Optimizing parcel exchange among landowners: a soft alternative to land consolidation

Abstract

For decades, public policy has favored the use of land consolidation to reduce the fragmentation of land ownership. Private actors, on the other hand, have focused on the purchase, rental and exchange of land plots. Plot exchange can be very useful in the restructuring of holdings, particularly when a large number of owners participate; however, the number of possible exchange combinations grows very quickly with the number of participating landowners and parcels. Finding an acceptable exchange solution can easily become challenging. In this paper we evaluate the practical use of a support system for land exchange processes. The system is based on the use of genetic algorithms, a particular kind of heuristics that loosely replicate the rules of evolution and natural selection. We assess the influence of the geometric distribution of parcels in the quality of the solution, as well as usefulness and performance of the system, via parallelization techniques. The proposed algorithm (GA-PE, Genetic Algorithm for Parcel Exchange) is tested with regards to several parameters, from several alternatives for certain steps of the algorithm to the resource distribution for the parallelizations implemented. We tested the algorithm in 6 different real and representative test cases, and provide results with different metrics. With the positive results obtained, we argue that land exchange is a process worth considering for private actors, and that genetic algorithms can be used to propose fair exchanges, even in complex scenarios, shortening in a meaningful way the time usually required to perform administrative procedures associated to land fragmentation problems.

Keywords: Land ownership, fragmentation, land market, geographic information systems, land management, land administration, global optimization, genetic algorithms

1. Introduction

For decades, agronomists and land planners have proposed measures to reduce fragmentation of land ownership (Binns, 1950). Very high levels of

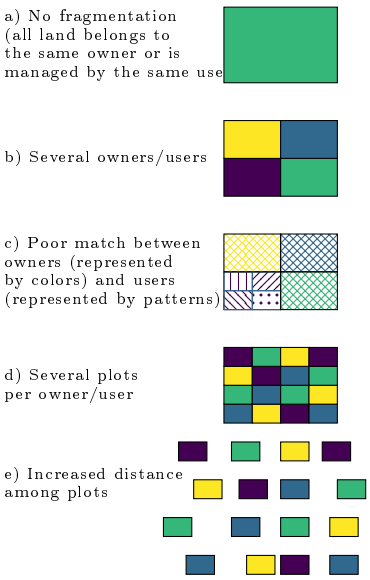


Figure 1: Schematic representation of different types of fragmentation, compared with a base situation (a). High number of owners or high number of users (b); lack of overlap between owners and users (c); high number of plots per owner/user (d); high distance between plots of the same owner/user (e).

fragmentation are considered to be an obstacle for the development of profitable agricultural (Orea et al., 2015; Lu et al., 2018) and forestry sectors (Rendenieks et al., 2015; Kilgore and Snyder, 2016). Therefore, instruments designed to reduce land fragmentation are principally —but not only— intended to improve the profitability of farms and forest holdings.

Following van Dijk (2003), four different types of fragmentation —not mutually exclusive— can be described: (1) a situation in which there is a high number of landowners, resulting in very small holdings; (2) a situation in which there is a high number of land users, resulting in very small farm/forest management units; (3) the lack of overlap between the groups of owners and users; and (4) a high ratio between the number of plots and the number of users, resulting in very small land plots. A fifth one can be added: the situation in which the parcels of a given holding are too far away from each other (Hartvigsen, 2014). Graphical representations of each case are shown on Figure 1. The concrete scenario in any given region is usually a combination of these five types, and the measures applied to reduce fragmentation will need to take this into account if they are to be successful.

Usually, private and public initiatives tend to focus on the fourth and fifth types described, therefore aiming to reduce the average number of land

plots per landowner and place them as close as possible. The public sector has often used land consolidation (LC), which can be summarily described as the process by which the plots of each owner are combined to form a smaller number of larger pieces of land (Pašakarnis et al., 2013; Lisec et al., 2014). When applied in large areas, LC can be very successful in reducing the average number of plots per owner but it does not necessarily improve farm productivity or income (Du et al., 2018; Djanibekov and Finger, 2018). On the other hand, as a completely new division of land plots is a common result (Lisec et al., 2014; Kupidura et al., 2014), environmentally valuable features of the previous landscape, like hedgerows and strips of semi-natural vegetation, tend to be reduced.

Private actors, on the other hand, often combine buying, renting, land banking and exchanging land plots in order to consolidate holdings (Vranken and Swinnen, 2006; Sklenicka et al., 2014). All of these agreements have the potential to allow the restructuring of holdings at a reduced environmental cost, as the landscape structure remains untouched. Additionally, plot exchange does not require owners to have financial resources available that would be needed to purchase new pieces of land. The potential benefits for landowners increase greatly when a large number of them are involved in an exchange agreement as the number of possible solutions grows very quickly. Unfortunately, in practical terms this makes it difficult to search for an optimal solution, even when relatively small numbers of landowners and land plots are considered.

In a similar way as to how software can help improve other alternatives to reduce land fragmentation (Touriño et al., 2003; Porta et al., 2013c), this work focuses on the practicality of an algorithmic decision support tool to help in parcel grouping tasks by means of voluntary plot exchange between multiple owners. The software developed aims to reduce time required for the principal task in this process: the reassignment of all the plots. The proposed tool focuses on providing, in short periods of time, firstly results that can be directly accepted by participants, or secondly used as a starting point for negotiation. In both cases, most of the work is performed by the software (all of the work in the former case). Also, in the second case when some landowners may not accept some particular assignment, adding small changes to the proposed solution to reach an agreement between the landowners is easier. Since it comes from an already satisfactory starting point (the solution proposed by the algorithm), the technicians can perform these changes in a reasonable amount of time and effort.

1.1. Genetic algorithms

The potential benefits of multiple parcel exchange are logically greater when a higher number of owners and parcels are involved in the same exchange process, compared to the exchange being done on a one-by-one parcel basis. The number of possible exchange combinations grows very quickly as both the number of landowners and parcels involved increase (Borgwardt et al., 2014). The time needed for trial-and-error performed by a computer, as well as combinatorial aproaches, grows exponentially with the number of parcels and owners, rendering them inviable for the cases that interest us. The use of heuristic algorithms is a common viable option in solving complex problems involving a large number of possible solutions. These algorithms reduce the time needed to achieve good solutions at the expense of not always finding the best possible solution (performance vs optimality), and could be easily considered a viable option in this case.

Generally, heuristic algorithms optimize a problem by iteratively attempting to improve a candidate solution with regard to a given measure of quality. They can be used to approach the solution of complex problems, such as those commonly found in land management or land administration, which would otherwise take a very long time to solve. Genetic algorithms (GA) are a particular kind of heuristics that mimic the principles of natural evolution: they produce an initial population of solutions which are evaluated using a fitness function (FF); the best individuals (i.e., those with a higher score in the FF) are then used to produce a new generation of possible solutions, and the cycle is repeated until stopping criteria are met.

Evolutionary optimization algorithms are well suited to support multiobjective spatial decision making (Bennett et al., 2004; Xiao et al., 2007). Published applications include land-use allocation (Porta et al., 2013b; Stewart and Janssen, 2014; Liu et al., 2015; Santé et al., 2016), automatic delimitation of population settlements (Porta et al., 2013a), open space planning (Xin and Zhi-xia, 2008; Vallejo et al., 2015) or even traditional land consolidation (Touriño et al., 2001).

Genetic algorithms have been proposed to support decisions concerning land reallotment in traditional land consolidation processes (Akkus et al., 2012; Demetriou et al., 2012; Uyan et al., 2015; Ertunç et al., 2018a). In all of these cases, GA are used to completely redesign the parcel structure. In this paper, however, GA are used to propose the most adequate parcel exchange for a given set of owners and, therefore, the pre-existing parcel structure remains unchanged.

Although other types of heuristic algorithms could probably suit the requirements of this work (Suárez et al., 2011), we have chosen a conventional GA, given that, as they have already been proven for similar problems, the

drawbacks thereof are not a concern (since the sub-optimal solutions provided can and are likely to be refined by the landowners before materializing). Moreover, as GA implies an evolution of possible solutions, these interim results can be stored for later analysis, or almost straigthforwardly even reused as a basis for exploring newer ones. Additionally, GAs allow for further improvements aimed at reducing executions times through parallelization techniques, as can be found on multiple ocassions in the literature. One of these improvememnts is the cooperation of multiple algorithm executions (Porta et al., 2013b), this being one of the parallelization techniques explored, as explained in detail in Section 3.

2. Genetic algorithm for parcel exchange(GA_PE)

This section introduces the terminology related to genetic algorithms, their meaning in the field of land consolidation (LC), and their application in this particular case. The goal of the algorithm is to find the *best landowner* to assign each parcel to, defining *best landowner* as the one that improves a metric the most.

Genetic algorithms (Goldberg, 1989) are search heuristics that are often applied to optimization or learning. They often use terms such as 'genes' and 'individuals', and operators such as 'selection', 'crossover' and 'mutation', similar terminology to that used in natural evolution.

In a GA, an *individual* encodes a candidate solution for the given problem: in this case, an individual is formed by the assignment of each parcel to one of the participating landowners, and we will use the term ownership pattern. Ownership patterns consist of as many *genes* as the number of parcels considered: each gene is simply a label that encodes the owner assigned to that parcel. In Figure 2, an ownership pattern is shown as an array of labels, each one indicating the owner of a specific parcel. A group of ownership patterns or possible solutions is called a *population* or *generation*. Ownership patterns resulting from the exchange of genes (crossover) and the mutation of some genes between ownership patterns of a population are called the offspring or children; together they form the next generation. Finally, a metric that represents the value of an ownership pattern according to some criteria is called a *fitness function* (FF).

A GA usually consists of four basic steps that are iterated for each new generation: selection of best ownership patterns from the population, crossover, mutation and creation of a new population. In our case, we also define three additional steps to allow more flexibility: ownership pattern generation, ownership pattern evaluation, and ownership pattern validation. The steps of generation, evaluation, validation and mutation are implemented

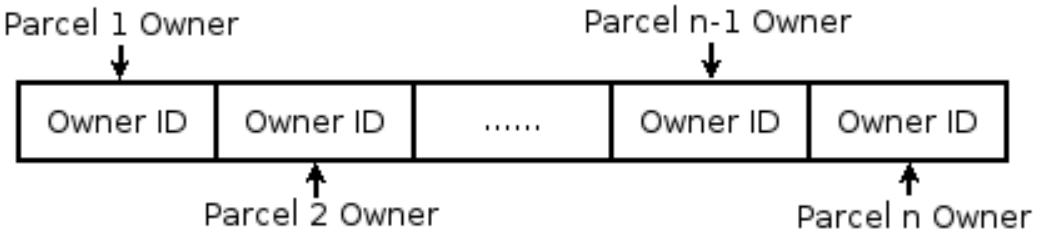


Figure 2: Individual or ownership pattern representation

in a modular way, while the other three steps use common techniques and operations. In each modular step, the code that performs the changes needed is implemented in an isolated function, and the exact function to be used can be configured between several options in each run of the algorithm. This flexibility also allows the easy implementation and integration of new functions (new alternative implementations) for each step of the algorithm, in case the need arises.

2.1. Selection of best ownership patterns

The selection of the best ownership pattern of each population is based on the fitness value calculated by the evaluation function. In this case, we select the ownership pattern with the lowest fitness value, since the system is designed as a minimization problem.

2.2. Creation of new populations

This step is somewhat implicit in the algorithm. The descendants of the current population are inserted into a list, and when all of the descendants are created, the list is considered as the new generation or population.

2.3. Crossover operation

The crossover step uses a basic technique, a two-point crossover, to reduce computational cost slightly. Two parents are selected randomly to create two different children. Two indexes are selected randomly, and the genes between those indexes are swapped in the children. A simplified diagram of the operation in a case with ten parcels and four different owners is shown in Figure 3. After that, the mutation function (see Section 2.4) is applied and the best child is selected as the descendant of the parents. To avoid regression, we apply elitism, taking into account the parents in the final selection; hence, if the children are worse than any of the parents, that parent is selected for the next generation.

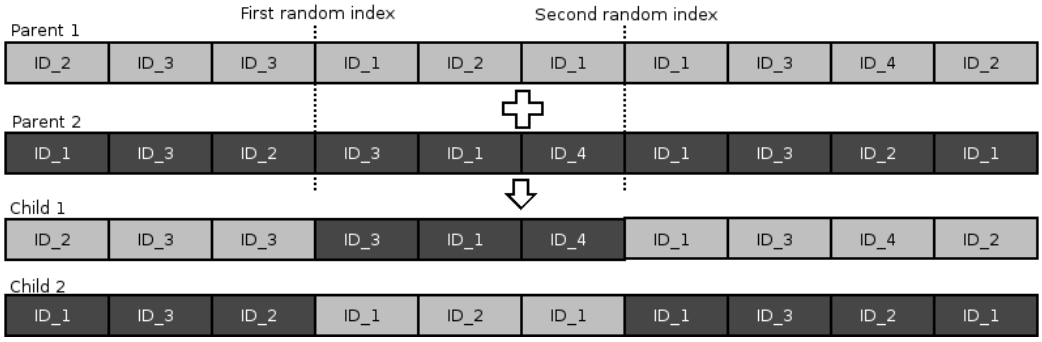


Figure 3: Crossover operation

2.4. Mutation function

The mutation method plays an important role in the algorithm, as it affects the width of the search and has a notable impact on the speed, since it can be applied multiple times to each ownership pattern.

We use a fixed mutation probability of 100%, due to the need to test many ownership patterns in order to be able to find those that are valid. After the crossover, the mutation method is applied to the two children, and if the resulting ownership patterns are not valid, the mutation method can be applied again. The mutation is executed multiple times until a valid ownership pattern is found or a maximum number of mutations has been performed on the same ownership pattern.

The algorithm currently supports two types of mutators: those based on random parcel assignment and those based on parcel exchange between two owners. We have developed several methods to see the effects in time to reach good solutions, seeking a good balance between complexity (and mutation time) and overall speed of the algorithm (or time to reach good solutions).

The mutation function based on random parcel exchange is called Random Change. It selects a random parcel and assigns it to a random owner (an option to repeat the process multiple times in each mutation is also available).

The second type of mutation functions are based on the exchange of parcels between two owners: the first, Swapping Mutator, simply, exchanges the owners of two parcels randomly selected, including the option to repeat the process in each mutation; the second, SwapNPlots, selects two random owners and exchanges one or more parcels between them (with options to repeat the process in each mutation). A diagram with an example of mutation using the Swapping Mutator is shown in Figure 4, for a case with ten parcels and two different owners.

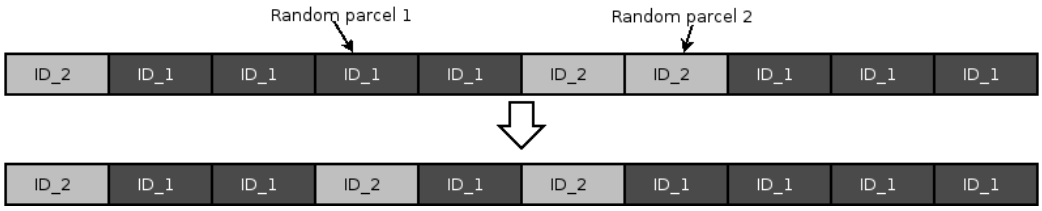


Figure 4: Swapping mutation diagram

The algorithm also supports the use of multiple mutation methods, applying all of them to the two children of a crossover, generating two new children for each mutation method in use, to broaden the search space at the cost of time needed to perform the mutation step. In Section 4.1 we test the three basic mutators mentioned above, seeking a combination that offers good results and speed.

2.5. Generation function

The algorithm supports different generation functions, which are used to generate the ownership patterns of the first population or generation, and to generate ownership patterns in case of stagnation.

In order to maintain diversity in the population, we have implemented a mechanism for stagnation detection. When all the ownership patterns of a population have the same fitness value, over multiple generations, we consider the population to be stagnated and the corresponding control procedures are triggered. We use this detection strategy as it is unlikely that two different ownership patterns will have the exact same fitness value, and is simple and rapid to check. The amount of generations with no changes and the same fitness value in all ownership patterns before activating stagnation control can be configured.

When stagnation is detected, the generation method is used to generate new ownership patterns, after which the best new ownership patterns replace some of the ownership patterns in the main population. The amount of ownership patterns generated and the ratio of the main population that is replaced can be configured.

We have developed several generation methods, but we will use one one of them as it has shown the best results during development. It uses the distances of each parcel to the reference point of each owner (usually indicating the location of the owner's farm, or the point where the owner prefers the parcels to be located), assigning the parcels to the closest reference point. To introduce diversity, the order the parcels are assigned is random, that way it can produce different ownership patterns each time a new element is needed.

2.6. Ownership pattern validation

To be of any use in the parcel exchange process, it was decided that the search for a better solution by the genetic algorithm should be constrained by some condition. This restriction is implemented as a function called ownership pattern *validator*, which determines whether an ownership pattern is a valid solution to the problem.

In its current implementation, there is one such condition, based on the value of the parcels. The total value assigned to each owner should not change more than a given percentage of the value prior to the exchange; for example, +-10% (the increase/decrease percentages can be changed independently, or even disabled entirely, if desired).

The default value of a parcel is its surface area, although the algorithm supports the provision of any other numeric values for all parcels. Using this numeric value allows other types of objectives (monetary or agronomic value) or even subjective values (value perceived by the owners, for example) to be considered.

2.7. Fitness function

In traditional LC processes, the euclidean distance from the centroid of each parcel and the center of gravity of all the parcels assigned to the same owner has been used as an indicator of success (Crecente et al., 2002). To provide flexibility and adapt to what the landowners value the most, we provide multiple metrics of success in parcel exchange (also called fitness functions in GA terminology) focused on different aspects.

All of these metrics can be summarized in the following groups: metrics based on the distance between parcels (with variants regarding aggregation operators and ponderations), metrics based on the distance to a reference point chosen by the owner (with variants regarding aggregation operators and ponderations), several combinations of the previous ones, and two miscellaneous ones explained in detail below. In the mathematical expressions of each one, LO is the number of owners present, n_o the number of parcels of owner o, $PD_{o,i,j}$ the distance between parcel i and j of the owner o, $RD_{o,i}$ the distance from parcel i of the owner o to the reference point of the owner o, and W_{TPD}, W_{APD}, W_{TRD} and W_{ARD} the weights given to the first, second, third and fourth evaluation methods, respectively.

The first group of fitness functions are based on the distance between parcels:

- TPD: Total Parcel Distance, which calculates the total of the distances between each pair of parcels of each owner. Equation 1.

- APD: Average Parcel Distance, which is the average distance between each pair of parcels of each owner. Similar to the first one, but using the average with regards to the number of parcels of each owner instead of the sum of all distances. Equation 2.

$$\frac{\sum_{o=1}^{o=LO}\left\{\sum_{i=1,j=i}^{i=n,j=n} PD_{o,i,j}\right\}}{LO} \tag{1}$$

$$\frac{\sum_{o=1}^{o=LO}\left\{\frac{\sum_{i=1,j=i}^{i=n,j=n} PD_{o,i,j}}{n_o}\right\}}{LO} \tag{2}$$

The ones on the second group are based on the distance to a reference point chosen by the owner:

- TRD: Total Reference point Distance, takes the total distances from each parcel to the reference point chosen by the owner. Equation 3.
- ARD: Average Reference point Distance, calculates the average distance from each parcel to the reference point of the owner. Similar to the first one, but using the average with regards to the number of parcels of each owner instead of the sum of all distances. Equation 4.

$$\frac{\sum_{o=1}^{o=LO}\left\{\sum_{i=1}^{i=n} RD_{o,i}\right\}}{LO} \tag{3}$$

$$\frac{\sum_{o=1}^{o=LO}\left\{\frac{\sum_{i=1}^{i=n} RD_{o,i}}{n_o}\right\}}{LO} \tag{4}$$

The third group comprises specific combinations of the previous fitness functions. These functions allow weights to be given for each one of the fitness functions they combine. For ease of use, if no weights are provided, they perform an automatic weight balancing. This weight balancing aims to give the same relevance to all the parts, taking into account the differences in magnitude of each fitness function (functions using the total values have less weight than the ones using the average values, as they have much higher absolute values). They are the following:

- TDC: Total Distances Combined, a combination of *TPD* and *TRD*. Equation 5.
- ADC: Average Distances Combined, a combination of *APD* and *ARD*. Equation 6.

- 4DC: 4 Distances Combined, the combination of the first four ones (*TPD*, *APD*, *TRD* and *ARD*). Equation 7.

$$\frac{\sum_{o=1}^{o=LO}\left\{W_{TPD}*\sum_{i=1,j=i}^{i=n,j=n}PD_{o,i,j}+W_{TRD}*\sum_{i=1}^{i=n}RD_{o,i}\right\}}{LO} \tag{5}$$

$$\frac{\sum_{o=1}^{o=LO}\left\{W_{APD}*\frac{\sum_{i=1,j=i}^{i=n,j=n}PD_{o,i,j}}{n_o}+W_{ARD}*\frac{\sum_{i=1}^{i=n}RD_{o,i}}{n_o}\right\}}{LO} \tag{6}$$

$$\frac{\sum_{o=1}^{o=LO}\left\{W_{TPD}*\sum_{i=1,j=i}^{i=n,j=n}PD_{o,i,j}+W_{APD}*\frac{\sum_{i=1,j=i}^{i=n,j=n}PD_{o,i,j}}{n_o}+W_{TRD}*\sum_{i=1}^{i=n}RD_{o,i}+W_{ARD}*\frac{\sum_{i=1}^{i=n}RD_{o,i}}{n_o}\right\}}{LO} \tag{7}$$

Lastly, the miscellaneous group which contains two fitness functions designed to be used together. The first one is the number of parcels present (Number of PLots, NPL), equation 8. Actually, the geometries and number of parcels do not change in the algorithm, so in order to use this fitness function properly, we have had to implement a way of allowing change in the geometries to change the number of parcels. This geometric transformation is performed in the second miscellaneous fitness function.

$$\sum_{o=1}^{o=LO} n_o \tag{8}$$

This second miscellaneous fitness function performs a geometric union of the parcels of each owner and then applies another fitness function, using the value returned by it. This means that two or more adjacent parcels will be considered as one when computing the fitness value, but will remain separated outside the fitness function (allowing for ownership to change independently of each other at a later point), at the expense of an important increase in the computational cost of the fitness function. Implementing the geometric transformation this way, we can use it with every fitness function, not only the previous one.

Other authors have expressed the owners' preference for keeping most of their parcels -or at least the largest ones, the most productive ones, or simply those in which they have fixed investments in place such as buildings or greenhouses- excluded from the exchange or LC process (Cay et al., 2010; Cay and Uyan, 2013). This is supported natively by the algorithm, allowing the selected parcels that must not be subject to the exchange process to be indicated (these parcels will however be taken into account for the fitness value).

2.8. Implementation

Fig. 5 and Fig. 6 show the pseudocode and a flowchart of the GA-PE algorithm, respectively. In the following paragraphs, all the steps are explained using the numbered lines of the pseudocode as reference. Let P be the population with M ownership patterns (algorithm solutions), P_i an ownership pattern with size N (parcels) whose genes are represented by P_{i_k} with $k = 1..N$ (gene k represents the landowner of parcel i). $F(P_i)$ represents the value of the fitness function of P_i.

In our implementation, an initial population (P) is created using the generation method selected (Section 2.5), accepting only valid solutions (lines 1-6). The algorithm then enters the main evolution loop, selecting two ownership patterns (P_i, determined by the loop index, P_j, randomly selected) from P (line 10), and applies the crossover (Section 2.3) to generate two children (lines 11-13). Then these two new children are mutated (applying the mutation function selected, see Section 2.4) (lines 14-25). If a child is not valid, it is mutated again, until a maximum number of mutations, in which case that child is discarded. After evaluating the remaining (valid) children (through the fitness function selected, see Section 2.7), the best ownership pattern among the children and parents is selected (line 26) and added to a new population (the next generation). This process is repeated for each ownership pattern in the population, until the new population is complete (the same amount of ownership patterns as in the previous generation) and. If the stopping criteria are satisfied (line 7), the algorithm returns the best ownership pattern as solution of the problem (line 31). While the stopping criteria are not satisfied, the new population is used starting the next iteration of the algorithm. Usually, two possible stopping criteria are defined for a genetic algorithm: a threshold value of the fitness function, or a given amount of processing time. In this case, establishing a threshold is not viable, since the optimal value is not known in each situation, so only processing time limit was used. To avoid spending resources when the population has already achieved a good fitness value and does not improve further, when the algorithms detects stagnation (a number of generations with all the elements with almost the same fitness value) several times, it can stop by itself. The amount of generations without changes and the number of times it has to be detected before stopping can be configured.

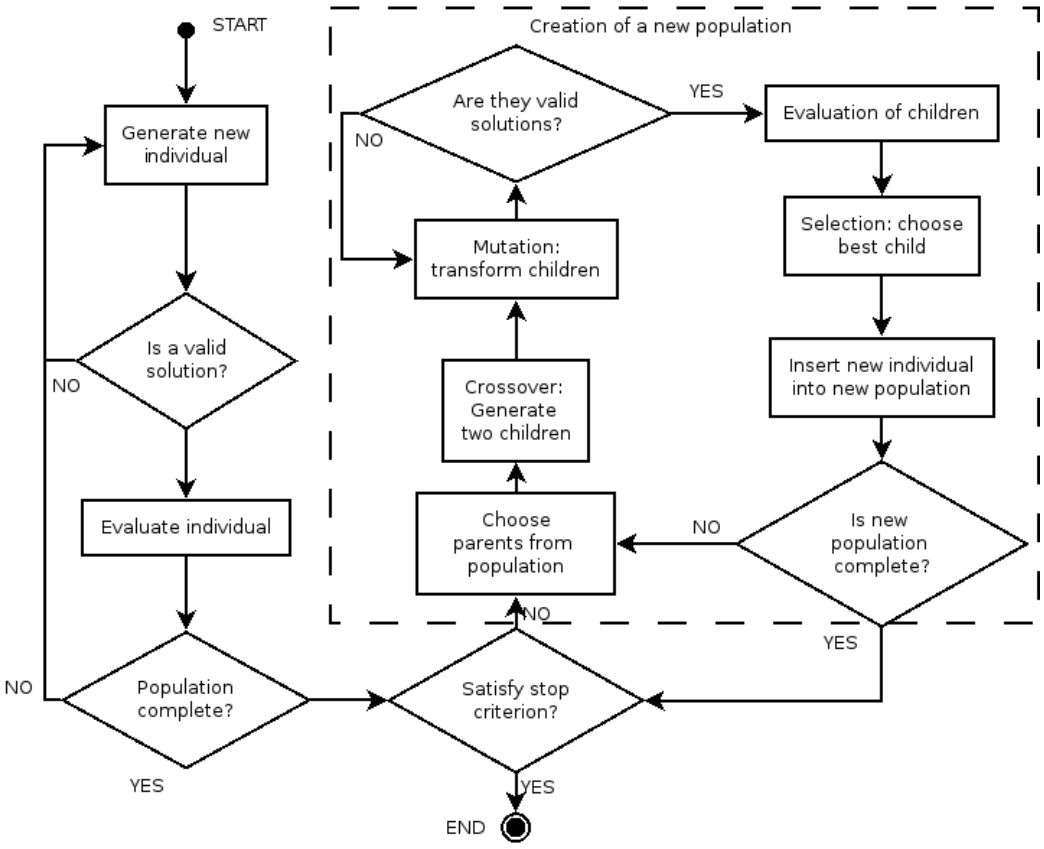
The algorithm was implemented using the Java programming language and it has been adapted to be executed from a command-line interface, in a server through a GIS Web Application, or in an external server (GeoServer). In the former case, the application uses local data and the computational resources of the client. In the second case, the user can visualize the input and output data using a GIS viewer integrated with the algorithm, allowing

Figure 5: Sequential GA_PE pseudocode.

```
1  for i = 1 to M do
2  |   Create P_i;
3  |   while P_i is not valid do
4  |   |   P_i = Create P_i;
5  |   end
6  end
7  while execution_time ≤ max_time do
8  |   i = 1;
9  |   while i ≤ M do
10 |   |   Select P_j where 1 ≤ i, j ≤ M and i ≠ j;
11 |   |   Add P_i and P_j to offspring candidates;
12 |   |   Randomly select k_0 and k_1 where 1 ≤ k_0 ≤ k_1 ≤ N;
13 |   |   P̃_i = P_{i_1}...P_{i_{k_0}} P_{j_{k_0+1}}...P_{j_{k_1}} P_{i_{k_1+1}}...P_{i_N};
14 |   |   P̃_j = P_{j_1}...P_{j_{k_0}} P_{i_{k_0+1}}...P_{i_{k_1}} P_{j_{k_1+1}}...P_{j_N};
15 |   |   Mutate P̃_i;
16 |   |   mutations_made = 1;
17 |   |   while P̃_i is not valid and mutations_made ≤ Max_mutations do
18 |   |   |   Mutate P̃_i;
19 |   |   |   mutations_made = mutations_made + 1;
20 |   |   end
21 |   |   if P̃_i is valid then
22 |   |   |   Add P̃_i to offspring candidates;
23 |   |   end
24 |   |   Mutate P̃_j;
25 |   |   mutations_made = 1;
26 |   |   while P̃_j is not valid and mutations_made ≤ Max_mutations do
27 |   |   |   Mutate P̃_j;
28 |   |   |   mutations_made = mutations_made + 1;
29 |   |   end
30 |   |   if P̃_j is valid then
31 |   |   |   Add P̃_j to offspring candidates;
32 |   |   end
33 |   |   P'_i = ownership pattern with best fitness value from offspring candidates;
34 |   |   i = i + 1;
35 |   end
36 |   P = P';
37 end
38 return P̂_γ where F(P̂_γ) = max{F(P_0), ..., F(P_M)};
```

the algorithm to be started and stopped, and the status of executions thereof checked. The last case uses the Web Processing Service (WPS) defined by the Open Geospatial Consortium (OGC) to allow integration with existing servers and applications. The last two cases allow users to send the input

Figure 6: Flowchart representing the steps followed by GA-PE



data from their client computers, let the server carry out the calculations, and receive the results once they are produced.

2.8.1. GA_PE Input/Output

The input of the algorithm consists of several files, these being the configuration file, the parcel data, and (optional in some cases) the reference points for each owner. The algorithm produces data in the same format as the input, with a different (potentially) landowner assigned to each parcel. The configuration file determines all the parameters involved in the algorithm, from population size, to the evaluation or generation method to use, allowing for full customization depending on the specific case and needs of the landowners.

The format selected for the geometric information of the input data, and the results produced, is the ESRI shapefile format. The input file containing

the parcel data must have at least one field in its attribute table, an identity code to represent the landowner before the exchange process. Optionally, it can have two more attributes: firstly, the value of the parcel (if not present the surface value of the parcel will be used), and secondly, whether or not the parcel should be excluded from exchanges (but taken into account for the metric); if not present all parcels are available for exchanging.

In some cases (if the selected evaluation method is based on distances to the reference points) another additional file is required, also in shapefile format, containing the reference point for each landowner. As the parcel file, the landowner code attribute is mandatory, with another optional attribute: the total value for that landowner, before the exchange process. If the total value is present, the validity of the ownership patterns will be checked against that value, otherwise, the total value will be calculated using the parcels assigned to the landowner in the input data.

When the algorithm starts, this information is processed, and the distances between parcels and to the reference points are calculated and stored, since they do not change during the execution of the algorithm, to improve performance eliminating repeated geometric operations.

The output of the algorithm is saved in the ESRI shapefile format, with the geometries and landowners assigned. If the value of the parcel has been provided in the input file, it will be included in the output file. Likewise, the information about excluding parcels from the exchange process is saved if initially provided. During the execution, the algorithm saves progress data in a log file, the ownership patterns of the first and last generation. It also has support for stopping the execution or saving the current population (and best ownership pattern at that moment) on demand.

3. Performance considerations: parallelization of GA-PE

Depending on the size of the problem (number of parcels and landowners involved) and on the configuration in use (especially if the parcel union is active) the computational time needed to reach a good solution may be excessive (upwards of 5 hours to achieve most of the fitness improvements using fast configurations, and increases with the size of the problem). Since the aim is to provide a useful tool to end users, a number of improvements focused on performance increase have been incorporated, taking advantage of parallel processing techniques.

The first improvement is the parallelization of the calculation of the next generation of ownership patterns, using the multithreaded capabilities of current microprocessors, and shared memory parallel programming. The steps of crossover, mutation, validation and evaluation are grouped and isolated

for each ownership pattern. Using the threads available (which can be configured), they are computed until the new population is completed. With this approach, in the event that one ownership pattern takes more time than others, the rest of the threads can continue with other ownership patterns (as long as there are ownership patterns pending for the next generation), thus reducing idle times. This parallelization strategy increases the depth of the search, as more generations can be computed in the same time.

The second improvement is the parallelization of the algorithm as a whole. In this approach, multiple populations are created and assigned to different threads, and each one evolves independently of each other, with occasional communication of the best ownership patterns found. Each population can be configured independently of each other, allowing the user to try different configurations in the same execution. In regular intervals (every nth generation, each population can have different values) a population sends its best ownership patterns to the other populations, and adds the ones sent to it to the next generation, replacing the worst ownership patterns. This communication is asynchronous, since the sender population does not wait until the other populations receive the elements, and when receiving elements, if it has not received any new ownership patterns, it continues with the next generation. These ownership pattern communications allow the cooperation of the populations: if one population finds a good ownership pattern, it will send it to the others, allowing them to get out of a local optimum, or speed up their evolution. When using this parallelization, the stop by stagnation only occurs when all populations have stagnated the required times at the same time, since some populations can help others move out of the stagnation state sending new ownership patterns. This parallelization has been designed in a way that can be adapted to other types of parallel programming paradigms (distributed computing, for example) with minor changes needed on the algorithm. Increasing the number of parallel populations, the algorithm makes a wider search, as each population can follow a different search path. While computation time reduction is not the focus of this approach, the cooperation of the different populations has a slight positive effect, and can reduce the time needed to achieve the same fitness values.

Finally, the two parallelization strategies can be combined, using multiple threads for each population. This allows more flexibility using all the resources available, providing means to assign them as desired. The number of threads to use in each population can be configured independently, so users can balance the depth and the width of the search. It can also be used to focus on a known good configuration, but having other populations in parallel with fewer resources to provide diversity to the main one, attempting to reduce stagnation by reaching a local optimum.

4. GA_PE validation results

In this section we will analyze the results obtained, using an extensive set of tests, each one focused on one aspect of the algorithm. The final goal is to acquire a global understanding of certain good practices to achieve good results using our algorithm.

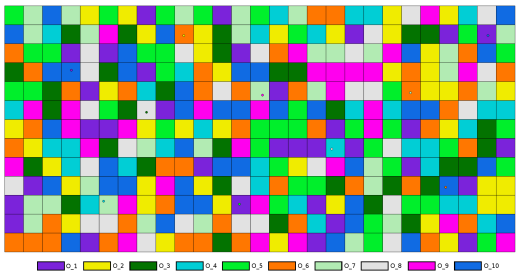
First, in order to demonstrate the capabilities of the algorithm in an ideal situation, in Fig 7a a synthetic parcel holding is shown, with equally sized adjacent parcels and 10 owners. In Fig 7b the result of the algorithm is shown. For this initial test the development computer was used (4 cores, 8 threads), with less than ten minutes of execution time and a configuration that showed its efficiency during testing. We used 4DC as the fitness function, for reference.

In Figure 7c the result of one execution using the PDA fitness functions is displayed. While the variability of the 4DC is low according to internal testing, this is not the case for every fitness function. For example, PDA is not aware of the reference points, so the groups of parcels can be anywhere in the parcel holding; they happen to be close to the corresponding reference point because the generation method used does take into account the reference point.

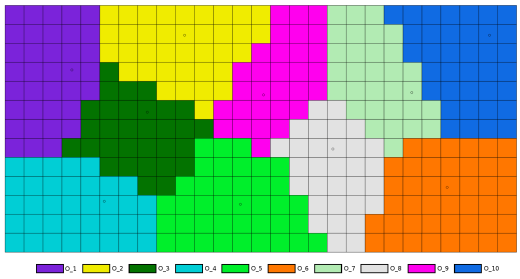
Lastly, in Figure 7d the result when using the Number of Parcels is shown. This fitness function does not take into account distances, so the shape of the parcel groups is very irregular. Additionally, using by itself this fitness function struggles with isolated parcel groups. If the number of parcels separating two groups of the same owner is greater than a few, the mutation is not able to change all the them to the same owner in order to join the two groups.

Moving on to real test cases, in Fig. 8 three real cases to be used for testing are shown, color-coded by the landowner of each parcel. The cases are representative of a variety of real situations, such as adjacent parcels and good land coverage (high percentage of land available for the exchange process) *vs* more disperse parcels, isolated groups of parcels versus only one group of parcels, regularly-shaped parcels versus irregular shapes and similarly-sized parcels versus big parcels alongside with small parcels. Although only 3 test cases could be considered a small test set, we are quite confident that they cover the majority of representative situations of land fragmentation present in the region under study: Galicia, Spain. All test cases have been executed on a system equipped with an AMD Threadripper 1950X, a CPU with 16 cores. Table 1 contains a summary of the details of each test case.

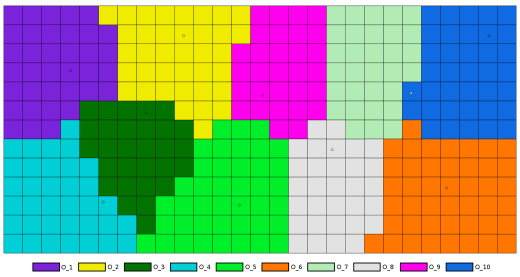
The testing procedure to be performed in this section is divided into several stages, each one providing information that is useful for the next ones.



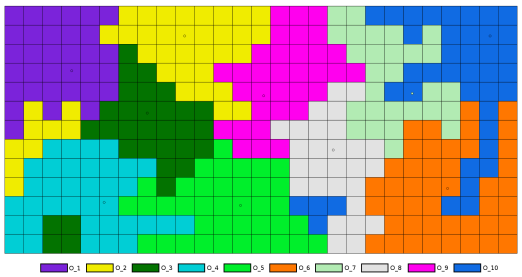
(a) Original synthetic parcel distribution with random owners.



(b) Algorithm results using 4DC fitness function.



(c) Algorithm results using average distance between parcels fitness function (PDA).



(d) Algorithm results using number of parcels fitness function (NPL).

This way, at the end we will have a few recommendations to maximize the benefits of the results obtained. We will start testing the mutation methods, to determine which one has greater potential. Once that is determined, that mutator will be used in the following tests. Next we will assess shortly the parallelization techniques and several fitness functions, to determine the best configuration for general use.

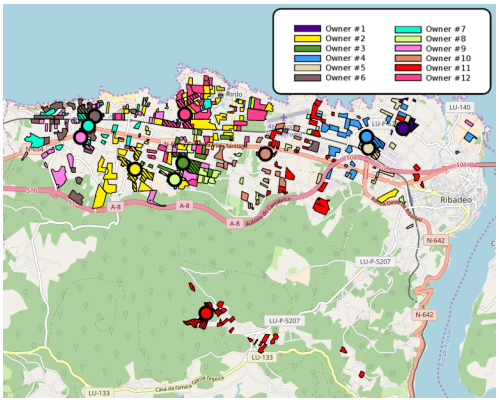
Using the best mutation method, the best hardware resources distribution and the best overall evaluation function, we will run the algorithm on the real test cases, and analyze the results obtained. Finally, we will see the potential of the use of heterogeneous architectures, using multiple populations with different algorithm parameters in parallel.

4.1. Mutation methods evaluation

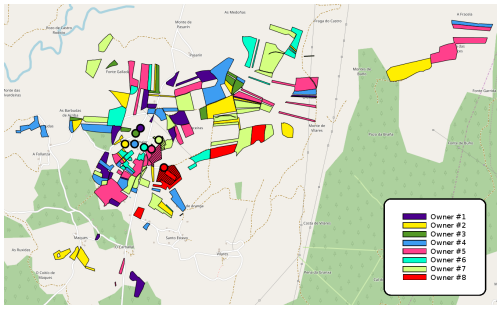
Since the mutation method plays an important role in the evolution of the algorithm, we will study which of the mutation methods implemented provides the best results. The testing is based on several executions of the algorithm using each mutation method, keeping the other parameters unchanged to reduce the variability present in the problem to the mutation method in use. We test the three mutation methods explained in section 2.4

Test cases	Municipality	Parcels	Owners	Characteristics
Test case 1	Ribadeo	329	12	Low size variation Low land coverage High parcel dispersion High reference points dispersion
Test case 2	Aranga	155	8	Low size variation Average land coverage Average parcel dispersion Low reference points dispersion Two parcel groups far away
Test case 3	Pol	146	20	Low size variation High land coverage Low parcel dispersion Average reference points dispersion

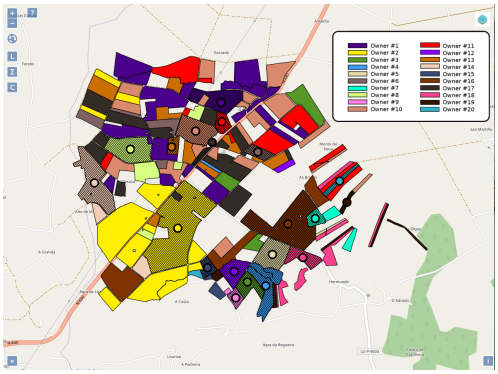
Table 1: Test cases information



(a) Test case 1, Ribadeo



(b) Test case 2, Aranga



(c) Test case 3, Pol

Figure 8: Real test cases. Each color represents an owner.

using two different combinations of them. The algorithm was configured to allow up to 100 consecutive mutations of the same ownership pattern if the result is not valid. The rest of the parameters are set to values that provided acceptable results during development and initial testing.

When using more than one method, each mutation is applied to both children resulting from the crossover This technique generates 2N children, N being the number of mutation methods to apply and in case remutations are needed, the next mutation method is applied in consecutive mutations. For example, child one is mutated with first mutator; if not valid, second mutator is applied; if still not valid, third mutator is applied, and so on. When all of them were applied and more mutations are still needed, the process is started again with the first of all mutation methods.

The first combined method uses the Swapping and Random Change simple mutators. In this case, the swapping mutator performs one parcel swap, and the random change performs two random changes. The second mutation method tested is a combination of SwapNPlots and Random Change. The SwapNPlots is configured to select two random owners and exchange two parcels of each one, while the Random Change mutator performs 2 changes.

In Figure 9 the results are shown. Each configuration was run ten times, and the average fitness value of the ten executions is displayed in the graph, taking measurements in a 15-second interval. The graph shows that the combined configuration that uses the Swapping and Random Change (first configuration) mutation methods is the fastest one, while also capable of achieving lower fitness values, thus providing greater potential. The second configuration gets close in terms of fitness value achievable, but takes more time to reduce the fitness value.

4.2. Parallelization evaluation

To evaluate the parallelizations implemented, and their effect on the algorithm results, we tested the efficiency when increasing computational resources. For this testing we used the same approach as in the previous section, 10 runs and averaging the results. To avoid overextending in the paper and since this testing falls outside the field of the journal, we will keep this section short and focused on the conclusions.

We tested different distributions of the computational resources between number of populations and resources for each population. The results revealed that the best approach is a balanced distribution of resources, in this case 4 populations using each of them 4 of the available threads. This distribution presents an average behavior regarding execution time until stagnation, but achieves the best fitness value. On the following tests we will use this resource distribution.

Figure 9: Fitness evolution with different mutation methods

4.3. Fitness functions evaluation

The best fitness function is notably subjective, as it depends on the needs and priorities of the group of landowners. For our testing, we opted for a balanced fitness function that has positive effects on multiple aspects of the parcel holding. That way we can recommend a generic configuration that works in the majority of situations, and for those that have special needs or priorities, the user can choose the fitness function that fits better.

After our testing, the 4DC fitness function has proven to be the one that improves the four basic ones based on distances (between plots and to the reference point). According to the synthetic testing, it is also the most stable, providing more repeatable results. For this reasons, we will use this fitness functions for testing from now on.

4.4. Real case testing

Using the knowledge acquired in the previous testing, we proceed to apply our best strategies to real situations. Using the best mutation strategy found in Section 4.1 (Swapping + Random Change), the best resource distribution found in Section 4.2 (4 populations of 4 threads) and the best overall evaluation function found in Section 4.3 (4DC), we run the algorithm one time for each test case, with a maximum execution time of 24 hours. The rest of the algorithm parameters are the same as in the previous section.

First, in Table 2, values of the fitness function using the chosen best strategy for our GA_PE algorithm (4DC) together with those calculated using

the fitness functions based on distances, and the number of plots are shown (six fitness functions in total). Table 2a shows, for each test case described in Table 1, the fitness value at the starting situation and the value of the best solution achieved, for each fitness function shown. We provide the fitness functions values in addition to 4DC values in order to facilitate the reader translating those results to the real world. When combining multiple fitness functions it is more difficult to extract conclusions, because the effect of each ownership pattern fitness function to the combined result is not known. A positive result may hide a negative change in some of the other fitness functions that form a combined fitness function (see test cases 3 and 4).

Table 2b shows the relative improvement between the initial and final values. A value of 30% indicates a reduction of 30% in the evaluation function. As can be seen, a minimum reduction of at least 38.64% is achieved in all cases using 4DC fitness function, with some cases reaching a reduction of almost 70% in the value of the evaluation function. In some cases, the total distance between parcels is increased (the source of the negative value in the table), as that test case has some parcels far away from others, and they are assigned to a owner with a high number of parcels, so the distance to all of them adds up. Since the weight of that evaluation function is very low compared to the others, if a change improves the other evaluation functions a little, it can offset that increase in the total distance between parcels, and provide a reduction in the total fitness value.

	Starting values						Final values					
	TPD	**APD**	**TRD**	**ARD**	**NPL**	**4DC**	**TPD**	**APD**	**TRD**	**ARD**	**NPL**	**4DC**
Test case 1	749,019.0	1,289.4	34,802.4	1.053.3	320.0	1582.76	411,026.7	681.5	21,874.7	662.4	265.0	929.80
Test case 2	354,505.5	1,183.2	19,660.7	859.6	151.0	1233.37	411,847.5	520.7	18,153.5	543.7	117.0	756.71
Test case 3	26,349.7	552.1	3,571.4	447.4	146.0	528.47	8,107.1	113.4	1,878.2	168.8	89.0	161.68

(a) Changes on the five basic fitness functions and 4DC in each test case

	Improvement (%)					
	TPD	**APD**	**TRD**	**ARD**	**NPL**	**4DC**
Test case 1	45.1	47.1	37.1	37.1	17.2	41.25
Test case 2	-16.2	56.0	7.7	36.7	22.5	38.64
Test case 3	69.2	79.5	47.4	62.3	39.0	69.40

(b) Improvement of each fitness function in each test case

Table 2: Changes on the fitness functions in each test case

Lastly, in Figures 10 to 12, the effects on the final distribution of the parcels are shown visually. The parcels with diagonal black lines are parcels that are taken into account for the evaluation function but which were not allowed to change ownership during the algorithm. On the left the initial situation is shown, color-coded by landowner, and on the right, the best solution found by the algorithm. As can be observed, there is a significant parcel

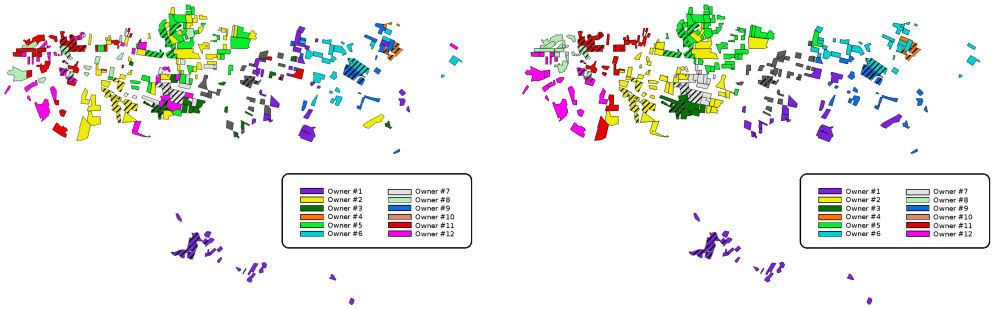


Figure 10: Case 1: Similar size and disperse parcels, low land coverage.

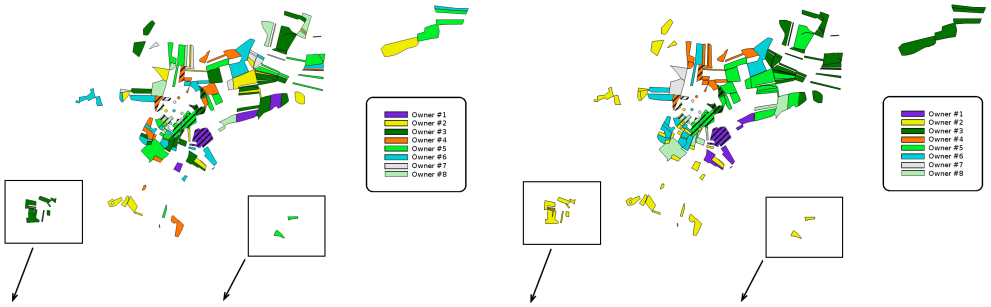


Figure 11: Case 2: Similar size with all reference points together. The parcels in the rectangles are located far away in the direction indicated. They are displayed out of their real place for visualization purposes.

distribution improvement, displaying a good amount of grouped parcels as Table 2 numerically indicated, especially in cases with high land coverage.

The best improvements was made in case 3, with an improvement of over 60% on *4DC* and the best improvements in number of parcels. This is due to the high coverage ratio, and the high percentage of contiguous fragments of land involved in the exchange process. This situation allows multiple parcels to be joined, forming larger pieces of land assigned to the same owner, greatly reducing some of the metrics.

The worst improvement was in case 2, with 38.64% of improvement. Even this case reduces the number of parcels, but the groups of parcels far away from the rest have a significant impact of some of the metrics (*TPD* and *TRD*). Due to the fact that all the owners' reference points are close together, every owner competes with the others for the same parcels, and the metrics involving the reference points show the lowest improvement values in this case. On average, the algorithm achieves a reduction of 49.76% on the value of 4DC.

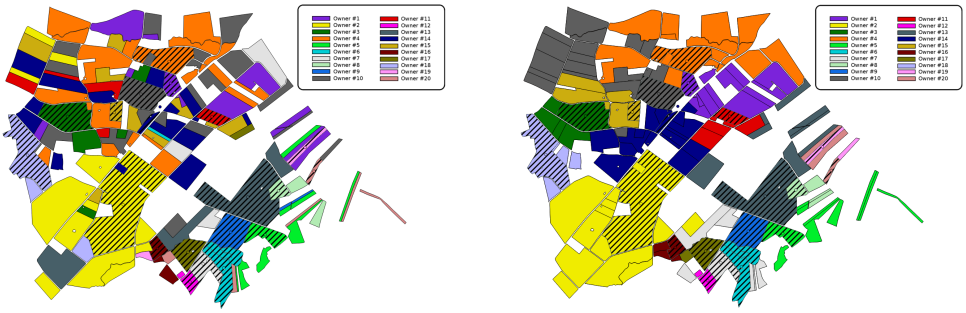


Figure 12: Case 3: Similar size adjacent parcels, high land coverage.

4.4.1. GA_PE versus farmers experience

The solutions produced for some of the test cases were presented to a group of the landowners involved, during individual interviews. Most of the interviewed participants expressed interest in the proposed solution, in general, indicating that it would be preferable to the current situation. Nevertheless, most of them also suggested small deviations from the proposal, usually involving specific parcels which on second thought, they would rather not include in the exchange process. Less commonly, some owners identified small changes in the solution that would be preferable owing of reasons not originally included in the computation process (e.g., ease of access to a block of parcels assigned to them by the proposed solution). This was in line with our expectations that the solution produced by the algorithm could not be immediately implemented in the field, but it showed clearly that it was a good starting point for negotiation.

Analyzing the results obtained, we can extract a number of relations between the fitness functions and the different possible situations that can be encountered when using the algorithm.

First, the algorithm achieves better results in situations with high land coverage and adjacent parcels (case 3). This is due to the higher probability of forming large groups of adjacent parcels assigned to the same owner, which considerably reduces the fitness value of 4DC. This is the best possible scenario for the algorithm, particularly if the fitness function that joins adjacent parcels of the same owner is involved.

Secondly, situations where the reference points are close together can decrease the improvement achieved (case 2 with TRD is a good example). In this case, it is better to employ fitness functions that do not use the reference points, since different owners will compete for the same parcels and cause the closest parcels to be fragmented among each other. At the very least, the user should combine that fitness function with other ones that also

take into consideration other parameters. The algorithm can then use those parameters to determine which possible ownership pattern is better.

In situations with low land coverage and dispersed parcels and reference points (case 1), fitness functions that use the distance to the reference point provide good results, given the intrinsic limitations of the situation. Fitness functions that only take into account distance between parcels may form parcel groups that are far away from the owner's reference point, but closer together than assigning parcels nearer to the reference point.

4.5. Cooperation testing

To close the results section, we will test how our implementation can benefit from the use of different algorithm parameters in each parallel population. We will use the best mutation method, the best resource distribution and the best overall fitness function (trying some variations) found in previous sections, with GA_PE being executed using the second test case. Results shown in Figure 13 come from the average of ten runs.

The best resource distribution uses 4 populations, and we will consider several combinations with regards to the fitness functions assigned to each population. On the one hand, we will use the 4DC fitness function (referred to as *fast*); and on the other hand, to test a fitness function with high execution times requirements we will use 4DC but applied to the union of the parcels of the same owner (referred to as *slow*). The results are shown in Figure 13. The series naming indicates the number of populations using the fast and slow fitness functions, F4P_S0P indicates that all 4 populations use the fast (F) fitness function and none the slow (S) one, while F1P_S3P indicates that 1 population uses the fast (F) fitness function and the other 3 use the slow (S) one. The *_4DC* suffix indicates that the fitness value is reported by the fast fitness function, and the *_U* suffix indicates that the fitness value is reported by the slow one (the parcel union was performed).

The fast metric alone (F4P_S0P) quickly converges to an almost steady fitness value, and stagnates, while the slow metric alone (F0P_S4P) has not reached a stagnation value by the end of the allowed time. However, in any of the configurations where both metrics are used, there is a remarkable reduction in the fitness value reported by the slow fitness function; this behavior is mainly due to the first communication between populations, when the ownership patterns of the population that uses the fast metric, which have achieved a notable improvement of their fitness, are inserted into the population that uses the slow metric (another smaller reduction takes place during the second communication). On the other hand, the use of the two metrics has very little impact on the evolution of fast populations.

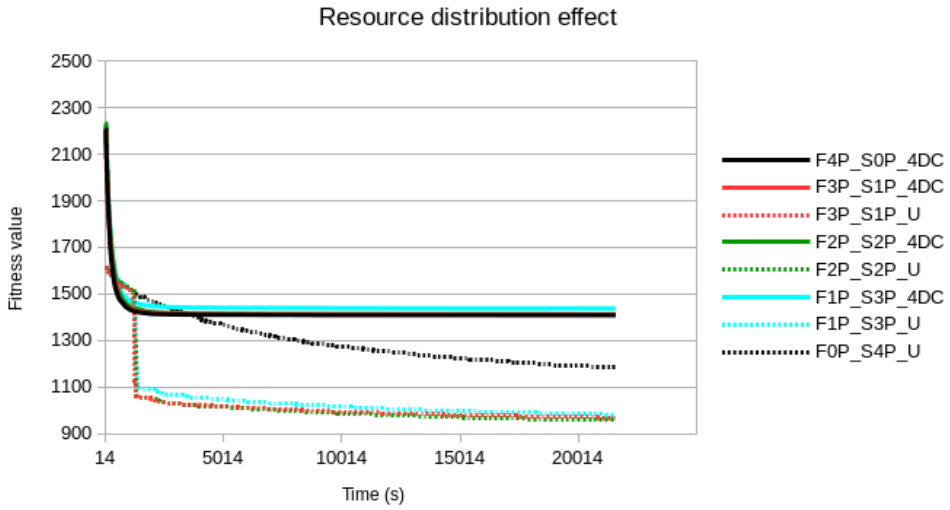


Figure 13: Heterogeneous configuration resource distribution comparison

Although allowing more execution times could still improve the metric when using the slow fitness function (downward trend when it stops), there is a remarkable improvement using the two fitness functions, especially if they are synergetic (an improvement in one tends to cause an improvement in the other). For this reason we encourage the use of different fitness functions when using our GA_PE algorithm.

5. Conclusions

A genetic algorithm for parcel exchange (GA_PE) has been designed. Tests performed have allowed us to determine a basic configuration with regards to mutation and evaluation methods, and a recommendation about the computational resource distribution (balanced approach between populations and threads per population). These recommendations can be used as default configuration when employing the algorithm, but the user is encouraged to test other options to check whether they are a better fit for their test case. Finally, a heterogeneous configuration has demonstrated their potential in our testing, improving the results notably when using different synergetic evaluation methods.

The use of an evolutionary algorithm makes parcel exchange among multiple landowners a viable tool for reducing the internal fragmentation of existing holdings as it helps to overcome its main practical limitation: finding an adequate exchange solution in reasonable time. A solution that groups each landowner's parcels as close as possible could be easily implemented on

the field with no modification of current parcel structure and with little variation of the total area managed by each owner. Nevertheless, in the real test cases explored in this paper, it was common for owners to propose small deviations from the solution produced by the algorithm, but these departures usually involved a small number of parcels and could be easily negotiated among participants. Accordingly, the solution produced by the algorithm appeared as a solid base for negotiations that could not be achieved in the initial situation.

As future work, there are possible improvements to be made, for instance adding a new step in the algorithm to perform a local search on certain ownership patterns. This search can be more computationally intensive, as it would not be applied to all ownership patterns, and not on every generation. To avoid doing work on already explored ownership patterns, a taboo list could be used. Another possible improvement would be to apply a similar technique to the one explained in Ertunç et al. (2018b), changing the mutation configuration based on the stagnation detected, by either changing the mutation method configuration, changing to more aggressive mutation methods or a combination thereof, using more conservative and faster mutation options when the algorithm evolves fast.

Role of the funding source

This research has been supported by the Government of Galicia (Xunta de Galicia) under the Consolidation Programme of Competitive Reference Groups, co-founded by ERDF funds from the EU [Ref. ED431C 2017/04]; under the Consolidation Programme of Competitive Research Units, co-founded by ERDF funds from the EU [Ref. R2016/037]; by Xunta de Galicia (Centro Singular de Investigación de Galicia accreditation 2016/2019) and the European Union (European Regional Development Fund, ERDF) under Grant Ref. ED431G/01.

References

Akkus, M., Karagoz, O., Dulger, O., July 2012. Automated land reallotment using genetic algorithm. In: Innovations in Intelligent Systems and Applications (INISTA), 2012 International Symposium on. pp. 1–5.

Bennett, D. A., Xiao, N., Armstrong, M. P., 2004. Exploring the geographic consequences of public policies using evolutionary algorithms. Annals of the Association of American Geographers 94 (4), 827–847.

Binns, B., 1950. The consolidation of fragmented agricultural holdings. Food and Agriculture Organization of the United Nations.

Borgwardt, S., Brieden, A., Gritzmann, P., 2014. Geometric clustering for the consolidation of farmland and woodland. The Mathematical Intelligencer 36 (2), 37–44.

Cay, T., Ayten, T., Iscan, F., 2010. Effects of different land reallocation models on the success of land consolidation projects: Social and economic approaches. Land Use Policy 27 (2), 262–269, forest transitions Wind power planning, landscapes and publics.

Cay, T., Uyan, M., 2013. Evaluation of reallocation criteria in land consolidation studies using the Analytic Hierarchy Process (AHP). Land Use Policy 30 (1), 541–548.

Crecente, R., Alvarez, C., Fra, U., 2002. Economic, social and environmental impact of land consolidation in Galicia. Land Use Policy 19 (2), 135–147.

Demetriou, D., Stillwell, J., See, L., 2012. Land consolidation in Cyprus: Why is an Integrated Planning and Decision Support System required? Land Use Policy 29 (1), 131–142.

Djanibekov, U., Finger, R., 2018. Agricultural risks and farm land consolidation process in transition countries: The case of cotton production in uzbekistan. Agricultural Systems 164, 223 – 235.

Du, X., Zhang, X., Jin, X., 2018. Assessing the effectiveness of land consolidation for improving agricultural productivity in china. Land Use Policy 70, 360 – 367.

Ertunç, E., Çay, T., Haklı, H., 2018a. Modeling of reallocation in land consolidation with a hybrid method. Land Use Policy 76, 754 – 761.

Ertunç, E., Çay, T., Haklı, H., 2018b. Modeling of reallocation in land consolidation with a hybrid method. Land Use Policy 76, 754 – 761.

Goldberg, D. E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning, 1st Edition. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Hartvigsen, M., 2014. Land reform and land fragmentation in Central and Eastern Europe. Land Use Policy 36 (0), 330–341.

Kilgore, M. A., Snyder, S. A., 2016. Exploring the relationship between parcelization metrics and natural resource managers' perceptions of forest land parcelization intensity. Landscape and Urban Planning 149, 43 – 48.

Kupidura, A., Luczewski, M., Home, R., Kupidura, P., 2014. Public perceptions of rural landscapes in land consolidation procedures in Poland. Land Use Policy 39 (0), 313 – 319.

Lisec, A., Primožič, T., Ferlan, M., Šumrada, R., Drobne, S., 2014. Land owners' perception of land consolidation and their satisfaction with the results – Slovenian experiences. Land Use Policy 38 (0), 550–563.

Liu, Y., Tang, W., He, J., Liu, Y., Ai, T., Liu, D., 2015. A land-use spatial optimization model based on genetic optimization and game theory. Computers, Environment and Urban Systems 49, 1 – 14.

Lu, H., Xie, H., He, Y., Wu, Z., Zhang, X., 2018. Assessing the impacts of land fragmentation and plot size on yields and costs: A translog production model and cost function approach. Agricultural Systems 161, 81 – 88.

Orea, L., Perez, J. A., Roibas, D., 2015. Evaluating the double effect of land fragmentation on technology choice and dairy farm productivity: A latent class model approach. Land Use Policy 45 (0), 189 – 198.

Pašakarnis, G., Morley, D., Maliene, V., 2013. Rural development and challenges establishing sustainable land use in Eastern European countries. Land Use Policy 30 (1), 703–710.

Porta, J., Parapar, J., Doallo, R., Barbosa, V., Santé, I., Crecente, R., Díaz, C., 2013a. A population-based iterated greedy algorithm for the delimitation and zoning of rural settlements. Computers, Environment and Urban Systems 39 (0), 12–26.

Porta, J., Parapar, J., Doallo, R., Rivera, F. F., Santé, I., Crecente, R., 2013b. High performance genetic algorithm for land use planning. Computers, Environment and Urban Systems 37 (0), 45–58.

Porta, J., Parapar, J., García, P., Fernández, G., Touriño, J., Doallo, R., Ónega, F., Santé-Riveira, I., Díaz, P., Miranda-Barrós, D., Crecente-Maseda, R., 2013c. Web-gis tool for the management of rural land markets - application to the land bank of galicia (nwspain). Earth Science Informatics 6, 209–226.

Rendenieks, Z., ģerts Nikodemus, O., Brūmelis, G., 2015. The implications of stand composition, age and spatial patterns of forest regions with different ownership type for management optimisation in northern latvia. Forest Ecology and Management 335, 216 – 224.

Santé, I., Rivera, F. F., Crecente, R., Boullón, M., Suárez, M., Porta, J., Parapar, J., Doallo, R., 2016. A simulated annealing algorithm for zoning in planning using parallel computing. Computers, Environment and Urban Systems 59, 95 – 106.

Sklenicka, P., Janovska, V., Salek, M., Vlasak, J., Molnarova, K., 2014. The farmland rental paradox: Extreme land ownership fragmentation as a new form of land degradation. Land Use Policy 38 (0), 587 – 593.

Stewart, T. J., Janssen, R., 2014. A multiobjective gis-based land use planning algorithm. Computers, Environment and Urban Systems 46, 25 – 34.

Suárez, M., Santé, I., Rivera, F. F., Crecente, R., Boullón, M., Porta, J., Parapar, J., Doallo, R., 2011. A Parallel Algorithm Based On Simulated Annealing For Land Use Zoning Plans. In: The 2011 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA). Las Vegas, Nevada, EEUU, pp. 360–366.

Touriño, J., Rivera, F. F., Álvarez, C., Dans, C. M., Parapar, J., Doallo, R., Boullón, M., Bruguera, J. D., Crecente, R., González, X. P., 2001. COPA: a gis-based tool for land consolidation projects. In: ACM-GIS 2001, Proceedings of the Ninth ACM International Symposium on Advances in Geographic Information Systems, Atlanta, GA, USA, November 9-10, 2001. pp. 53–58.
URL https://doi.org/10.1145/512161.512174

Touriño, J., Parapar, J., Doallo, R., Boullón, M., Rivera, F. F., Bruguera, J. D., González, X. P., Crecente, R., Álvarez, C., 2003. Research article: A gis-embedded system to support land consolidation plans in galicia. International Journal of Geographical Information Science 17 (4), 377–396.

Uyan, M., Cay, T., Inceyol, Y., Hakli, H., 2015. Comparison of designed different land reallocation models in land consolidation: A case study in konya/turkey. Computers and Electronics in Agriculture 110 (0), 249 – 258.

Vallejo, M., Rieser, V., Corne, D. W., 2015. Genetic algorithm evaluation of green search allocation policies in multilevel complex urban scenarios.

Journal of Computational Science 9, 57 – 63, computational Science at the Gates of Nature.

van Dijk, T., 2003. Scenarios of Central European land fragmentation. Land Use Policy 20 (2), 149–158.

Vranken, L., Swinnen, J., 2006. Land rental markets in transition: Theory and evidence from hungary. World Development 34 (3), 481 – 500.

Xiao, N., Bennett, D. A., Armstrong, M. P., 2007. Interactive evolutionary approaches to multiobjective spatial decision making: A synthetic review. Computers, Environment and Urban Systems 31 (3), 232 – 252, geoComputation 2005.

Xin, H., Zhi-xia, Z., 2008. Application of genetic algorithm to spatial distribution in urban planning. In: IEEE International Symposium on Knowledge Acquisition and Modeling Workshop. pp. 1026–1029, wuhan, China.