neighboring lots. This is because the majority of the units in a lot are not on its border, but rather in the central part, and thus need not be considered.

A rough approximation based on the typical perimeter-area ratio of convex polygons is that a neighborhood considering only the lot borders has size $O(\sqrt{|L|})$. This can be a significant improvement over one that considers all $O(|L|)$ land units. To this end, we use a data structure that allows us to access, modify and iterate over lot borders in constant time. For each lot i, we keep a doubly-linked circular list $b_i = (b_{i1}, \ldots, b_{i|B_i|}, b_{i1})$ of units on its border. Note that this list is not necessarily ordered by any criterion, and adjacency in b does not imply adjacency in L. Additionally, we keep a mapping M that maps a unit $u \in L$ to the corresponding node in $b_{S(u)}$, or is undefined if $u \notin B_i$. This allows us to perform the following operations in constant time:

- $test(i, u)$: to test if unit u is in B_i, we check if $S(u) = i$ and $M(u)$ is defined.
- $insert(i, u)$: when unit u becomes a border in lot i, we add it to the end of list b_i and update $M(u) \leftarrow i$.
- $remove(i, u)$: when unit u ceases being a border of lot i, we look up u's node in

b_i via $M(u)$ and erase it. Since b_i is a doubly-linked list, erasing a node is done in $O(1)$.

- $next(i, u)$: given a unit $u \in B_i$, we can find another unit in B_i by accessing the *next* pointer in $M(u)$. By applying *next* successively on this node until we return to $M(u)$, we can iterate over all units of B_i without repetition. Note again that units iterated in sequence are not necessarily adjacent.

2.2.1 Fitness function

Our proposed algorithms satisfy the connectivity and accessibility constraints by design, but not necessarily the balance and equality constraints. In order to handle incomplete or infeasible solutions, our fitness function considers not only the sum of squares SS but also the severity of the violation of these constraints. The violation of the balance constraint is measured by the total excess area A of all lots with access to water over the area of the smallest lot without access to water, normalized by $|L|$, and the violation of the equality constraint is measured by $\lambda^+ = \max(\lambda - \bar{\lambda}, 0)$ (recall that λ is the ratio between the largest and smallest lot areas, and $\bar{\lambda}$ is a fixed maximum threshold). We then consider the fitness function $\varphi = (A, \lambda^+, SS)$ in lexicographic order, i.e. we minimize first the violation of the balance constraint, then the violation of the equality constraint, followed by the standard deviation of the lot aptitude. This order was chosen because the balance constraint is considerably more difficult to satisfy, and thus should be prioritized. When comparing two candidate solutions, we use lazy evaluation to avoid computing parts of φ that do not affect the outcome of the comparison, i.e., if two solutions differ in A, we do not need to compute their λ^+ and SS components.

2.2.2 Dynamic fitness updates

One of the most frequent tasks performed by our algorithm is recomputing φ after a change in the assignment of a single unit. If not performed efficiently, this task can easily become the bottleneck of the algorithm. In this section, we expose in detail how the components of φ can be updated efficiently after an unassigned unit u is assigned to a lot i. We omit the case of an assigned unit becoming unassigned or two lots swapping neighboring units, since it does not occur in our methods; we believe, however, that the

algorithms below can be easily adapted to accommodate such situations).

2.2.2.1 Updating SS

We can speed up the calculation of the objective function SS given a modification in the value v_j of a lot to $v'_j = v_j + \delta$, where δ is the aptitude value q_u of the unit gained, by expanding the sum of squares as

$$SS = \sum_{i \in [k]} (v_i - \bar{v})^2 = \sum_{i \in [k]} v_i^2 - k\bar{v}^2 = W - V^2/k,$$

where $W = \sum_{i \in [k]} v_i^2$ and $V = \sum_{i \in [k]} v_i$. Since we can update variables W and V in constant time under modification of some value v_j to v'_j, we also find the new sum of squares $SS' = W' - V'^2/k$ in constant time. If the number of spatial units that change is of order k or more, however, it is faster to first update all v_i, and then compute the new sum of squares SS'.

2.2.2.2 Updating A

In order to update the violations of the balance constraint A we maintain an array of lot indices sa sorted in ascending order of lot areas, a mapping id such that $sa(id(i)) = i$, and the index of the smallest lot l without water access. (We assume at least one lot does not have access to water; otherwise, A would always be zero and, since we only consider incremental updates, it would not be necessary to update it.) Then, A is updated as follows. When unit u is assigned to lot i, i's area increases by 1, and we update the ordered areas sa by swapping $sa(id(i))$ forward until $|C_{sa(id(i)+1)}| \geq |C_i|$, or until $id(i) = k$. This process is similar to an iteration of the bubble sort algorithm. With each swap, id is also updated accordingly. The worst case requires $O(k)$ swaps and happens when all lots have the same area. In a typical situation, however, the number of swaps is low. Next, we then have three different cases to consider:

- If $i \neq l$ and $|C_i| \geq |C_l|$ and ($B_i \cap R \neq \emptyset$ or c is next to water), i's violation is increased, so we increment A by 1.

- If $i = l$ and c is next to water, then i will not be the smallest lot without water access, as it gains its access through c. We then search for a new l by iterating forward on sa, starting from $id(i)$, until we find the next j such that $|C_j| \leq |C_l|$

and $B_j \cap R = \emptyset$. If $|C_l|$ increases, we then recompute A by iterating over sa. This process could take at most k steps, but it is performed only when $i = l$ and thus the cost is amortized over all possible i.

- If $i = l$ and c is not next to water, we subtract from A the number of lots with area equal to $|C_i| + 1$ that have access to water. These can be found by iterating over the positions in sa adjacent to $id(i)$. As before, there may be at most k such positions, but the cost is amortized over all possible i. If, however, there is another lot j with $B_j \cap R = \emptyset$ and $|C_j| = |C_i|$, A remains unchanged and we update l. Once again, this can be detected by a simple linear search on sa.

2.2.2.3 Updating λ^+

The smallest and largest lots are already maintained the sa array of the previous section, so it is trivial to maintain the area ratio $\lambda^+ = \max(0, sa(k)/sa(1) - \bar{\lambda})$.

2.3 Constructive heuristic

The initial population of the GA, as well as the r new solutions introduced to the population at each iteration, are generated using a randomized greedy constructive heuristic. The heuristic follows the location-allocation approach of Kalcsics, Nickel and Schröder (2005) and thus consists of two phases. In the location phase, we heuristically select k random, dispersed land units to serve as initial seeds for each lot. Then, in the allocation phase we expand the lots from their respective seeds by iteratively giving them a greedily-selected unit on their border, until the solution is complete.

2.3.1 First phase: location

Because water bodies or natural preservation areas may divide the usable land into several connected components, we perform a preprocessing step which identifies the connected components of the land graph G. The location phase starts by selecting k seed units at random from L, such that each connected component receives a number of seeds proportional to its area. It is assumed that no connected component in G has size smaller than $|L|/k$; otherwise, a preprocessing step moves the units in these components from

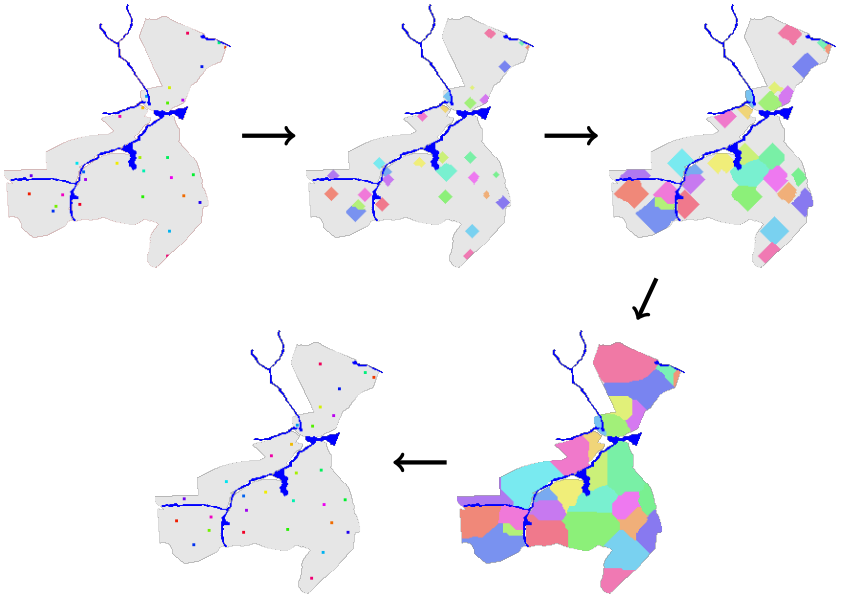
L to R. We do this in order to avoid early on that lots be placed in small regions that constrain the lot's maximum size and aptitude, thus never allowing it be balanced with respect to the other lots.

Having selected the k random seeds, the algorithm then expands them into regions by a shortest distance graph search on a directed version of G where the weight of an arc (u, v) is q_v. The search maintains a set of active units, which are initially the k seeds, and a mapping of units to their "parent" lots. It repeatedly selects the active unit u with shortest distance from the start, and assigns it to its parent lot. Neighbors of u that have not been discovered yet or whose distances to the start can be improved by a path through u are then set as active and assigned the same parent as u. The search ends when all units have been assigned. Using a priority queue to order the active units allows us to select the next one in $O(\log |L|)$ at each step, and because G is planar and thus has at most $O(|L|)$ edges, the search algorithm runs in $O(|L| \log |L|)$. In other terms, we effectively compute the discrete weighted Voronoi regions of the selected seeds, with respect to land aptitude. This algorithm by design generates connected regions and, given a reasonably smooth distribution of soil aptitudes, which is the case for most instances, also convex and without enclaves.

Finally, we recompute the k seeds as the geometric centroids of their respective regions. If a centroid falls outside of L, we move it to the closest reachable land unit. Similarly to the well-known k-means clustering algorithm (LLOYD, 1982), this approach could be applied in an iterated fashion by re-expanding these new seeds into regions using the same shortest-distance graph search, then reassigning them as the regions' centroids, then re-expanding them again and so on, until the seeds converge to the same location in two successive iterations. We have found, however, that repeating this process more than once was relatively time-consuming and did not yield a significant improvement.

The location process is illustrated in Figure 2.1. Algorithm 2 describes the location phase in detail. First, the connected components cc of G are identified (line 2) and an appropriate number of seeds per component is chosen (lines 5–6). Next, we initialize the data structures used for the graph search with the k seeds $s_1, \ldots, s_k$ (lines 8–16). Then, we iteratively select the nearest active node with respect to $dist$ (line 18), with ties broken in favor of the oldest active unit, assign it to its parent region (line 19), and set its undiscovered neighbors (with $dist = \infty$) as being active with the same parent (lines 21–25). If the new distance of an active neighbor through the current unit is improved, we update it. Finally, when there are no more active nodes, we compute and return the

Figure 2.1: The location process in instance Veredas. The arrows represent the sequence of operations. Top left: the initial seeds chosen at random. Top center, top right, bottom right: expansion of seeds into weighted Voronoi regions. Bottom left: final seeds recomputed as the centroids of each expanded region. It is easy to see that the final seeds are more evenly dispersed than the original ones.



geometric centroids of the expanded regions (lines 27–28).

2.3.2 Second phase: allocation

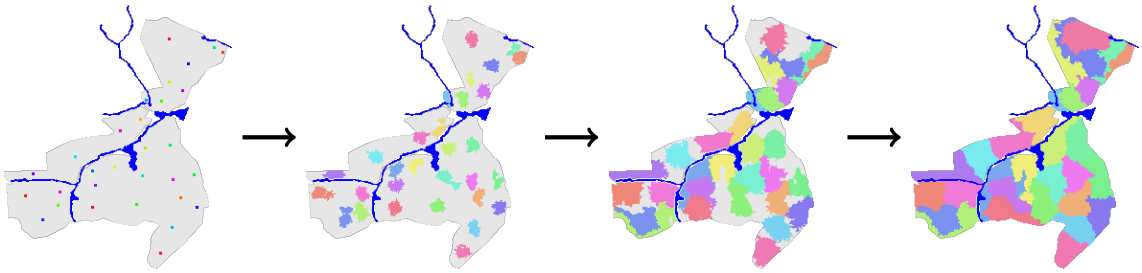
Once defined the initial seed for each lot, we expand them into regions a second time using a greedy constructive algorithm. The algorithm maintains a list of valid candidate assignments of a unit to a lot, and iteratively applies the best candidate with respect to the objective function φ, until there are no more possible assignments. An assignment is considered feasible if it satisfies the connectivity and accessibility constraints.

Differently from the location phase, in this phase we cannot use a priority queue to order the active candidates, since the cost φ of each candidate is subject to change after any assignment. Thus, at each step the algorithm must recompute the cost of all candidates. Even using the fast dynamic fitness updates described in Section 2.2.2, this is the main bottleneck of our method, since the number of candidates is usually very large. On the other hand, given a reasonably fine-grained discretization of the map, the objective function φ typically changes very little from one step to the next. For this reason, to speed up the process we define a batch size b and, at each step, perform the best

Algorithm 2 Greedy constructive algorithm: location phase

1: **procedure** LOCATION
2: $cc \leftarrow$connectedComponents(G)
3: $j \leftarrow 0$
4: **for** $c \in cc$ **do**
5: $f \leftarrow \lfloor k|c|/|L| \rfloor$
6: $(s_j, \ldots, s_{j+f}) \leftarrow$ randomChoice(c, f)
7: $j \leftarrow j + f$
8: **for** $u \in [L]$ **do**
9: $dist(u) \leftarrow \infty$
10: $active(u) \leftarrow$ false
11: $parent(u) \leftarrow -1$
12: **for** $i \in [k]$ **do**
13: $C_i \leftarrow \emptyset$
14: $dist(s_i) \leftarrow 0$
15: $active(s_i) \leftarrow$ true
16: $parent(s_i) \leftarrow i$
17: **repeat**
18: $u \leftarrow$active node with smallest $dist(u)$
19: $C_{parent(u)} \leftarrow C_{parent(u)} \cup \{u\}$
20: $active(u) \leftarrow$false
21: **for** $v \in N(u) \cap L$ **do**
22: **if** $dist(v) > dist(u) + q_v$ **then**
23: $active(v) \leftarrow$true
24: $dist(v) \leftarrow dist(u) + q_v$
25: $parent(v) \leftarrow parent(u)$
26: **until** there are no more active nodes
27: $s_i \leftarrow$geometric centroid of C_i $\forall i \in [k]$
28: **return** $(s_1, \ldots, s_k)$

Figure 2.2: The allocation process in instance Veredas. Arrows represent sequence of operations. Far left: the initial seeds computed by the location heuristic. Center left, center right: partial solutions during the allocation process. Far right: the complete solution.



b candidate assignments. If we maintain a list candidates in memory, we can select the best b in average time $O(n)$ using a selection algorithm, for n candidates.

Figure 2.2 illustrates the allocation process. The allocation process is detailed in Algorithm 3. We are given as input a partial solution S with non-empty lots. The algorithm populates the unassigned land units and returns a complete solution that satisfies connectivity and accessibility. (We use a partial solution as input rather than a set of seed units because this procedure is reused in further components; when applied in sequence to Algorithm 2, we can simply define $C_i = \{s_i\}$.) The algorithm starts by computing a set of candidate assignments K by iterating over unassigned border units (line 3). Then, it repeatedly selects the best b elements from K with a selection algorithm (line 5) and performs the respective assignments (line 8), each time updating the candidate set K (lines 9–10) with the unassigned neighbors of the units assigned. By adding to K only border units, we ensure that the lots are connected. Before each assignment we test if the respective unit is still unassigned, since the same unit may appear in more than one candidate in B. Note that the order of the elements in B is defined by the selection algorithm used. The algorithm stops when there are no more candidates, and returns the current solution.

2.4 Recombination

Recombination operators typically operate on two "parent" solutions and aim to create a "child" solution that, while different from the parents, maintains some of the common structure between them, therefore introducing diversity to the population. It is expected that high-quality parent solutions will produce high-quality children, and so the parents are typically selected among the best in the population. Additionally, applying a

Algorithm 3 Greedy constructive algorithm: allocation phase

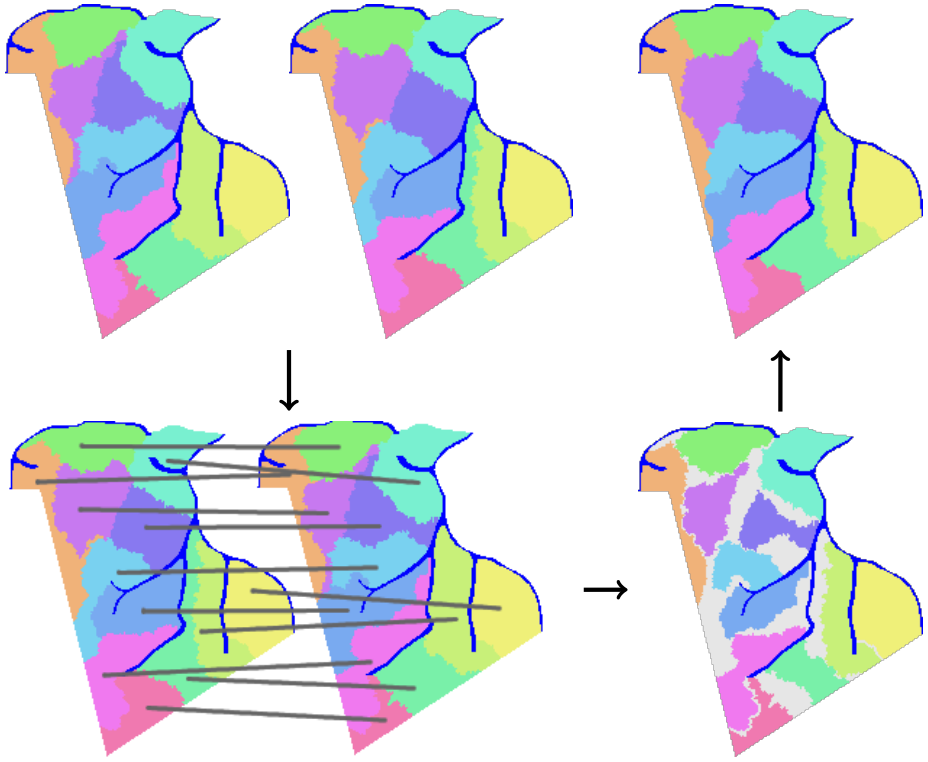
```
1:  procedure ALLOCATION
2:      input: a partial solution S = (C_1, ..., C_k)
3:      K ← ⋃_{i∈[k]} {(u, i) | u is unassigned ∧ u ∈ B_i}
4:      repeat
5:          B ← best b candidates in K with respect to φ
6:          for (u, i) ∈ B do
7:              if u is unassigned then
8:                  S(u) ← i
9:                  K ← K ∪ {(v, i) | v is unassigned ∧ v ∈ N(u)}
10:                 K ← K \ {(u, j) | j ∈ [k]}
11:     until K = ∅
12:     return S
```

recombination operator is usually much faster than constructing a new solution from the beginning, which allows us to explore more solutions in the same time.

Our recombination operator attempts to maintain contiguous regions that are assigned to a single lot in both parent solutions as also being assigned to a single lot in the child solution. We first make the observation that lots in PROTERRA are *anonymous*, i.e. there is no direct correspondence between lots in different solutions. For example, the lot with index i in one solution is likely to have no spatial relation to the lot with index i in another solution. In order to establish such a correspondence, we construct a complete weighted bipartite graph where the nodes in each part correspond to lots in each of the solutions, and the weight of an edge connecting two nodes is equal to the area of intersection of the corresponding lots. We then compute a maximum weight perfect matching on this graph, and construct a child solution whose lots correspond to the intersections of the parents' matched lots. A special case occurs when a pair of matched lots do not intersect: in this case, we introduce the new lot as a single unit chosen randomly over the unassigned land units. Finally, the child solution is reconstructed to completion using the greedy allocation procedure of Section 2.3.2. The whole recombination process is illustrated in Figure 2.3.

Algorithm 4 presents the method in detail. First, a complete bipartite graph H is created as explained above (line 3), and a maximum cost perfect matching M is computed for it (line 4). Then, we assign each child lot C_{3i} to the intersection of lot i in the first parent and its match in the second parent (lines 6–10). If this intersection is empty, we add this lot to the set E (line 10), and in a later step assign to it a random unassigned seed unit in L (lines 11 – 12). This is done in a posterior step to avoid that the selected seed be overridden by some other lot. Finally, we reconstruct the child solution $(C_{31}, \ldots, C_{3k})$

Figure 2.3: The recombination operator applied to instance Fortaleza. The number of lots was reduced from 40 to 12 to facilitate visibility. Arrows represent the sequence of operations. Top left: the two parent solution. Bottom-left: the matching between the lots in both parent solutions. Bottom right: the child solution with only the intersection of the parents' matched lots. Top left: the child solution after reassigning the free space using the constructive algorithm.



using Algorithm 2 and return it.

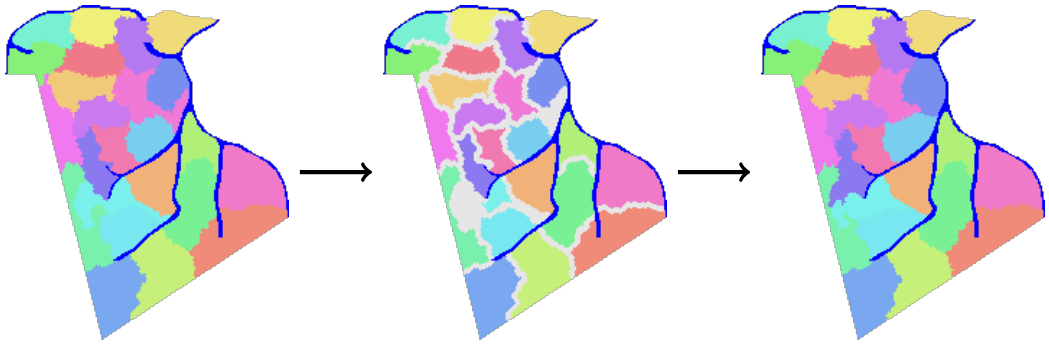
Algorithm 4 Recombination

procedure RECOMBINATION
  input: parent solutions $(C_{11}, \ldots, C_{1k})$ and $(C_{21}, \ldots, C_{2k})$
  $H \leftarrow$ a complete bipartite graph where an edge linking lots C_{1i} and C_{2j} has weight $|C_{1i} \cap C_{2j}|$, for $i, j \in [k]$.
  $M \leftarrow$ maxCostPerfectMatching(H)
  $E \leftarrow \emptyset$
  for $i \in [k]$ **do**
    if $|C_{1i} \cap M(C_{1i})| \neq \emptyset$ **then**
      $C_{3i} \leftarrow C_{1i} \cap M(C_{1i})$
    else
      $E \leftarrow E \cup \{i\}$
  for $i \in E$ **do**
    $C_{3i} \leftarrow \{$a random unassigned unit in $L\}$
  return Allocation$((C_{31}, \ldots, C_{3k}))$

2.5 Mutation

A mutation operator aims to introduce variability to the population by making a small, random perturbation to an existing solution. We implement mutation in our algorithm by "erasing" (i.e., setting as unassigned) all units within a given distance of the lot borders, and then reconstructing the partial solution using the greedy allocation heuristic. Erasing the borders can be done by a fixed-distance breadth-first search (BFS) seeded by all units in $B_i, i \in [k]$. We have chosen this distance to be a fixed 2 spatial units. As in the recombination, if a lot ends up being erased completely, we reintroduce it as randomly-chosen seed. Further, if the the erasing step disconnects a lot, we discard all but the largest connected component of that lot. Because the greedy allocation heuristic is deterministic, we ensure variability in successive mutations of the same solution by introducing a parameter α to the constructive heuristic: instead of executing the best b assignments at each step, we execute b assignments chosen uniformly at random among the αb best. This selection can be done in time $O(n)$ for n candidates using a selection algorithm. The mutation process is illustrated in Figure 2.4.

Figure 2.4: The mutation operator applied to instance Fortaleza, with a reduced number of lots. Arrows represent the sequence of operations. Left: the original solution. Center: after removing units near the borders between lots. Right: greedily reconstructed solution.



2.6 Computational experiments

2.6.1 Test instances

We have conducted experiments on five real-world instances: "Veredas" (Minas Gerais, Brazil), "Olhos D'Água" (Minas Gerais, Brazil), "Iucatã" (Acre, Brazil), "Belo Vale" (Minas Gerais, Brazil) and "Fortaleza" (Acre, Brazil). These instances were first used by (FERREIRA, 2015), to which we refer the reader for more details on them. Figure 2.5 shows the topologies of instances Olhos D'Água, Belo Vale and Fortaleza (instance Veredas was already shown as an example in Figure 1.1).

For a more thorough experimentation, we have also generated an additional 25 artificial instances of sizes $n \times n$, with $n \in \{200, 300, 400, 500, 600\}$, and number of lots $k \in \{20, 40, 60, 80, 100\}$. For a given size n, we use the same artificial map topology for the 5 levels of k. These levels have been chosen to match the typical range of the real-world instances.

In order to generate the artificial instances we use Perlin noise (PERLIN, 1985), a well-known noise function which maps a 2D coordinate to a value in the range $[-1, 1]$. Perlin noise is typically used in applications such as generating random height maps or adding the appearance of realism to textures. It has four main parameters: frequency f, number of octaves o, granularity r and gain g. In preliminary experimentation we have fixed these to $f = 10^{-3}, o = 16, r = 2$ and $g = 0.5$. We create the instance's soil by generating an $n \times n$ noise matrix and quantizing it by a desired number of classes of aptitude. The value of each class of aptitude is chosen randomly from $[30, 100]$. Perlin noise ensures that transitions in aptitude between adjacent regions are smooth, and areas

Figure 2.5: Map topologies of the real-world instances Belo Vale (top left), Fortaleza (top right), Iucatã (bottom left) and Olhos D'Água (bottom right).

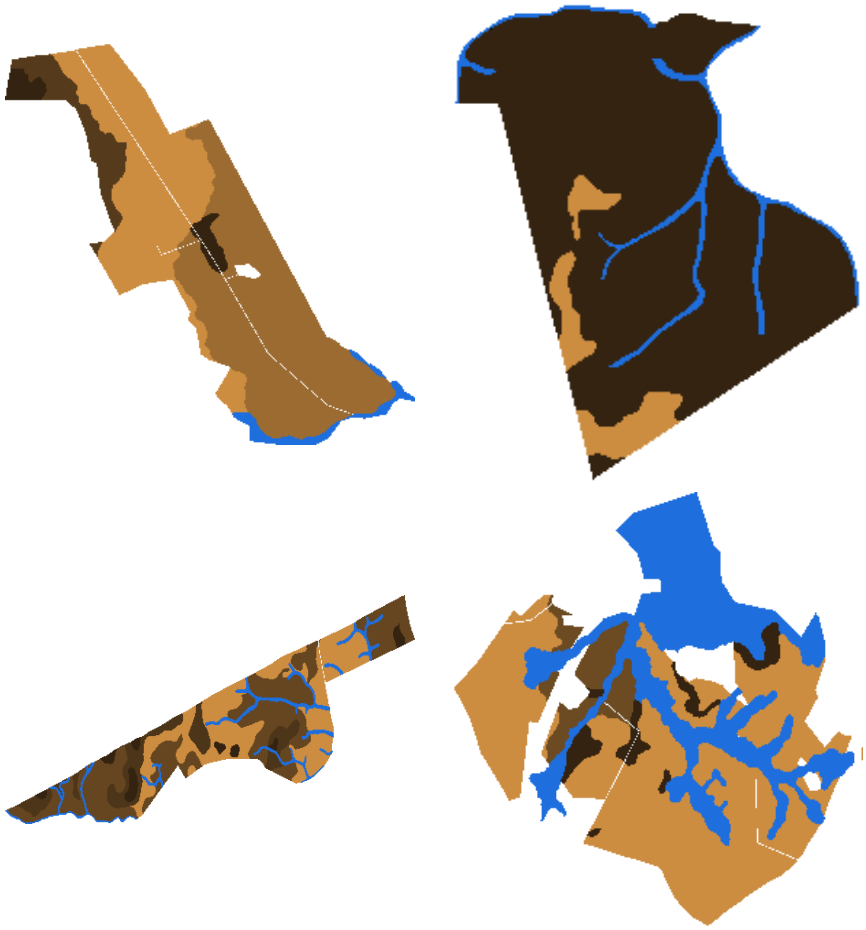
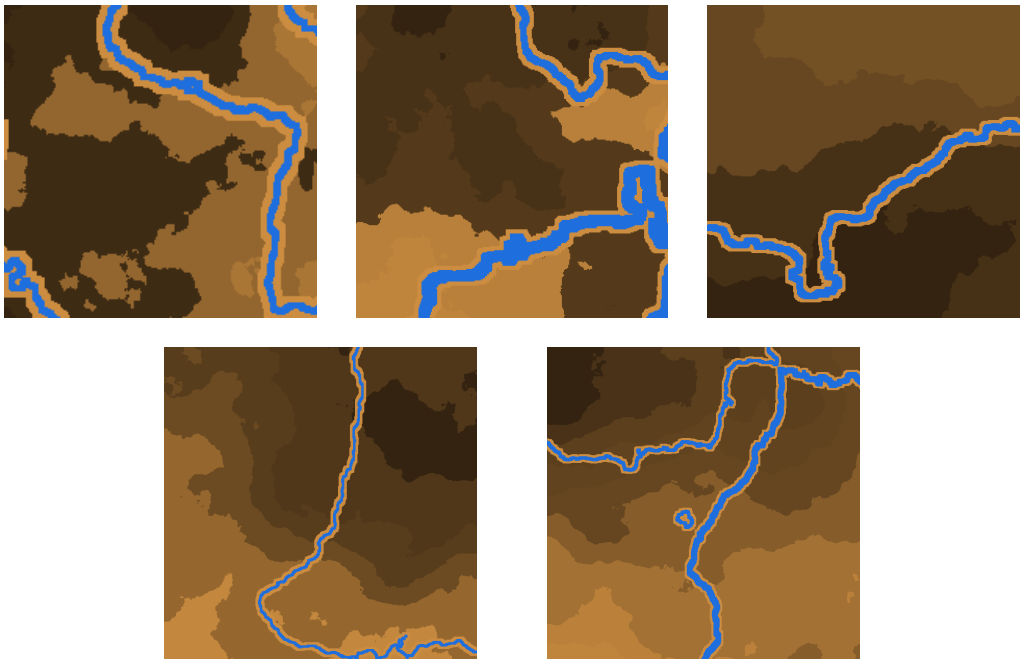


Figure 2.6: The 5 artificial topologies generated, ordered by size from left to right, top to bottom.



generated with the same level of aptitude are somewhat regular and with few connected components.

Next, given a desired percentage of water units, we generate water bodies by a second Perlin noise matrix with $o = 10$ and whose parameter f is binary searched in $[0, 1]$. At each step we generate a new Perlin noise matrix, quantize it into two levels, define river outlines with the cells on the border between the two levels, and expand these outlines by a BFS with a fixed width chosen randomly in the integer range $[2, 5]$. If the total number of water cells is within 1% of the desired percentage we accept the current matrix and overlay it onto the one defining the instance's soil; otherwise, we adjust f accordingly and continue. After the desired percentage is met, we replace connected components of land of area smaller than $|L|/2k$ by water, in order to prevent island-like structure that are unlikely to appear in a real-life scenario. Finally, we set the aptitude of all cells within 5 units of a river to be the highest aptitude class. In each instance, the desired number of water units was chosen randomly between 2% and 5%. For the artificial instances we chose not to generate natural preservation areas, as they do not contribute to the objective function and we have found them to have little impact difficulty. The 5 artificial topologies generated are shown in Figure 2.6.

Table 2.1 summarizes the characteristics of the test instances. The top five rows correspond to the five real-world instances, and the bottom five rows to the five artificial

Table 2.1: Characteristics of the test instances.

Name	w	h	k	Apt.	Land (%)	River (%)	Res. (%)	#CC
Belo Vale	300	300	30	5	41.6	1.0	57.4	1
Fortaleza	449	250	40	2	20.7	1.2	78.1	1
Iucatã	449	250	40	5	26.1	2.5	71.4	12
Olhos D'Água	300	300	24	4	38.2	15.8	46.0	9
Veredas	300	300	26	6	42.0	1.5	56.5	10
	200	200		5	94.4	5.6	0.0	7
	300	300		6	90.9	9.1	0.0	4
Artificial	400	400		5	95.8	4.2	0.0	2
	500	500		7	98.0	2.0	0.0	7
	600	600		10	95.2	4.8	0.0	4

topologies. We report the map width (w) and height (h) in spatial units, the number of lots k (relevant only for the five real-world instances), the number of different classes of soil aptitude (Apt.), the percentage of the total area representing land, water and natural preservation areas, and the number of connected components of land regions (#CC).

2.6.2 Methodology

We have implemented our algorithms in C++ and compiled them with GCC 5.3.1 and maximum optimization. The tests have been executed on a PC with an 8-core AMD FX-8150 processor and 32GB of main memory. Each test was executed on a single core. The code, instances, instance generator and full results of the experiments reported below are available to the community at <http://www.inf.ufrgs.br/algopt/proterra/>.

In the following we report on four experiments. The first determines the best batch size b of the constructive heuristic. Next, we calibrate the parameters of the GA. The second experiment studies how the GA scales with different instance sizes, the third evaluates its effectiveness in comparison to simpler methods, and the last compares its results to known solutions of real-world instances.

2.6.3 Experiment 1: batch sizes

This experiment aims to choose the best batch size b which determines the number of unit assignments performed at each step of the greedy constructive heuristic. We have performed two tests on batch sizes $b \in \{32, 64, 128, 256, 512, 1024, 2048, 4096\}$, using

Table 2.2: Results of the calibration of the optimal batch size.

Batch size	Fixed time				50 replications			
	$A\ (m)$	λ	σ	Repl.	$A\ (m)$	λ	σ	t (s)
32	40.3	4.3	44.6	2,494	28.7	4.0	35.7	668
64	28.4	3.8	27.6	6,118	35.5	3.8	29.3	301
128	28.4	3.7	32.0	13,760	34.4	4.3	36.7	120
256	23.3	4.1	31.8	27,102	44.2	5.5	42.0	52
512	28.3	4.3	44.5	45,790	53.8	4.0	34.7	26
1,024	42.2	4.2	41.5	67,831	73.6	4.8	36.8	15
2,048	51.1	4.5	65.4	88,424	94.3	6.8	58.8	10
4,096	67.9	6.4	82.4	111,203	118.7	6.3	81.6	8

only the 25 artificial instances. The first test executes the constructive heuristic for a fixed number of 50 replications, reporting the best one in the end, and aims to evaluate how increasing batch sizes affect running time and solution quality. The second test executes as many replications of the algorithm as possible within a fixed time limit of 10 minutes, and reports the best one in the end.

The results are shown in Table 2.2. For each test, we report the violation of the balance constraint A $(\times 10^{-3})$, of the equality constraint λ, and the standard deviation in soil aptitude σ, averaged over the 25 instances. For the test with a fixed time limit, we report the average number of replications performed ("Repl."), and for the test with a fixed number of replications, the average running time ("t") in seconds.

As expected, because larger batch sizes require fewer construction steps, with a fixed time limit the number of replications increases as b increases, while with a fixed number of replications larger b yield lower running times. In both tests, since most of the replications did not achieve feasible solutions with respect to balance, the equality constraint was typically not satisfied and the soil aptitude deviation fluctuates between 30% and 80%, so A is a good estimate of the quality of a solution.

With a fixed number of replications, the solution quality degrades as b increases, as larger values of b yield coarser approximations of the "pure" constructive heuristic (with $b = 1$), but remains still low at $b = 4096$ with $A = 1.19\%$ of the total area. With a fixed time limit, the final fitness function φ has its best average value with $b = 256$, and degrades as b grows too small or too large. Once again, this occurs because larger b give worse approximations, whereas smaller values of b require more running time per replication and thus produce fewer replications. For this reason, we chose to fix $b = 256$ for the remaining experiments.

Table 2.3: Parameters of the GA: initial ranges and optimal setting found by iterative racing.

Description	Initial range	Best value
Population size p	$[10, 50]$	15
Mutation α	$[2, 5]$	3.72
Elite rate e	$[0, 1]$	0.59
Offspring rate o	$[0, 1]$	0.38
Random rate r	$[0, 1]$	0.03

2.6.4 Parameter calibration

With a fixed $b = 256$, we have calibrated the parameters of the GA with the irace package in GNU R (LÓPEZ-IBÁÑEZ et al., 2016), with a budget of 1000 runs and a time limit of 5 limits per run, over all the instances in our data set. Table 2.3 displays the parameters, their calibration ranges and the best settings found by iterative racing.

2.6.5 Experiment 2: scalability

In this experiment we analyze how our methods scale as instance sizes grow. We have performed 5 replications of the algorithm on each of the 25 artificial instances, with a running time of 30 minutes per replications. Table 2.4 shows the results. As before, we report the violation of the balance constraint A ($\times 10^{-6}$), of the equality constraint λ, and the standard deviation in soil aptitude σ ($\times 10^{3}$), averaged over the 5 replications. We also report the number of feasible solutions out of the 5 replications ("Feas."), and the number of evaluated solutions ("Evals.").

The GA found feasible solutions on the majority of replications (112 out of 125), being unable to achieve feasibility only for the instance of size 200×200 with 100 lots. We believe this is because of the large number of lots in a very small area, which makes it harder to satisfy the balance constraint. For the instance of size 300×300 and 20 lots only one replication was feasible: the other 4 were able to satisfy the balance but not the equality constraint. We can also see that the soil aptitude deviation σ decreases as k increases. This is expected: as there are more lots to allocate, the average soil aptitude per lot decreases. Finally, we see that the number of fitness evaluations declines as both the number of lots and instance sizes grow, as expected. Empirically, the number of evaluations is about $2.1 \times 10^{12}\, n^{-2.7}\, k^{-0.8}$ (log-log regression, $R^2 = 0.9812$) for instances

Table 2.4: Scalability experiment.

Size	Lots	Evals.	A (μ)	λ	σ (K)	Feas.
200x200	20	138,732	0.00	1.70	5.66	5
	40	79,999	0.00	1.75	1.83	5
	60	56,350	0.00	1.81	1.78	5
	80	43,927	0.00	1.84	1.38	5
	100	37,721	436.24	5.39	5.10	0
300x300	20	60,013	0.00	3.40	81.19	1
	40	31,882	0.00	1.90	10.40	5
	60	22,982	0.00	1.88	6.93	5
	80	17,937	0.00	1.91	5.33	5
	100	15,214	0.00	2.11	4.70	5
400x400	20	22,761	0.00	1.29	26.83	5
	40	13,254	0.00	1.43	12.93	5
	60	8,923	0.00	1.46	7.83	5
	80	7,135	0.00	1.51	5.75	5
	100	6,237	0.00	1.57	7.07	5
500x500	20	9,970	0.00	2.39	66.52	5
	40	5,704	0.00	2.62	32.37	5
	60	4,058	4.12	6.17	28.19	4
	80	3,143	0.00	2.73	18.57	5
	100	2,714	0.00	2.81	14.54	5
600x600	20	7,502	0.00	1.89	297.75	5
	40	5,384	0.00	2.20	128.50	5
	60	4,125	11.10	2.45	82.63	4
	80	3,396	35.63	3.45	58.65	3
	100	2,233	0.00	2.54	47.35	5

of size $n \times n$ with k lots, so the cost grows slightly more than the total number of cells, and close to linear with the number lots. This is reasonable for current instance sizes, but may be a bottleneck for very large or very fine-grained instances.

2.6.6 Experiment 3: effectiveness of the genetic algorithm

This experiment evaluates the effectiveness of the GA by comparing it to two simpler approaches: one that generates lot seeds uniformly at random and then expands them with a simple BFS, and the constructive heuristic of Section 2.3. Both approaches repeatedly generate new solutions within the specified time limit and report the best one at the end. For the second approach, we have also used $b = 256$. Note that the second approach is equivalent to the GA with $P = r = 1$ and $e = o = 0$.

For this experiment we have used the 5 real-world instances and a time limit of 30

Table 2.5: Effectiveness experiment.

Instance	Evals. (m)			A (m)			λ			σ (% r.d.)		
	BFS	Cons.	GA	BFS	Cons.	GA	BFS	Cons.	GA	BFS	Cons.	GA
Belo Vale	65.9	2.0	14.4	0.1	0.0	0.0	9.99	2.99	1.83	657.4	196.6	5.9
Fortaleza	83.2	2.4	16.2	38.7	25.1	0.0	59.79	4.86	1.19	1,025.7	479.4	4.7
Iucatã	62.9	3.6	34.5	0.0	64.8	0.0	10.26	18.59	2.83	582.9	463.0	63.0
Olhos D'Água	75.5	5.2	59.8	0.0	82.7	0.0	5.67	8.97	1.93	165.2	54.8	9.7
Veredas	146.1	2.6	23.4	1.9	49.0	0.0	22.62	3.87	1.70	424.8	95.7	12.1

minutes, and report averages over 5 replications. Table 2.5 shows the results. We report, for each approach, the number of evaluated solutions ($\times 10^4$), the violation of the balance constraint A ($\times 10^{-3}$), the violation of the equality constraint λ, and the standard deviation in soil aptitude σ as the relative deviation, in percent, from the best known value for that instance.

We can see that the BFS is better at satisfying the balance constraint than the greedy constructive heuristic in 3 of the 5 instances (Iucatã, Olhos D'Água and Veredas). However, it performs significantly worse when it comes to the equality constraint, producing solutions where the largest lot is up to 59.79 times larger than the smallest lot on average, for the instance Fortaleza. Further, since it does not account for soil aptitude, the BFS approach leads to significantly higher soil aptitude deviations. The GA found feasible solutions for all instances, with significantly lower deviations in soil aptitude among lots. As expected, being algorithmically simple the BFS explored the most solutions, while the constructive heuristic the fewest, since it constructs full solutions whereas the GA mostly reconstructs partial solutions during mutation and recombination.

2.6.7 Experiment 4: comparison to manual allocation

Our final experiment compares the results of the GA to the existing manual allocations by the responsible government entity INCRA. We have excluded instance Fortaleza from this comparison as we did not have access to the official allocation. Because the manual allocations do not consider water obstacles, thus allowing lots to have land on both sides of a river, for example, we do not penalize solutions that violate the connectivity constraint. We have executed the GA with 30 minutes of running time and report averages over 5 replications. Table 2.6 shows the results.

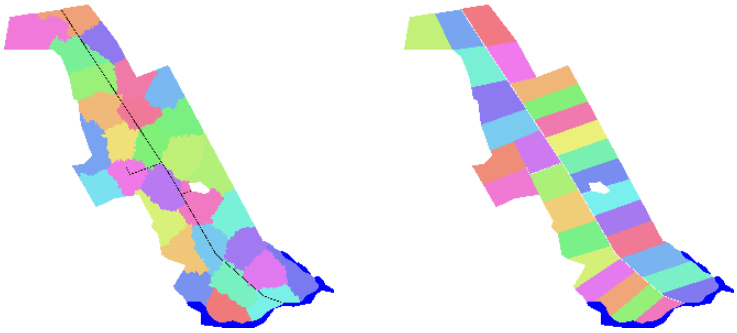
We can observe that, in all cases, the GA produces solutions that are better with respect to all three components of the objective function. The manual allocations violate

Table 2.6: Comparison of our genetic algorithm to manual allocation made by INCRA.

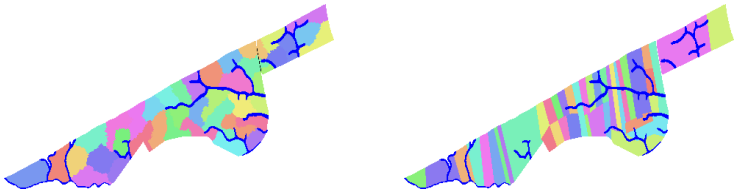
Instance	A (m)		λ		σ	
	Manual	GA	Manual	GA	Manual	GA
Belo Vale	19.7	0.0	2.77	1.83	5,378.5	1,888.6
Iucatã	735.9	0.0	22.38	2.83	18,444.9	2,806.5
Olhos D'Água	830.5	0.0	28.81	1.93	19,972.8	11,716.1
Veredas	0.0	0.0	5.78	1.70	3,964.4	1,161.5

the balance constraint in instances Belo Vale, Iucatã and Olhos D'Água and the equality constraint in instances Iucatã, Olhos D'Água and Veredas, whereas the GA found feasible solutions for all four instances. The standard deviation in soil aptitude among lots found by the GA is a factor of 1.7 to 7 times lower. Figure 2.7 visually compares the best solution among the 5 replications and the official allocation.

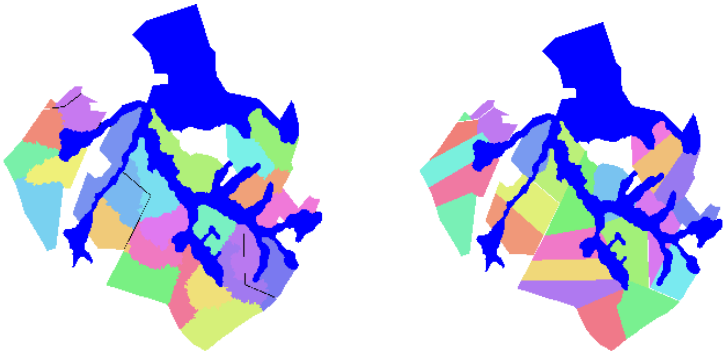
Figure 2.7: Comparison between the allocation produced by the GA (left) and the manual allocation done by INCRA (right) for instances Belo Vale 2.7a, Iucatã 2.7b, Olhos D'Água 2.7c and Veredas 2.7d.



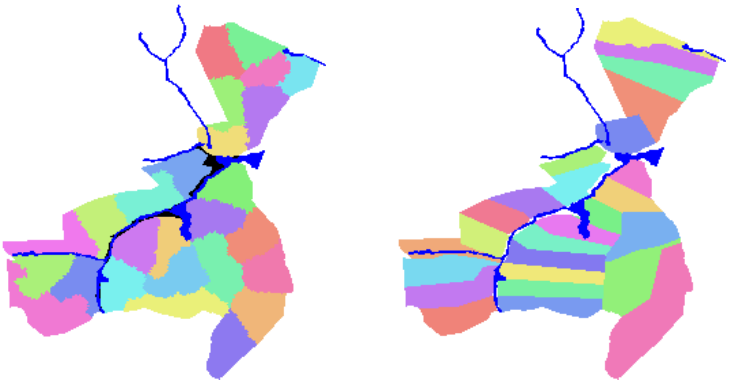
(a) Belo Vale.



(b) Iucatã.



(c) Olhos D'Água.



(d) Veredas.

3 CONCLUSION AND FUTURE WORK

We have introduced the Territorial Organization in Agrarian Reform Projects and Environmental Planning Problem (PROTERRA). It consists of subdividing a large parcel of farmland into smaller lots that are balanced with respect to area, soil aptitude and access to hydric resources. Inspired by previous works and its similarity to districting models, we represent instances as regular 2D grids, which can be described by weighted planar graphs. To solve PROTERRA we propose a greedy location-allocation approach to generate new solutions and a genetic algorithm with problem-specific recombination and mutation operators. Experimental results show that the GA scales reasonably well with the size of the instance, and significantly outperforms simpler approaches, indicating that the proposed operators are effective. In comparison to current manual allocations the GA led to fairer allocations in all instances considered, suggesting that it could be an acceptable option for practical application. Still, we believe there is significant potential for further investigation.

Because we use a planar graph model, our solution structure is identical to most districting problems, and thus the genetic operators we have proposed can probably be applied effectively to other districting problems for which a good randomized constructive heuristic exists. Further, although PROTERRA can be modeled as a districting problem, we have not considered lot compactness directly. Currently there is no formal regulation regarding the expected shape of the lots, but it is understood that shapes should ideally be convex polygons with few edges. We believe that integrating a formal compactness measure into the solver is an essential step in order to bring current solutions closer to practical application.

Currently one of the main bottlenecks are large instance sizes: with an ideal cell precision of around 100m^2, instances would have in the order of 10^7 to 10^8 nodes, meaning heuristic operators with a complexity beyond linear are too costly to perform repeatedly. We believe we can exploit the fact that we work with planar graphs, and in particular regular grid graphs, in order to implement more efficient methods for dynamic operations such as connectivity and accessibility queries upon swapping spatial units between lots. This opens the way to local search-based heuristics, such as GRASP or tabu search, which could greatly improve the quality of current solutions.

Finally, it could be useful to explore purely geometric representations of the problem. There are several approaches which have been used successfully in the literature both

in districting and land allocation, such as the genetic algorithm of Demetriou, See and Stillwell (2013b) which optimizes lots through Voronoi diagrams, the clustering-based approaches of Brieden, Gritzmann and Klemm (2017) and Borgwardt, Brieden and Gritzmann (2014) which group spatial units based on geometric clustering, or the recursive greedy bisection algorithm of Kalcsics, Nickel and Schröder (2009), which constructs lots by recursively bisecting an area into two balanced partitions.

ACKNOWLEDGEMENTS

I would like to thank my advisor, my colleagues and my girlfriend Micheli for their continuous support. The development of this thesis was funded by CNPq (grant 420348/2016-6), FAPEMIG (grant TEC-APQ-02694-16) and by Google Research Latin America (grant 25111).

REFERENCES

ASSIS, L. S. D.; FRANCA, P. M.; USBERTI, F. L. A redistricting problem applied to meter reading in power distribution networks. **Computers and Operations Research**, Elsevier, v. 41, n. 1, p. 65–75, 2014. ISSN 03050548.

BAÇÃO, F.; LOBO, V.; PAINHO, M. Applying genetic algorithms to zone design. **Soft Computing**, v. 9, n. 5, p. 341–348, 2005. ISSN 14327643.

BENZARTI, E.; SAHIN, E.; DALLERY, Y. Operations management applied to home care services: Analysis of the districting problem. **Decision Support Systems**, Elsevier B.V., v. 55, n. 2, p. 587–598, 2013. ISSN 01679236.

BLAIS, M.; LAPIERRE, S. D.; LAPORTE, G. Solving a home-care districting problem in an urban setting. **Journal of the Operational Research Society**, v. 54, n. 11, p. 1141–1147, 2003. ISSN 0160-5682.

BORGWARDT, S.; BRIEDEN, A.; GRITZMANN, P. Geometric Clustering for the Consolidation of Farmland and Woodland. **Mathematical Intelligencer**, v. 36, n. 2, p. 37–44, 2014. ISSN 03436993.

BOZKAYA, B. et al. Designing new electoral districts for the city of Edmonton. **Interfaces**, v. 41, n. 6, p. 534–547, 2011. ISSN 00922102.

BRIEDEN, A.; GRITZMANN, P.; KLEMM, F. Constrained clustering via diagrams: A unified theory and its application to electoral district design. **European Journal of Operational Research**, v. 263, n. 1, p. 18–34, 2017. ISSN 03772217.

BUI, Q. T.; PHAM, Q. D.; DEVILLE, Y. Solving the Agricultural Land Allocation Problem by Constraint-Based Local Search. In: **Lecture Notes in Computer Science**. [S.l.: s.n.], 2013. v. 8124, p. 749–757. ISBN 9783642406263.

BUTSCH, A.; KALCSICS, J.; LAPORTE, G. Districting for Arc Routing. **INFORMS Journal on Computing**, v. 26, n. October, p. 809–824, 2014. ISSN 1091-9856.

CAMACHO-COLLADOS, M.; LIBERATORE, F.; ANGULO, J. M. A multi-criteria Police Districting Problem for the efficient and effective design of patrol sector. **European Journal of Operational Research**, Elsevier Ltd., v. 246, n. 2, p. 674–684, 2015. ISSN 03772217.

CLEMENTS, E. A. Agrarian Reform, Food Sovereignty and the MST: Socio-environmental Impacts of Agrofuels Production in the Pontal do Paranapanema Region of São Paulo State, Brazil. **Revista NERA**, v. 15, n. 21, p. 8–32, 2012.

DEMETRIOU, D.; SEE, L.; STILLWELL, J. A parcel shape index for use in land consolidation planning. **Transactions in GIS**, v. 17, n. 6, p. 861–882, 2013. ISSN 13611682.

DEMETRIOU, D.; SEE, L.; STILLWELL, J. A spatial genetic algorithm for automating land partitioning. **International Journal of Geographical Information Science**, v. 27, n. 12, p. 2391–2409, 2013. ISSN 13658816.

DUQUE, J. C.; ANSELIN, L.; REY, S. J. The Max-p-Regions Problem. **Journal of Regional Science**, v. 52, n. 3, p. 397–419, aug 2012. ISSN 00224146.

ESRI, A.; PAPER, W. ESRI Shapefile Technical Description. **Computational Statistics**, v. 16, n. July, p. 370–371, 1998. ISSN 01679473.

FERREIRA, F. M. **Aptidão agrícola das terras como função de otimização para o ordenamento territorial e planejamento ambiental: uma anãlise do SOTER-PA**. 117 p. Dissertation (Master) — Programa de Pós-Graduação em Extensão Rural, Universidade Federal de Viçosa, Brazil, 2015.

FORMAN, S. L.; YUE, Y. Congressional Districting Using a TSP-Based Genetic Algorithm. In: **Genetic and Evolutionary Computation - GECCO 2003**. [S.l.: s.n.], 2003. p. 2072–2083. ISBN 3-540-40603-4.

GARCÍA-AYALA, G. et al. A novel model for arc territory design: Promoting Eulerian districts. **International Transactions in Operational Research**, v. 23, n. 3, p. 433–458, 2016. ISSN 14753995.

GENDREAU, M.; POTVIN, J.-Y. (Ed.). **Handbook of Metaheuristics**. 2nd. ed. [S.l.]: Springer, 2010.

GLIESCH, A.; RITT, M.; MOREIRA, M. C. O. A genetic algorithm for fair land allocation. In: **Genetic and Evolutionary Computation Conference - GECCO '17**. New York, New York, USA: ACM Press, 2017. p. 793–800. ISBN 9781450349208.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization, and Machine Learning**. [S.l.]: Addison-Wesley, 1989. ISBN 978-0201157673.

GUO, D.; WANG, H. Automatic Region Building for Spatial Analysis. **Transactions in GIS**, v. 15, n. SUPPL. 1, p. 29–45, 2011. ISSN 13611682.

HESS, S. W. et al. Nonpartisan Political Redistricting by Computer. **Operations Research**, v. 13, n. 6, p. 998–1006, 1965. ISSN 0030364X, 15265463.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems**. [S.l.]: University of Michigan Press, 1975.

KALCSICS, J. Districting Problems. In: **Location Science**. Cham: Springer International Publishing, 2015. p. 595–622. ISBN 978-3-319-13110-8.

KALCSICS, J.; NICKEL, S.; SCHRÖDER, M. Towards a unified territorial design approach - applications, algorithms and GIS integration. **Top**, v. 13, n. 1, p. 1–56, 2005. ISSN 1134-5764.

KALCSICS, J.; NICKEL, S.; SCHRÖDER, M. A generic geometric approach to territory design and districting. **Berichte des Fraunhofer ITWM Nr 153**, v. 153, n. 153, p. 42, 2009.

KING, D. M.; JACOBSON, S. H.; SEWELL, E. C. The geo-graph in practice: creating United States Congressional Districts from census blocks. **Computational Optimization and Applications**, Springer US, p. 1–25, 2017. ISSN 15732894.

LEI, H. et al. Dynamic design of sales territories. **Computers and Operations Research**, Elsevier, v. 56, p. 84–92, 2015. ISSN 03050548.

LI, W.; CHURCH, R. L.; GOODCHILD, M. F. The p-compact-regions problem. **Geographical Analysis**, v. 46, n. 3, p. 250–273, 2014. ISSN 15384632.

LLOYD, S. Least squares quantization in PCM. **IEEE Transactions on Information Theory**, v. 28, n. 2, p. 129–137, mar 1982. ISSN 0018-9448.

LÓPEZ-IBÁÑEZ, M. et al. The irace package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, v. 3, p. 43–58, 2016. ISSN 22147160.

MOREIRA, M. C. d. O. et al. O uso da busca tabu no ordenamento territorial em assentamentos rurais. **Desenvolvimento Rural, sustentabilidade e ordenamento territorial**, 2011.

MORENO-REGIDOR, P.; García López de Lacalle, J.; MANSO-CALLEJO, M.-Á. Zone design of specific sizes using adaptive additively weighted Voronoi diagrams. **International Journal of Geographical Information Science**, v. 26, n. 10, p. 1811–1829, 2012. ISSN 1365-8816.

MOURA, R. A. de et al. Reforma agrária e desenvolvimento: A reconstrução e uma questão polêmica. **ReBraM**, v. 14, n. 2, p. 95–106, 2011.

NETO, J. A. F. et al. Aptidão agrícola e algoritmos genéticos na organização espacial em projetos de reforma agrária. **Revista Brasileira de Ciência do Solo**, v. 35, n. 1, p. 255–261, 2011. ISSN 01000683.

PERLIN, K. An image synthesizer. In: **SIGGRAPH Comput. Graph.** [S.l.: s.n.], 1985. p. 287–296.

REIS, R. R. O direito à terra como um direito humano: a luta pela reforma agrária e o movimento de direitos humanos no brasil. **Lua Nova: Revista de Cultura e Política**, SciELO Brasil, n. 86, p. 89–122, 2012.

RICCA, F.; SCOZZARI, A.; SIMEONE, B. Weighted Voronoi region algorithms for political districting. **Mathematical and Computer Modelling**, v. 48, n. 9-10, p. 1468–1477, 2008. ISSN 08957177.

RICCA, F.; SCOZZARI, A.; SIMEONE, B. Political Districting: From classical models to recent approaches. **Annals of Operations Research**, v. 204, n. 1, p. 271–299, 2013. ISSN 02545330.

RICCA, F.; SIMEONE, B. Local search algorithms for political districting. **European Journal of Operational Research**, v. 189, n. 3, p. 1409–1426, 2008. ISSN 03772217.

RÍOS-MERCADO, R. Z.; ESCALANTE, H. J. GRASP with path relinking for commercial districting. **Expert Systems with Applications**, Elsevier Ltd, v. 44, n. September 2015, p. 102–113, 2016. ISSN 09574174.

RÍOS-MERCADO, R. Z.; FERNÁNDEZ, E. A reactive GRASP for a commercial territory design problem with multiple balancing requirements. **Computers and Operations Research**, v. 36, n. 3, p. 755–776, 2009. ISSN 03050548.

SALAZAR-AGUILAR, M. A.; RÍOS-MERCADO, R. Z.; CABRERA-RÍOS, M. New Models for Commercial Territory Design. **Networks and Spatial Economics**, v. 11, n. 3, p. 487–507, 2011. ISSN 1566113X.

SALAZAR-AGUILAR, M. A.; RÍOS-MERCADO, R. Z.; GONZÁLEZ-VELARDE, J. L. GRASP strategies for a bi-objective commercial territory design problem. **Journal of Heuristics**, v. 19, n. 2, p. 179–200, 2013. ISSN 15729397.

STEINER, M. T. A. et al. Multi-objective optimization in partitioning the healthcare system of Parana state in Brazil. **Omega (United Kingdom)**, Elsevier, v. 52, p. 53–64, 2015. ISSN 03050483.