# A genetic algorithm for fair land allocation

Alex Gliesch, Marcus Ritt
Institute of Informatics, Federal University of Rio Grande do Sul
Porto Alegre, Rio Grande do Sul, Brazil
azgliesch@inf.ufrgs.br,marcus.ritt@inf.ufrgs.br

Mayron C. O. Moreira
Department of Computer Science, Federal University of Lavras
Lavras, Minas Gerais, Brazil
mayroncesar@gmail.com

## ABSTRACT

The goal of the PROTERRA project of the Brazilian National Institute of Colonization and Agrarian reform (INCRA) is to establish settlements of multiple families on land which formerly had a single owner. The main problem which arises in the project is to subdivide the land into lots to be designated to the families. This problem is difficult since several constraints stemming from legal or ethical considerations have to be considered. Among the constraints are respecting natural reserves, balancing access to rivers with the size of the lots, and a fair distribution with respect to soil quality.

This problem has been mainly solved manually until now. In this paper we propose a genetic algorithm to solve it. We present several algorithmic components including a constructive heuristic and recombination and mutation operators that take into account the specifics of the problem, propose a technique to generate artificial instances for testing, and report on experiments with five real-world and 25 artificial instances.

## CCS CONCEPTS

**•Theory of computation → Optimization with randomized search heuristics; •Computing methodologies → Genetic algorithms; •Applied computing → Computers in other domains;**

## KEYWORDS

Agrarian reform, Land allocation, Genetic algorithm

**ACM Reference format:**
Alex Gliesch, Marcus Ritt and Mayron C. O. Moreira. 2017. A genetic algorithm for fair land allocation. In *Proceedings of GECCO '17, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3071178.3071313

## 1 INTRODUCTION

The *territorial organization in agrarian reform projects and environmental planning problem* (PROTERRA, acronym of the name in portuguese: *problema de organização territorial em projetos de reforma agrária e planejamento ambiental*) aims at the fair allocation of land to families, taking into account the land aptitude and technical constraints. Natural reserves are not allocated during the delimitation of the lots, in order to preserve areas with environmental importance and to avoid undue interventions that can affect the feasibility of settlement.

Studies concerning efficient procedures to land parceling emerge from the district allocation problem, which aims at partitioning territories into districts, respecting constraints such as connectivity, equality, and compactness. The design of political constituencies [3, 27, 28] and planning of sales or delivery regions [6, 14, 31] are two examples of contexts where we can find similar problems. A related problem concerns farmland consolidation, whose objective is to cluster a scattered number of lots and assign them to farmers in order to reduce cultivation costs. [2, 4] propose mathematical models, clustering algorithms and a computational system applied in Bavaria (Germany). [5] develop a local search algorithm for three provinces of Vietnam, as a first optimization technique ever approached in such areas.

Other studies address the multi-land use allocation problem (MLUA), where given a set of different spatial units, the goal is to assign the land parcels to pre-defined activities (e.g. agricultural, commercial, industrial, residential, park) considering their suitability for each purpose. Constraints like compactness, connectivity, allowed shapes or local legislation, are the most common. [1, 9] are the two most prominent studies considering mathematical models for MLUA. Even though they are effective for small problems, these formulations face difficulties to find solutions for instances with dimensions larger than $50 \times 50$ spatial units. Due to this fact, the main procedures to solve MLUA are metaheuristics such as simulated annealing [29], ant colony optimization [19], particle swarm optimization [21], genetic algorithms [7, 20, 30] and hybrid algorithms [22].

[18] propose an adaptation of a genetic algorithm taking as objective functions maximum economic benefit, maximum ecological benefit, maximum suitability, and maximum compactness. The authors also impose residential space demand and regulatory knowledge as constraints. Computational experiments conducted in the regional district of Central Okanagan (Canada) showed that the solutions obtained with this approach were able to satisfy both economic and environmental factors.

PROTERRA is related to the Agrarian Reform process, one of the main elements for the sustainable development of


*GECCO '17, Berlin, Germany*
 978-1-4503-4920-8/17/07...$15.00
DOI: http://dx.doi.org/10.1145/3071178.3071313

countries [8, 10, 26]. Results coming from successful implementation of agricultural settlements have a direct impact on good land use, food production, job creation and eradication of hunger. In Brazil, although the National Institute of Colonization and Agrarian Reform (INCRA) establishes technical rules to design agrarian settlements, the process of defining the lots is still mainly manual, which leads to regular territories with few angles and homogeneous areas, even though relief and hydrographic characteristics are heterogeneous.

In this study, inspired by the results found in literature of land allocation problems, we propose a constructive heuristic and a genetic algorithm (GA) for PROTERRA to minimize the standard deviation of land aptitude of parceled lots. We consider spatial objectives such as connectivity, compactness and shape, besides some desirable characteristics related to lot-sizing. We adopt a grid data input instead of vector format due to its increased flexibility [18], and propose new data structures and fast procedures to deal with large numbers of spatial units.

### 1.1 Previous work

[12, 13] are the first works to consider land allocation in the context of agrarian reform projects. These authors propose a genetic algorithm for a version of PROTERRA which does not deal with lot-sizing constraints. The algorithm was tested in a settlement in the state of Minas Gerais (Brazil) and produced better results than parceling performed by INCRA. On the other hand, [23] show that the offspring generated by crossover operator implemented for the GA produced solutions that did not satisfy the connectivity constraint, which must be reconstructed by a costly repair procedure. Then, they propose a tabu search for PROTERRA, regarding minimum and maximum sizes of lots. According to that study, the tabu search leads more frequently to feasible solutions. In all previous works, a grid-based structure was adopted to represent a solution of PROTERRA. However, the authors find that this data structure leads to inefficiencies in computing the objective function and when generating the neighbors in local search procedures.

The remainder of this paper is organized as follows. Section 2 presents the formal definition of the problem to be solved in the PROTERRA project. Section 3 presents the detailed description of the proposed constructive heuristic and the genetic algorithm, besides the new data structures and the fast procedure to update objective function value. Experimental results with real and artificial instances are presented in Section 4. Section 5 concludes this paper.

## 2 FORMAL PROBLEM DEFINITION

Let $k$ be the number of lots to allocate, $U = [n] \times [m]$ be the spatial units available, $R \subseteq U$ the units representing river, and $P \subseteq U$ the units in a natural reserve. For every spatial unit $u \in L$, where $L = U \setminus (R \cup P)$, which represents the usable land, we are given a productivity index $q_c$. Given a spatial unit $u = (x, y) \in L$, we define its neighbours $N(u)$ as the spatial units at positions $(x+1, y)$, $(x, y+1)$, $(x-1, y)$, $(x, y-1)$, whenever they exist. We extend the neighborhood to lots in the obvious way by $N(C) = (\cup_{u \in C} N(u)) \backslash C$. A unit $u$ or a lot $C$ is next to a river if any unit in is neighborhood is in $R$. A unit $u$ is said to be on the border between lot $i$ to $j$ if $u \in C_i$ and at any of its neighbors is in $C_j$. We define $B_{ij}$ as the spatial units on the border between lot $i$ to $j$. The total border of lot $i$ then is $B_i = \bigcup_{j \in [k]} B_{ij}$.

A solution is a surjective mapping $S : L \to [k]$ that assigns a lot to each spatial unit in $L$. Such a mapping defines a $k$-partition $L = \bigcup_{i \in [k]} C_i$ of the usable land, where $C_i = \{u \in L \mid S(u) = i\}$. Additionally, a solution has to satisfy four constraints.

The *connectivity* constraint requires the lots to be connected. In particular no lot can be divided by a river or a natural reserve. The *accessibility* constraint forbids enclaves, i.e. no lot is allowed to be completely enclosed within another lot. For a fair land distribution it also desirable that lots have access to a river have a smaller area than lots which do not. This is justified since they are already favored by direct access to the water resource. This leads to the *balance* constraint

$$\forall C_i : N(C_i) \cap R \neq \emptyset, \forall C_j : N(C_j) \cap R = \emptyset, |C_i| \leq |C_j|.$$

Finally, it is desirable that the lots have about the same area. This is handled by the *equality* constraint. Equality is measured by the ratio $\lambda$ of the area of the largest lot to the area of the smallest lot. A feasible solution must satisfy $\lambda \leq \bar{\lambda} = 3$.

Given a solution, the total quality value of lot $i$ is $v_i = \sum_{u \in C_i} q_u$. The objective function is to minimize the standard deviation $\sigma$ of the values $v_i$ where

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

and $\bar{v}$ is the average $v_i$. To compare two candidate solutions, we can omit the square root and the divison 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. The algorithms presented below satisfy connectivity and accessibility by construction, but not necessarily the balance and equality constraints. For this reason we measure the violation of the balance constraint by the total area excess $A$ of the lots with access to a river which exceed the area of the smallest lot without access, normalized by the total area of the instance, and of the equality constraint by $\lambda^+ = \max(\lambda - \bar{\lambda}, 0)$. We then consider the objective 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 quality.

A typical instance and its solution can be seen in Figure 1.

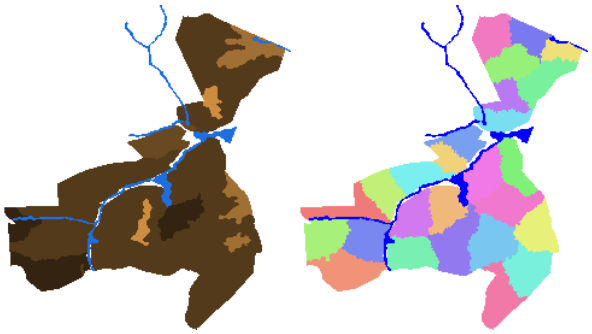

**Figure 1: Real-world instance "Veredas". Left: lighter shades of brown represent higher quality soil, blue rivers, and white natural reserves. Right: An example solution for this instance. Each lot is shown in a different color.**

# 3 A GENETIC ALGORITHM FOR LAND ALLOCATION

A genetic algorithm evolves an initial population of solutions by applying recombination and mutation operators to individuals chosen from the population according to their fitness, and a criterion which selects the individuals which will compose the next generation of the population. For a thorough, detailed introduction to genetic algorithms we refer the reader to the excellent literature available (e.g. [15–17]). The following subsections explain each of the main components of a genetic algorithm in detail.

## 3.1 Solution representation

The instances are defined on a regular, rectangular grid of cells. Any assignment of the spatial units to the $k$ lots that satisfies the criteria explained in Section 2 is a valid solution. We therefore represent a solution as an integer vector of length $n \times m$. This representation is inefficient for operations such as construction and local search, since they only need to modify cells on the border of lots. To speed up such operations we use a specific data structure that allows us to store, access, update and iterate over the border cells in a fast way. This is always possible, but particularly simple in our case, since we require the regions to be connected.

To this end, for each lot $i$, we keep a doubly-linked, cyclic list $b_i = (b_{i1}, \ldots, b_{i|B_i|}, b_{i1})$ of the spatial units on its border. Additionally, we keep a mapping $M$ that maps a unit $u \in L$ to the corresponding node in $b_{S(c)}$, or is undefined if $u \notin B_i$. This data structure permits 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.
- $insert(i, u)$: to add $u$ to $B_i$, we add it to the list $b_i$ and update $M(u)$ with the added node.
- $remove(i, u)$: to remove a unit $u$ from $B_i$, we look up $u$'s node in $i$ via $M(u)$ and remove it from $b_i$. Note that a removal can be done in constant time if we have a pointer to the desired node.
- $next(i, u)$: given $u \in B_i$, we can get the next element in $B_i$ by accessing the "next" pointer on the node $M(u)$. Note that applying $next$ successively on a given unit $u$ will cycle through all units in $B_i$ and return to $u$.

## 3.2 Updating objective function values

We consider a modification on the value of a single lot $v_j$ to $v'_j = v_j + \delta$, where $\delta$ can be positive or negative, depending whether lot $j$ lost or gained a cell. To speed up the calculation of the objective function under small changes, we can expand 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. Clearly, when the number of spatial units that change is of order $k$ or more, it is faster to first update all $v_i$, and then compute the new sum of squares $SS'$.

## 3.3 Initial population

We use a constructive heuristic to seed the initial population with good solutions. For each individual, the heuristic selects in a first phase $k$ spatial units as seeds for the $k$ lots and then in a second phase grows the full lots out of them, until all reachable free units have been assigned to some lot.

In the first phase, the $k$ seed units are generated in two steps. In the first step we select $k$ random spatial units. Since a river or a natural reserve may divide the usable land units $L$ into several connected components, and since the connectivity constraint requires every lot to be contained in a single component, we select in each component a number of random spatial units which is proportional to its area. Note that if a component has less than $|L|/k$ cells it will not receive a seed. The motivation of this choice is to minimize the differences in average area of the components, which helps satisfying the equality and balance constraints.

In the second step we partition the area into regions using an algorithm that is inspired by Voronoi diagrams. Given a set of $k$ seed points, a Voronoi diagram partitions the space into $k$ convex regions that contain the points that are closest to their corresponding seed. Since we are interested in minimizing the standard deviation of soil quality, we modify the distance criterion to a soil quality criterion. To this end we construct a directed graph whose vertices are the land units $L$, and whose arcs correspond to the 4-neighborhood $N$ defined above. The length of arc $(c, c')$, for $c \in L$, $c' \in N(c)$ is the target soil quality $q_{c'}$.

We then compute the Voronoi regions corresponding to the $k$ random seeds on this graph. This can be achieved by a shortest distance graph search. The search maintains a

set of active spatial units, which initially are the $k$ random seeds. It then repeatedly selects the active spatial unit of smallest distance to one of the initial seeds and assigns it to the corresponding lot. The neighbors of the selected active spatial unit then are made active or have their distance updated if they were active already. Using a priority queue to maintain the active spatial units the algorithm takes at most $O(|L| \log |L|)$ steps. The areas generated by this approach will be connected, and, for a reasonably smooth soil quality distribution, also be convex and without enclaves. Finally, for each lot $i$, we recompute its seed as the centroid of all units assigned to it weighted by their soil quality. If a centroid falls on a river or a natural reserve, we move it to the closest reachable spatial unit.

After generating the $k$ seeds, a greedy constructive algorithm constructs the final lots in the second phase. We expand lots starting from the initial seeds by repeatedly selecting the best feasible candidate among all free spatial units with respect to objective function $\varphi$ that border a lot, until no candidates left. A candidate is considered feasible when the connectivity and accessibility constraints are satisfied. When comparing two candidates lexicographically by $\varphi$, we use lazy evaluation to avoid computing parts of $\varphi$ that do not change the value of the comparison.

In each step the constructive algorithm needs to update the contribution of the spatial units to the objective function $\varphi$. This is the main bottleneck of the algorithm, since there can be a large number of spatial units. On the other hand the objective function values change very little from one step to the next. For this reason we define a batch size $b$, and assign in each the best $b$ candidates to the lots. As shown below larger batch sizes make the algorithm significantly faster with a small loss in solution quality.

### 3.4 Recombination and mutation operators

A recombination operator usually is designed to preserve desirable solutions traits of the parent solutions in the child. Our recombination operator implements this principle by trying to maintain the common structure of the assignment of cells to lots in both parent solutions. It is based on the observation the lots are *anonymous*, i.e. any lot in one of the two parent solutions could correspond to any lot in the other parent solution. Therefore, to maintain the common structure as much as possible, we construct a complete weighted bipartite graph where each vertex represents a lot and the two parts of the graph represent the two solutions. The weight of an edge between two lots is the number of cells in their intersection. We compute a maximum weight perfect matching to match the lots of the two solutions. We then allocate in the recombined solution the cells in the intersection in each matched pair of lots to the same lot. A special, rare case arises if there is some pair of lots with no intersection: in this case we have to complete the solution by introducing another seed for each such lot. Finally, the complete solution is reconstructed using the constructive algorithm seeded with the current cells and the remaining free space, as explained above. This process is shown in Figure 2.

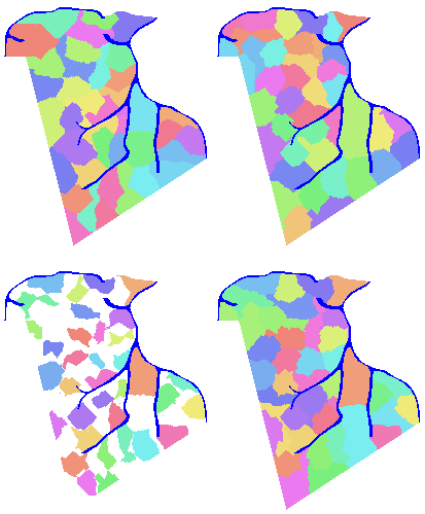

**Figure 2: The recombination operator applied to instance "Fortaleza". The top two images top show the parents. Bottom-left: the child solution after matching lots by area of intersection. Bottom-right: the child solution after reassigning the free space using the constructive algorithm.**

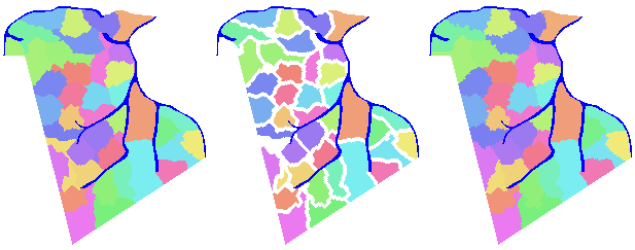

**Figure 3: The mutation operator applied to instance "Fortaleza". Left: the original solution. Middle: after removing units near the borders between lots. Right: greedily reconstructed solution.**

The mutation operator applies a small perturbation to the borders of the current lots of a solutions. To this end, we remove all spatial units within a given distance from any border between two lots. We have fixed this distance to a 2 units. Then, as for the recombination operator, we seed the constructive algorithm with these "core lots", and reconstruct the borders greedily. In order to introduce variability in repeated mutations of the same solution, we introduce a parameter $\alpha$ in our greedy construction: instead of selecting the $b$ best candidate units at every step, we select $b$ units with uniform probability among the $\alpha b$ best candidates. This can done with a selection algorithm in time $O(n)$, for $n$ candidates. Figure 3 illustrates the mutation process.

### 3.5 Selection strategies

The selection scheme is a 3-tournament for both parents of the recombination. The current population selection strategy is generational. For a population of size $P$, the next generation is composed of the best $e\,P$ individuals of the current generation, $o\,P$ individuals generated by recombination and mutation, and $r\,P$ random individuals. The elite rate $e$, offspring rate $g$, and random rate $r$ must satisfy $e+g+r=1$.

## 4 COMPUTATIONAL EXPERIMENTS

### 4.1 Test instances

We have tested our algorithm on five real-world instances, some of which have been used in the literature before [11]. For a more thorough testing of the algorithm we have generated another 25 artificial instances resulting from the combination of two experimental factors with five levels each: the instance size $n \times n$, for $n \in \{200, 300, 400, 500, 600\}$ and the number of lots $k \in \{20, 40, 60, 80, 100\}$. The levels have been chosen to match the typical range of the real-world instances.

Besides its size and topology, an instance is characterized by its soil quality. The soil quality usually is not continuous, but each spatial unit is classified according to a fixed number of aptitude classes. The number of aptitude classes is fixed in a given instance, and is normally between 5 and 10. The aptitude values for each class are within the interval $[30, 100]$. These aptitude classes appear in contiguous regions of the input matrix, and have smooth transitions between them. Additionally, in the artificial instances, the spatial units within a fixed distance of a river always belong to the highest aptitude class. We have fixed this distance to 5 spatial units. According to [11], the aptitude values of the five real world instances were obtained by an analysis of physical, agronomic and social factors of such settlements.

Table 1 summarizes the main characteristics of the test instances: their width ("w") and height ("h"), the number of aptitude classes ("Apt."), the percentage spatial units representing land, river, natural reverses, and the number of connected components ("#CC"). The instances will be made available online. The first five entries in the table correspond to the artificial instances and present averages of the five different numbers of lots, the last five entries are the real-world instances "Belo Vale" (Minas Gerais, Brazil), "Fortaleza" (Tocantins, Brazil), "Iucatã" (Acre, Brazil), "Olhos D'Água" (Minas Gerais, Brazil) and "Veredas" (Minas Gerais, Brazil).

To generate the artificial instances we use Perlin noise, a well-known 2D noise function that maps a coordinate to a real number in $[-1, 1]$, which is commonly used to randomly generate height maps [25]. Perlin noise has four main parameters: frequency $f$, octaves $o$, granularity $r$ and gain $g$. In preliminary tests they have been fixed to $f = 10^{-3}$, $o = 16$, $r = 2$ and $g = 0.5$. To generate the soil, we produce an $w \times h$ noise matrix and quantize it according to the desired number of aptitude classes. Perlin noise ensures more or less contiguous aptitude regions with smooth transitions.

**Table 1: Characteristics of the test instances.**

| w | h | Apt. | Land (%) | River (%) | Res. (%) | #CC |
|---|---|---|---|---|---|---|
| 200 | 200 | 5 | 94.4 | 5.6 | 0.0 | 7 |
| 300 | 300 | 6 | 90.9 | 9.1 | 0.0 | 4 |
| 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 |
| 300 | 300 | 5 | 41.6 | 1.0 | 57.4 | 1 |
| 449 | 250 | 2 | 20.7 | 1.2 | 78.1 | 1 |
| 449 | 250 | 5 | 26.1 | 2.5 | 71.4 | 12 |
| 300 | 300 | 4 | 38.2 | 15.8 | 46.0 | 9 |
| 300 | 300 | 6 | 42.0 | 1.5 | 56.5 | 10 |

In order to generate the rivers, we use another Perlin matrix with $o = 10$, quantize it into two levels, seed the river with the spatial units on the border between the two levels, and expand the seed to a fixed river width (in the experiments we have used a random dilation in $[2, 5]$ corresponding to half of the river width). If the total number of river cells deviates from the desired percentage, we adjust the frequency $f$ of the Perlin noise, which defines how frequently changes in noise value will occur, by a binary search and repeat the process until the deviation is minimal. In each instance, we fix the total number of river cells randomly to be between 2% to 5% of the total number of spatial units. Finally, we replace all connected components of land with an area smaller than $|L|/2k$ by water, to prevent island-like structures that are too small for a lot to be placed, and would not normally appear in real-life scenarios.

### 4.2 Methodology

We have implemented our algorithms in C++, compiled them with GCC 5.3.1 and maximum optimization, and tested them on a PC with an 8-core AMD FX-8150 processor and 32 GB of main memory. For each test only one core has been used. The parameters of the genetic algorithm have been calibrated using the irace package in GNU R, with a budget of 1000 runs and a time limit of 5 minutes per run. The iterative racing algorithm searches the parameter space for a good configuration, by sampling configurations according to a model that converges to the best performing configurations, and repeatedly eliminates poorly performing solutions after a Friedman test finds significant performance differences among the population of tested configurations [24]. The parameters, their initial ranges, and the optimal parameter setting found are shown in Table 2.

Below we report on four experiments. The first determines the best batch size for the constructive algorithm, the second studies the scalability of the GA, the third evaluates its effectiveness compared to simpler methods, and the last compares its results to known solutions of real-world instances.

### 4.3 Experiment 1: Optimal batch size

In our first experiment we calibrate the optimal batch size of the constructive algorithm. We have used batch sizes

**Table 2: Parameters of the genetic algorithm: initial ranges and optimal setting found by iterative racing.**

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

**Table 3: Results of the calibration of the optimal batch size.**

| Batch | Fixed time | | | | 50 replications | | | |
|---|---|---|---|---|---|---|---|---|
| size | $A$ $(m)$ | $\lambda$ | $\sigma$ | Repl. | $A$ $(m)$ | $\lambda$ | $\sigma$ | 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 |

$2^b$, for $b \in [5, 12]$ and the 25 artificial instances. We have done two experiments. In the first experiment we performed a fixed number of 50 replications to analyze the trade-off of computation time versus solution quality. The second experiment runs the maximum number of replications for a fixed time limit of 10 min. Table 3 shows the results. For both tests we report the balance violation $A$ ($\times 10^{-3}$), the equality ratio $\lambda$ and the soil quality deviation $\sigma$. For the experiment with a fixed time we report the number of replications, for the experiment with a fixed number of replications the total time in seconds. Values reported for $\sigma$ are relative deviations from the best known value.

Looking at a fixed number of replications, we see that the solution quality in terms of balance $A$ degrades with increasing batch sizes, but remains with 1.19% of the total area still low. Since the balance is not optimal the equality constraint is not satisfied, and the soil quality deviation varies between 30% to 80%. The running time is almost inversely proportional to the batch size. For a fixed time this trade-off leads to an optimal batch size of 256 with best balance $A$ of $23.3 \times 10^{-3}$ and similar values for $\lambda$ and $\sigma$. Thus, for the remaining experiments we have fixed the batch size at this value.

## 4.4 Experiment 2: Scalability

In this experiment we evaluate the performance of the GA for different instances sizes and different number of lots. We have run the GA on the 25 artificial instances for 30 min. Each run has been replicated 5 times. Table 4 shows the results. As above we report the average balance violation ($\times 10^{-6}$), the equality ratio $\lambda$ and the quality deviation $\sigma$ (absolute values). Column "n" reports the number of feasible solutions found.

**Table 4: Scalability experiment.**

| Size | Lots | Evals. | $A$ $(\mu)$ | $\lambda$ | $\sigma$ (K) | $n$ |
|---|---|---|---|---|---|---|
| 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 |

We see that the GA finds feasible solutions in 112 of the 125 tests, being unable to achieve feasibility only for the instance of size $200 \times 200$ with 100 lots. This is due to the large number of lots within a small area, which makes it harder to satisfy the balance constraint. For the instance of size $300 \times 300$ the equality constraint was satisfied only once. In the remaining instances the majority of the solutions is feasible. As expected, with a few exceptions the soil quality deviation decreases with an increasing number of lots.

Looking at scalability of the GA, we find that the number of fitness function evaluations decreases as expected with the size of the instance and the number of lots. 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 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.

## 4.5 Experiment 3: Effectiveness of the GA

In this experiment, we verify the effectiveness of the GA by comparing it to two simpler strategies: one that repeatedly

generates random seeds and expands them by breadth-first-search (BFS), and the repeated application of the pure greedy constructive approach of Section 3.3. In this experiment we used the real-world instances. As before, the time limit was 30 min and we report averages over 5 replications. Table 5 shows the results. We again report the balance violation $A$ ($\times 10^3$), the equality ratio $\lambda$, and the quality deviation $\sigma$ (as the relative deviation in percent from the best known value).

The repeated BFS found better solutions than the constructive approach with regards to the balance constraint in 4 of the 5 instances. However, its solutions tend violate the equality constraint much more, and since there is no consideration of the soil quality, it leads to even higher soil quality deviations. Being algorithmically simple, the repeated BFS achieves more fitness evaluations per time. The repeated constructive heuristic leads to overall more balanced and better, but infeasible solutions. It also achieves the least number of fitness function evaluations, since it performs full solution constructions. The GA found feasible solutions for all five real-world instances, and with considerably lower average soil quality deviations. Its number of fitness function evaluations is higher than the repeated constructive heuristic, since it performs only partial constructions after the recombination and mutation operators.

In summary, these results show that the GA is more effective than simpler strategies. In particular, its performance cannot be explained only by the quality of the initial solutions, or the repeated addition of a percentage of random individuals to the population, and suggests that the proposed recombination and mutation operators are effective.

### 4.6 Experiment 4: Comparison to manual allocation

In our final experiment we compare the results of the GA algorithm in four of the five real-world instances to existing allocations that were constructed manually by the responsible government entity INCRA (Instance "Fortaleza" has been excluded since we did not have access to the official allocation). The results of the GA have been obtained with the same time limit of 30 min and again we report averages over 5 replications. Table 6 shows the results. The allocation done by INCRA does not consider rivers as obstacles to the connectivity constraint, and allows lots to have land on both sides of a river. For the purpose of this comparison, solutions that violate the connectivity constraint were not penalized.

We can see that the GA produces solutions that are better with respect to all three components of the objective function. The manual allocations proposed by INCRA violate in all instances either the balance and the equality constraint, and often both of them, whereas the GA was always able to find feasible solutions. The soil quality deviation found by the GA is about a factor of 1.7 to 7 lower. Figure 4 shows an example of a manual allocation and an allocation obtained by the GA for instance "Veredas".

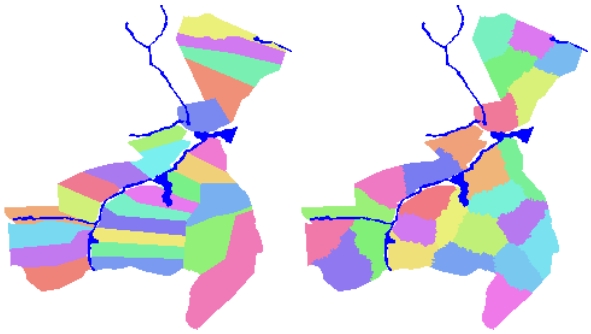

**Figure 4: Solutions of the real-world instance "Veredas". Left: the solution produced manually by INCRA. Right: the solution produced by our GA.**

## 5 CONCLUSIONS

This study addresses the *territorial organization in agrarian reform projects and environmental planning problem* (PROTERRA), an open problem, especially in developing countries. The objective is to find a fair distribution of land for families based on land aptitude, excluding natural reserves from parceling. We consider lot-sizing constraints based on balancing the area of lots with and without access to rivers and area equality constraints, as well as connectivity and accessibility of the lots. To solve the PROTERRA, we propose a constructive heuristic guided by modified Voronoi diagram and a genetic algorithm with problem-specific recombination and mutation operators.

We have evaluated the proposed methods on five real-world instances from Brazil and 25 artificial instances. The results show that the GA scales reasonably well with the size of the instances and the number of lots to be allocated. In comparison with simpler strategies we found that the components of the GA were effective, considering solution quality (standard deviation of lot aptitude) and constraints violation. The GA is fast enough to produce good solutions in a reasonably short time and its crossover and mutation operators showed to be a flexible to approach violation of connectivity restriction after their execution. Finally, comparing the land allocation of the GA to manual land allocation produced by INCRA, we find that the GA has the potential to lead to better solutions than those currently applied in practice.

## REFERENCES

[1] Jeroen C. J. H. Aerts, Erwin Eisinger, Gerard B. M. Heuvelink, and Theodor J. Stewart. 2003. Using Linear Integer Programming for Multi-Site Land-Use Allocation. *Geographical Analysis* 35, 2 (2003), 148–169.

[2] Steffen Borgwardt, Andreas Brieden, and Peter Gritzmann. 2014. Geometric clustering for the consolidation of farmland and woodland. *The Mathematical Intelligencer* 36, 2 (2014), 37–44.

[3] Burcin Bozkaya, Erhan Erkut, and Gilbert Laporte. 2003. A tabu search heuristic and adaptive memory procedure for political districting. *Eur. J. Oper. Res.* 144, 1 (2003), 12–26.

[4] Andreas Brieden and Peter Gritzmann. 2004. *A Quadratic Optimization Model for the Consolidation of Farmland by Means of Lend-Lease Agreements.* Springer Berlin Heidelberg, Berlin, Heidelberg, 324–331.

**Table 5: Effectiveness experiment.**

| Instance | Fitness evals. ($\times 10^4$) | | | $A$ (m) | | | $\lambda$ | | | $\sigma$ (% 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 |

**Table 6: Comparison of our genetic algorithm to manual allocation made by INCRA.**

| Instance | $A$ (m) | | $\lambda$ | | $\sigma$ | |
|---|---|---|---|---|---|---|
| | 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 |


[5] Quoc Trung Bui, Quang Dung Pham, and Yves Deville. 2013. Solving the agricultural land allocation problem by constraint-based local search. In *International Conference on Principles and Practice of Constraint Programming*. Springer, 749–757.

[6] A. Butsch, J. Kalcsics, and G. Laporte. 2014. Districting for Arc Routing. *INFORMS Journal on Computing* 26, 4 (2014), 809–824.

[7] Kai Cao, Bo Huang, Shaowen Wang, and Hui Lin. 2012. Sustainable land use optimization using Boundary-based Fast Genetic Algorithm. *Computers, Environment and Urban Systems* 36, 3 (2012), 257–269.

[8] Elizabeth Alice Clements. 2013. Agrarian reform, food sovereignity and the MST: socio-environmental impacts of Agrofuels production in the Pontal do Paranapanema Region of the state of São Paulo, Brazil (Reforma agrária, soberania alimentar e o MST: impactos socioambientais da produção de...). *Revista Nera* 21 (2013), 8–32.

[9] K.D. Cocks and I.A. Baird. 1989. Using mathematical programming to address the multiple reserve selection problem: An example from the Eyre Peninsula, South Australia. *Biological Conservation* 49, 2 (1989), 113–130.

[10] Roseni Aparecida de Moura, José Ambrósio Ferreira Neto, Sheila Maria Doula, and João Luiz Lani. 2011. Reforma Agrária e Desenvolvimento: A Reconstrução e uma Questão Polêmica. *ReBraM* 14, 2 (2011), 95–106.

[11] F. M. Ferreira. 2015. *Aptidão agrícola das terras como função de otimização para o ordenamento territorial e planejamento ambiental: uma anãlise do SOTER-PA*. Master's thesis. Programa de Pós-Graduação em Extensão Rural, Universidade Federal de Viçosa, Brazil.

[12] José Ambrósio Ferreira Neto, Edgard Carneiro dos Santos Junior, Urbano Fra Paleo, and Mayron César de Oliveira Moreira. 2011. Optimal subdivision of land in agrarian reform projects: an analysis using genetic algorithms. *Ciencia e Investigación Agraria* 38 (2011), 169–178.

[13] Jose Ambrósio Ferreira Neto, Mayron César de Oliveira Moreira, Edgard Carneiro dos Santos Junior, Urbano Fra Paleo, and João Luiz Lani. 2011. Aptidão agrícola e algoritmos genéticos na organização espacial em projetos de reforma agrária. *Revista Brasileira de Ciência do Solo* 35 (2011), 255–261.

[14] Gabriela García-Ayala, José Luis González-Velarde, Roger Z Ríos-Mercado, and Elena Fernández. 2016. A novel model for arc territory design: promoting Eulerian districts. *International Transactions in Operational Research* 23, 3 (2016), 433–458.

[15] Michel Gendreau and Jean-Yves Potvin (Eds.). 2010. *Handbook of Metaheuristics* (2nd ed.). Springer.

[16] David E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

[17] John H. Holland. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

[18] Xin Li and Lael Parrott. 2016. An improved Genetic Algorithm for spatial optimization of multi-objective and multi-site land use allocation. *Computers, Environment, and Urban Systems* (2016).

[19] Y.L. Liu, D.W. Tang, X.S. Kong, Y.F. Liu, and T.H. Ai. 2014. A land-use spatial allocation model based on modified ant colony optimization. *International Journal of Environmental Research* 8, 4 (2014), 1115–1126.

[20] Yaolin Liu, Man Yuan, Jianhua He, and Yanfang Liu. 2015. Regional land-use allocation with a spatially explicit genetic algorithm. *Landscape and Ecological Engineering* 11, 1 (2015), 209–219.

[21] Zohreh Masoomi, Mohammad Sadi Mesgari, and Majid Hamrah. 2013. Allocation of urban land uses by Multi-Objective Particle Swarm Optimization algorithm. *International Journal of Geographical Information Science* 27, 3 (2013), 542–566.

[22] Mahmoud Mohammadi, Mahin Nastaran, and Alireza Sahebgharani. 2016. Development, application, and comparison of hybrid meta-heuristics for urban land-use allocation optimization: Tabu search, genetic, GRASP, and simulated annealing algorithms. *Computers, Environment and Urban Systems* 60 (2016), 23–36.

[23] M. C. O. Moreira, J. A. Ferreira Neto, C. J. Einloft, and N. T. C. Silva. 2011. O uso da busca tabu no ordenamento territorial em assentamentos rurais: reconfigurando o SOTER-PA (Sistema de Ordenamento Territorial da Reforma Agrária e Planejamento Ambiental). In *Desenvolvimento Rural, sustentabilidade e ordenamento territorial* (1 ed.), José Ambrósio Ferreira Neto Carlos Joaquim Einloft and Renato Luiz Gonçalves (Eds.). Suprema, Visconde do rio Branco, 265–272.

[24] Manuel López-Ibá nez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari. 2016. The irace package: Iterated Racing for Automatic Algorithm Configuration. *Operations Research Perspectives* 3 (2016), 43–58.

[25] Ken Perlin. 1985. An Image Synthesizer. In *SIGGRAPH Comput. Graph.* 287–296. DOI:http://dx.doi.org/10.1145/325165.325247

[26] Rossana Rocha Reis. 2012. 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* 86 (2012), 89–122.

[27] Federica Ricca, Andrea Scozzari, and Bruno Simeone. 2008. Weighted Voronoi region algorithms for political districting. *Mathematical and Computer Modelling* 48, 9 (2008), 1468–1477.

[28] Federica Ricca, Andrea Scozzari, and Bruno Simeone. 2013. Political districting: from classical models to recent approaches. *Annals of Operations Research* 204, 1 (2013), 271–299.

[29] Inés Santé-Riveira, Marcos Boullón-Magán, Rafael Crecente-Maseda, and David Miranda-Barrós. 2008. Algorithm based on simulated annealing for land-use allocation. *Computers & Geosciences* 34, 3 (2008), 259–268.

[30] Theodor J. Stewart and Ron Janssen. 2014. A multiobjective GIS-based land use planning algorithm. *Computers, Environment and Urban Systems* 46 (2014), 25–34.

[31] Andris A Zoltners and Prabhakant Sinha. 2005. The 2004 ISMS Practice Prize Winner–Sales territory design: Thirty years of modeling and implementation. *Marketing Science* 24, 3 (2005), 313–331.