A simulated annealing algorithm for zoning in planning using parallel computing

Inés Santé[a], Francisco F. Rivera[b,*], Rafael Crecente[a], Marcos Boullón[a], Marcos Suárez[a], Juan Porta[c], Jorge Parapar[c], Ramón Doallo[c]

[a] *Land Laboratory. University of Santiago de Compostela. Spain.*
[b] *CiTIUS. University of Santiago de Compostela. Spain.*
[c] *Computer Architecture Group. University of A Coruña. Spain.*

Abstract

There is an increasing demand for tools that support land use planning processes, particularly the design of zoning maps, which is one of the most complex tasks in the field. In this task, different land use categories need to be allocated according to multiple criteria. The problem can be formalized in terms of a multiobjective problem. This paper generalizes and complements a previous work on this topic. It presents an algorithm based on a simulated annealing heuristic that optimizes the delimitation of land use categories on a cadastral parcel map according to suitability and compactness criteria. The relative importance of both criteria can be adapted to any particular case. Despite its high computational cost, the use of plot polygons was decided because it is realistic in terms of technical application and land use laws. Due to the computational costs of our proposal, parallel implementations are required, and several approaches for shared memory systems such as multicores are analysed in this paper. Results on a real case study conducted in the Spanish municipality of Guitiriz show that the parallel algorithm based on simulated annealing is a feasible method

*Corresponding author
Email address: ff.rivera@usc.es (Francisco F. Rivera)
URL: citius.usc.es (Francisco F. Rivera)

[1] **Acknowledgment:** In memory of Rafael Crecente. This work has been partially supported by the Ministry of Education and Science of Spain, FEDER funds under contract TIN 2013-41129P, and Xunta de Galicia, GRC2014/008 and EM2013/041. It has been developed in the framework of the European network HiPEAC-2, the Spanish network CAPAP-H, and Galician network under the Consolidation Program of Competitive Research Units (Network ref. R2014/049).

to design alternative zoning maps. Comparisons with results from experts are reported, and they show a high similarity. Results from our strategy outperform those by experts in terms of suitability and compactness. The parallel version of the code produces good results in terms of speed-up, which is crucial for taking advantage of the architecture of current multicore processors.
Keywords: Land use optimization, Land use planning, Parallel algorithms for multicores, Decision support, Simulated annealing

1. Introduction

The design of a land use map is a laborious task that requires deep knowledge and expertise. The development of new automatic processes and tools to help public administrations and technicians in this task is of strategic importance. In this work a novel mechanism to achieve near-optimal solutions to this problem is introduced. It is formulated in terms of a multiobjective optimization problem in which plots are allocated to the most appropriate land category for it. **Plots** are land basic elements that can be assigned one category, in our case they are cadastral plots. Figure 1 shows a group of 15 plots that are used as an example in this paper. Objectives to be considered often include land suitability for the land category [1] [2] [3]. Also, some authors consider spatial criteria, especially the compactness of the regions allocated to one single category [4] [5] [6] [7] [8] because an irregular allocation of land categories in small, scattered, unconnected areas is usually undesirable in terms of economic and technical impact.

The problem of allocating different categories to specific land units can be established formally as a combinatorial optimization problem. A large number of alternative solutions can be usually found, and their quantitative comparison is usually important to validate the quality of the solutions and to justify them. Moreover, the number of plots involved in a municipal land use plan is usually large. Because these two factors lead to a high computational load, the search for the optimal solution usually calls for the use of heuristic algorithms capable

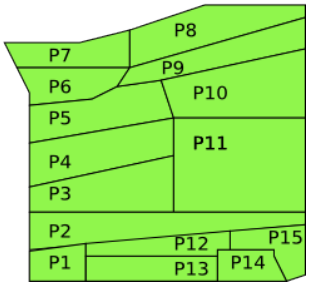


Figure 1: *Example of a set of plots*

of achieving near-best solutions in a reasonable time [9]. As a consequence, heuristics are used to obtain near optimal solutions. In particular, a number of authors have used algorithms based on the simulated annealing technique to optimize the allocation of land uses to spatial entities [10] [11] [12] [5] [13] [14] [15] [16]. All these iterative algorithms operate on a regular raster grid. However, land use allocation based on a regular grid is usually unrealistic as it may lead to a single-land use plot allocated to several categories or, more frequently, to a group of very different plots allocated to a single category. Therefore, the use of a coarse raster grid can create areas of assumed homogeneous land that may contain variability [17]. In addition, the planning laws in the study area often require land use zoning to be based on cadastral plots. We argue that the use of plot polygons instead of grid cells is more convenient but involves using complex compactness metrics based on geometric characteristics of these regions such as their area and perimeter.

The proposed objective function that guides the simulated annealing combines two subobjectives: maximization of land suitability and maximization of compactness. This approach complements a previous work [18] in which a first approach to the proposed objective function is established with preliminary results. In [19] genetic algorithms are used to deal with this problem based on the former objective function. In this paper we present an objective function that is a generalization of the ones presented in the mentioned couple of papers.

The data structure was also modified to improve the performance of memory accesses and computations. In this paper the problem is generalized, and a different heuristic is used that requires fewer control parameters. We show that the results are close to the solutions produced by experts. Two spatial metrics are proposed to evaluate compactness: a function based on **patches**, which are groups of adjacent plots with the same category, and another function based on categories,in which the plots are grouped into categories. Both metrics were introduced in [18]. The zoning solutions provided by the algorithm are better than or, at least, similar to the solutions provided by experts in terms of the objective function. These solutions also increase the rationality in the development of the zoning map. The algorithm was applied to land use zoning in the municipality of Guitiriz, Galicia, NW Spain, as a case study. This case study was also used in [18] and [19].

The large number of plots involved in municipal land use implies that the whole search space is usually huge. Therefore, the number of possible feasible solutions can be large. For this reason, the use of parallel computing has been considered as the only reliable alternative. In [18], message passing was used to parallelize the proposal in distributed memory systems. In contrast, this paper focuses on shared memory paradigm that take advantage of the fine grain parallelism, and complements message passing. Nowadays, parallel solutions are needed because of the presence of multicore processors in the market, and therefore shared memory implementations are more demanded. Celmatis et al. [20] and Mineter and Dowers [21] pointed out that the impact of parallel computing in Geographical Information Science is slight and that there is a need to develop parallel geoprocessing algorithms. Li [22] proposed parallel computation as one of the priority research lines in land use simulation and optimization models that can solve real-world application problems. Many proposals for parallelization can be found in the literature [23] [24] [25]. In this paper, a geometric parallelism, according to the classification of Ding and Densham [26], has been used to reduce execution time. This kind of parallelism is based on the partition of the spatial domain into sub-regions [27] that can be currently handled

by different processes.

A parallel simulated annealing algorithm for land use spatial allocation that uses an irregular spatial structure based on a cadastral parcel map is presented in this paper. The shared memory paradigm was used to implement the parallel code because it suits perfectly multicore systems. This strategy complements the message passing approach introduced in [19]. Typically, this kind of problems can not be solved analytically, and no unique solutions can be found. The use of plot polygons implies that the geometric characteristics of these regions must be recalculated at each iteration of the algorithm, which is a high time-consuming task. Our approach deals with this problem by decreasing the cost of this task by reducing the computations to those plots that change this category in each iteration. In addition, we propose a spatial parallelization to reduce execution time, by balancing the partition of the area under study into a number of so-called clusters that can be processed in parallel.

The paper is organized into three more sections. The first one defines the features of the optimization problem, the design of the simulated annealing algorithm, the implementation of the parallel version of the algorithm and the experimental results. The next section is devoted to the application of the proposed algorithm to a particular case study. Finally some conclusions and ideas for future work are presented.

2. Problem statement and methodology

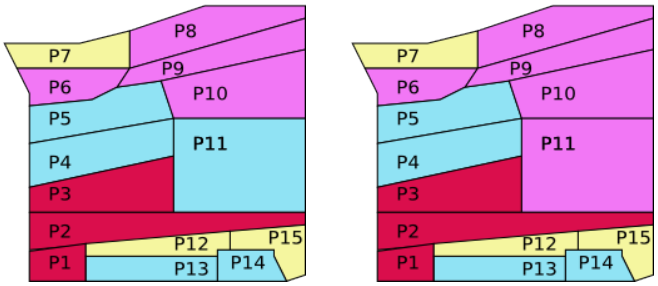
Land use planning laws define a set of land use categories and the restrictions enforced in each category. For some categories, spatial allocation is completely and uniquely determined by legal restrictions. We will refer to this group of categories as fixed categories. For example, in the study area the law establishes that water protection land corresponds to buffer zones around the waterways. Accordingly, land use allocation comprises two stages: the application of law restrictions for the delimitation of fixed categories, and the making of decisions by planners for the allocation of non-fixed categories.

For the first stage, a preprocessing module was used to allocate the fixed categories which are allocated by applying the planning laws using geometric operations (buffers, intersections, differences, etc.). This stage is the same as the one used in [18]. This stage is presented in Section 2.1.

For the second stage, a heuristic algorithm based on simulated annealing is proposed to delimit the non-fixed categories. A number of parameters are introduced to guide the process. In this way, by tuning these parameters, the user has the possibility of focusing the final result according to his/her preferences. At this point, it is important to note that laws and experts advise that spatial allocation should take the current boundaries of the existing plots in the municipality into account, i.e., a plot should not be split into several parts with different categories. Therefore, the problem can be defined as the distribution of N plots among M different non-fixed categories according to two objectives: maximization of the overall suitability S of the plots for the categories allocated to them, and maximization of the compactness C of the resultant land use patches. Land use patches are defined as the areas delimited by the polygons that result from the union of neighbouring plots allocated to the same category. Then, for a given distribution of plots to categories, a number of patches G is defined. Each of them is determined by the plots in a given category that are connected by neighbourhood.

Figure 2(a) shows an example of assignment for the 15 plots of figure 1 into 4 categories, shown in different colours. This example shows six patches defined by the sets of plots {P1, P2, P3}, {P4, P5, P11}, {P6, P8, P9, P10}, {P7}, {P12, P15} and {P13, P14}. Note that, patches {P4, P5, P11} and {P13, P14} correspond to the same category, and patches {P7} and {P12, P15} belong to the same category.

The optimization is subjected to the following constraint: the total area allocated to each non-fixed category cannot exceed the minimum and maximum values set by planners. Therefore, if S_{ij} is the suitability of plot i to the j-th category, and C_{kj} is the compactness of patch k that includes plots assigned to the j-th category, the optimization problem consists of obtaining the best



(a) (b)

Figure 2: *Examples of a set of plots assigned to 4 categories that define 6 patches*

distribution of plots among categories such that:

$$max\{f(S_{ij})\} \tag{1}$$

$$max\{g(C_{kj})\} \tag{2}$$

Where f and g are functions that aggregate all the values of S_{ij} and C_{kj} respectively in a single value.

The optimization is subjected to:

$$L_i \leq \sum_{j=1}^{N} A_j \delta_{ij} \leq U_i \qquad \forall i = 1, ..., M \tag{3}$$

Where; A_j is the area of the j-th plot, and $\delta_{ij} = 1$ if plot j is assigned to the i-th category, and $\delta_{ij} = 0$ otherwise. L_i and U_i are the lower and upper bounds for the total area assigned to i-th category.

This optimization stage is presented in detail in Section 2.2.

2.1. Initial conditions and preprocessing

The optimization problem requires three types of input data to be read initially: the characteristics of each plot, the parameters established for the

allocation of each category, and the geometric elements used to define fixed categories.

The characteristics of each plot include its geometry, area, list of neighbours, length of the borders with each current use, and a suitability score for each non-fixed category.

The parameters for the allocation include the maximum and minimum area for each category, L_i and U_i and the weights for the suitability and compactness criteria for each non-fixed category that are fixed by experts. These weights are used to tune the final results.

Finally, the elements that define the fixed categories correspond to layers of geometric elements like rivers, roads or archaeological sites, heritage and natural protection lands among others, that can delimit the fixed categories in three ways. First, by directly allocating these elements to a specific fixed category. Such a procedure also allows users to allocate a specific land category to areas that must be delimited. Second, by delimiting a buffer over the geometric elements defined by regulations at a certain distance established by law. This is the case for the legal protection areas for roads, for example. Third, by delimiting buffers at various distances according to an attribute of the geometric elements established by law. This is the case for water protection land in Spain, which is obtained by means of buffers of rivers at a maximum distance of 100 meters depending on the river category. Also, legal protection areas for roads define buffers at a distance of 7, 9, 12, 25, 30, 50 or 100 meters according to the type of road. The result of these procedures is a map of plots in which the fixed categories are delimited, so that the plots allocated to them are not considered in the optimization algorithm. Note that if a land plot is partially within the buffered area, the plot is divided into two new parcels in the preprocessing stage, in such a way that the new plot located inside the buffer is allocated to the fixed category. Figure 3 shows an example in which the buffers associated to a road and a river redefine the plots to be considered in the algorithm. For example, plot P51 is affected by the buffer defined by the road, plot P53 is affected by the buffer defined by the river, and P52 by both.

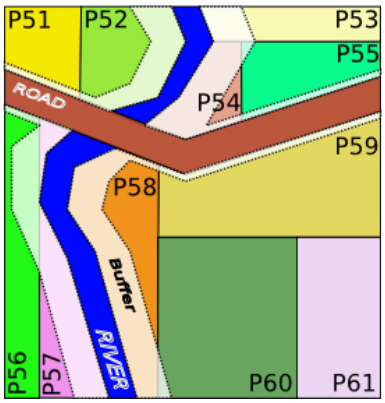


Figure 3: *Example of buffers defined by a river and a road*

A preprocessing stage performs all these calculations [18], most of which involve computationally expensive geometric operations (e.g. intersections) and the active interaction with the user. Such operations are executed just once. These operations were implemented using the JTS Topology Suite [28] library for spatial analysis operations and the Sextante framework [29].

Choosing the right data structure to store the neighbours and the length of the borderline is important insofar as this information is often accessed by the whole process, and it is important to minimize access time. As the number of neighbours in each plot may differ, two one-dimensional arrays are used to store the list of neighbours: an array of neighbours and an index array. The j-th entry of the index array stores the position in which the first neighbour of the j-th plot is stored in the array of neighbours. The neighbours of each plot are stored consecutively in the array of neighbours. Figure 4 shows an example of the two arrays. In this example, the neighbours of plot P2 are P1, P3, P11, P12 and P15, which are stored from position 4 to position 8 in the array of neighbours, respectively. Note that 4 is the value of the second entry in the index array. Additionally, information about the area and perimeter of each plot as well as the length of the border line between each pair of plots are stored using this

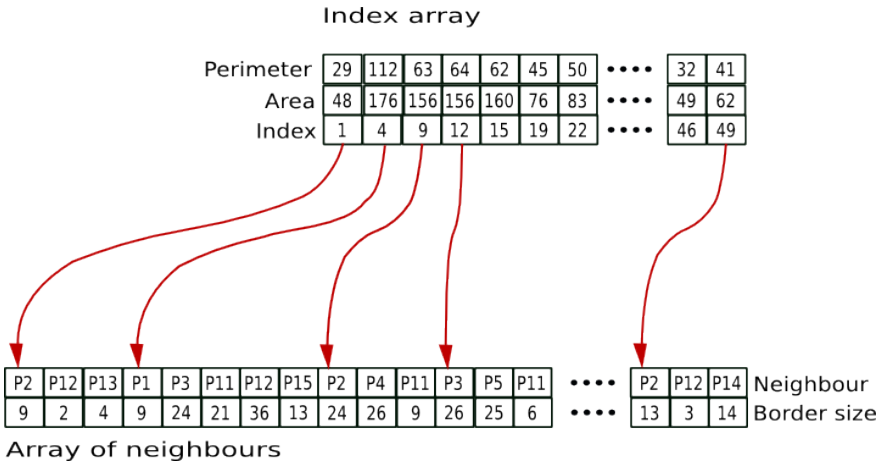


Figure 4: *Example of the arrays used to store the information about the neighbourhood*

data structure. Note that the size of the index and the area arrays is N, and the size of the neighbours and border lengths arrays are double than necessary because if P_i is stored as neighbour of P_j, then P_j is also stored as neighbour of P_i This storage scheme presents high locality and low latency in access when our code is used.

2.2. The optimization algorithm

The simulated annealing algorithm [30] is a heuristic for iteratively optimizing an objective or fitness function E ruled by a parameter termed temperature T that is used to control the thoroughness of the search for the optimum. The basic procedure consists of the following stages:

1. According to the stochastic nature of the simulated annealing algorithms, given the current configuration of the system, a trial configuration is generated by a method that includes some element of randomness.
2. The value of the objective function for the trial configuration, E_t, is compared with the value of the objective function for the current configuration, E_c. If E_t is better than E_c, the trial configuration is accepted as the current configuration, if E_t is worse than E_c, the trial configuration

is adopted as the next current configuration according to the Boltzmann probability distribution: $e^{(E_c - E_t)/T}$.

3. For each value of temperature, starting at a given initial value T_0, the system is allowed to explore the configuration space for a given number of iterations I. The value of T is then reduced, so that better E values are favored, and the loop starts again from step 1. Each reduction in T is determined by multiplying it by a constant factor.
4. The algorithm terminates upon satisfaction of some appropriate stop condition usually related with the values of I, T, and the number of updates of E_c.

In our case, an initial random solution that satisfies the constraints of maximum and minimum area for each category is generated at the beginning of the process. The trial solutions are generated by allocating a randomly selected plot to a new randomly selected category that satisfies the constraints. Large values of I minimize the effect of this random selection. Figure 5 shows the number of accepted new configurations as the temperature decreases in the algorithm for the case study introduced in section 3.

2.2.1. The objective function

As established above, the objective function combines two subobjectives: maximization of land suitability and maximization of compactness. These subobjectives are combined linearly:

$$E = w \cdot g(C_{kj}) + (1 - w) \cdot f(S_{ij}) \tag{4}$$

where w weighs compactness and $(1-w)$ weighs suitability. $0 \leq w \leq 1$. The subobjective functions are normalized to the range $[0, 1]$.

The relative importance of both suitability and compactness criteria can vary according to the target land category. For example, compactness is essential to urban land, whereas for natural spaces the importance of compactness is low. For this reason, planners must be able to quantify the relative importance of

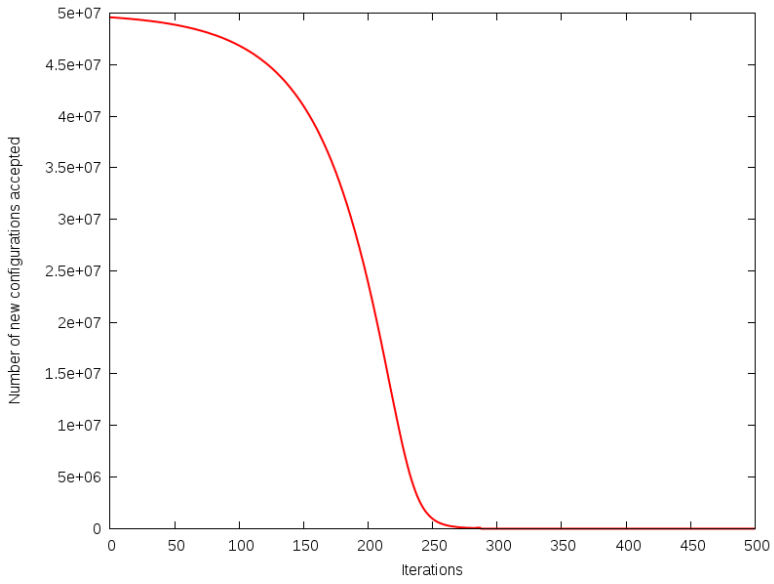


Figure 5: *Number of accepted new configurations for different updates in temperature for the case study introduced in section 3*

each objective in each land category. Actually, the possibility of defining such relative importance or weights is crucial for final users, insofar as the definition of weights will allow them to obtain a set of solutions for different scenarios and to compare them.

Suitability is calculated as the weighted average of the global suitability for each category. The global suitability for a category is obtained from the average of the suitability of the plots allocated to that category, weighted by the area of each plot and normalized by the total area assigned to the category, more formally:

$$Suitability = f = \sum_{i=1}^{M} \alpha_i \left(\frac{\sum_{j=1}^{N} S_{ij} A_j \delta_{ij}}{\sum_{j=1}^{N} A_j \delta_{ij}} \right) \tag{5}$$

where S_{ij} is established by experts, being $0 \leq S_{ij} \leq 1$, and α_i is the user-defined weight for the global suitability of the i-th category. Note that $\sum_{j=1}^{N} A_j \delta_{ij}$ is the area of the whole i-th category, and $0 \leq Suitability \leq 1$.

Compactness can be defined in different ways. Most of them are based on the metrics for each land category, including the number of patches for each land category, the largest patch for each land category [31] [6], or the number of neighbouring cells with the same category [4] [32]. More complex metrics are based on the relationship between area and perimeter for each land category patch for example, the average across all patches of the ratio of the number of perimeter cells to the total number of cells in the patch [6], or the average ratio of the perimeter divided by the square root of the patch area [31]. In our proposal, two different functions are considered: a function based on patches, and another function based on categories. The proposed compactness function based on patches is defined as:

$$Compactness_{patches} = g_p = 4\pi \sum_{i=1}^{M} \beta_i \left(\frac{\sum_{k=1}^{G} \delta_{ik}^* \frac{A_k^*}{(P_k^*)^2}}{\sum_{k=1}^{G} \delta_{ik}^*} \right) \tag{6}$$

Where β_i is the weight of the global compactness of the i-th category, A_k^* is the area of the k-th patch, and P_k^* is the length of its perimeter. $\delta_{ik}^* = 1$ if patch k is assigned to the i-th category, and $\delta_{ik}^* = 0$ otherwise. Note that $\sum_{k=1}^{G} \delta_{ik}^*$ is the number of patches of category i.

This proposal is based on the premise that, for any area, the so-called circularity is maximized by a circle (the maximum is 1) [33]. In the example of figures 2(a) and 4, circularity of plots P1, P2 and P3 are 0.717, 0.177 and 0.494 respectively, and circularity of patch {P1, P2, P3} is 0.251.

As an alternative, the compactness based on categories is also based on circularity:

$$Compactness_{categories} = g_c = 4\pi \sum_{i=1}^{M} \beta_i \left(\frac{\sum_{j=1}^{N} A_j \delta_{ij}}{\left(\sum_{k=1}^{G} P_k^* \delta_{ik}^* \right)^2} \right) \tag{7}$$

Note that $\sum_{k=1}^{G} P_k^* \delta_{ik}^*$ is the sum of perimeters of all the patches assigned to the i-th category. It can be computed as the sum of perimeters of plots assigned to the i-th category minus double the length of the part of its boundary shared with other plots allocated to the same category. In the example of figures 2(a)

Patch	Compactness	Category	Compactness
{P1, P2, P3}	0.251	{P1, P2, P3}	0.251
{P4, P5, P11}	0.442	{P4, P5, P11} {P13, P14}	0.219
{P7}	0.417	{P7} {P12, P15}	0.122
{P6, P8, P9, P10}	0.393	{P6, P8, P9, P10}	0.393
{P12, P15}	0.171		
{P13, P14}	0.282		
Global Compactness	**1.956**	**Global Compactness**	**0.985**

Table 1: *Compactness based on patches and categories for the example of figure 2(a)*

Patch	Compactness	Category	Compactness
{P1, P2, P3}	0.251	{P1, P2, P3}	0.251
{P4, P5}	0.725	{P4, P5} {P13, P14}	0.274
{P7}	0.417	{P7} {P12, P15}	0.122
{P6, P8, P9, P10, P11}	0.415	{P6, P8, P9, P10, P11}	0.415
{P12, P15}	0.171		
{P13, P14}	0.282		
Global Compactness	**2.261**	**Global Compactness**	**1.062**

Table 2: *Compactness based on patches and categories for the example of figure 2(b)*

and 4, its value for category {P1, P2, P3} is $\sum_{k=1}^{G} P_k^* \delta_{ik}^* = 29 + 112 + 63 - 2(9 + 24) = 138$. In this way, patches are not required to be identified, and the boundaries between the plots allocated to the same category do not decrease the value of the compactness function and the compactness of the patches is favored. Note that this function presents a lower computational cost than the one associated with equation (6) because it avoids the computation of patches. For categories composed by just one patch, both metrics are the same. Table 2.2.1 shows the compactness based on patches and categories for the example of figure 2(a).

2.2.2. The heuristic

The objective function for a trial solution E_t is computed by calculating the variation of E due to a change in the category of the randomly selected plot. This fact saves computing complexity because there is not a need for calculating the overall suitability and compactness for the whole plot map in each iteration. Figure 2(b) shows an example in which plot P11 changes the patch in one iteration of the algorithm. The new compactness is shown in table 2.

To apply the algorithm, a number of decisions have to be made. For ex-

ample, one must determine the cooling schedule, the initial temperature, the stopping condition, etc [34]. In particular, in order to minimize the influence of the random generation of the initial solution some experimentation should be performed by the user to select appropriate parameters. Note that our proposal is intended to be used in an interactive way. In any case, default calibration values for the parameters of the annealing schedule are given to the planner. In general, it is recommended that the initial value of T ensures that about 80% of trials are successful at this stage. This value was used for the case study in Section 3. As an example of the behaviour of our proposal, figure 6 shows the evolution of the objective function (figure 6(c)) as the temperature decreases for different values of T_0 for the case study in section 3. This example corresponds to a case in which $\omega = 0.50$, the multiplicative factor of temperature is 0.95, there are 50000 iterations for each temperature, and the algorithm finished after 100 iterations of temperature. Both components, suitability and compactness are also shown in figures 6(a) and 6(b) respectively. Note that the resulting values of the objective function do not differ much, and there is not a direct relationship between them and T_0. This is a typical behaviour we found for all the parameters. Note that the compactness changes in steps when a solution is found that differs much from the previous one. Figure 6(d) shows also the corresponding execution times of each iteration of temperature. Note the strong dependency of these times with the number of accepted trials, which in turn decrease as the temperature does. Low values of compactness are due to the typical shape of the plots in Galicia, which presents large length and small width. This is also the reason for the variation in compactness obtained for different initial temperatures. Note that there are large steps when some trial finds a more compact solution. This effect happens at temperatures between 5 and 20 when only trials that improve the compactness are likely accepted even though the suitability does not improve.

Based on exhaustive experiments, an initial temperature of 80 was selected when using the compactness function based on categories. Results for the compactness function based on patches allowed for the selection of $T_0 = 200$. Sim-

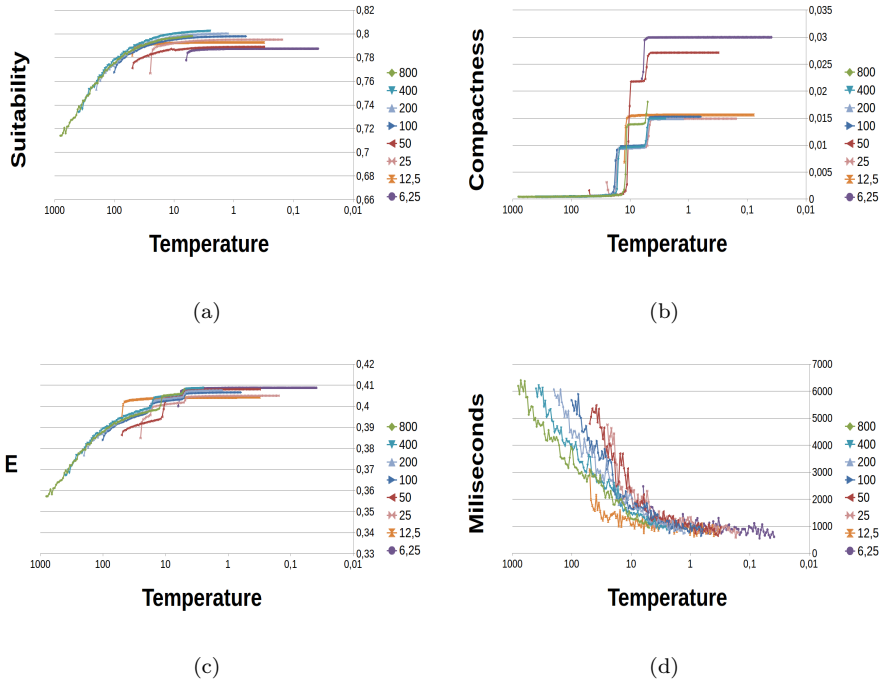


Figure 6: Suitability, compactness, objective function and sequential execution time for each temperature for the case study introduced in section 3

ilarly, the number of iterations executed at each temperature, that is, the heat balance condition, was set to 200000 after exhaustive trial and error studies. Each reduction in T was determined by multiplying it by a constant factor, which is 0.95 in our case of study. Note that the objective function improves fast in the initial iterations when the temperature is high, and then it achieves values that correspond to a local minimum. The stop condition of the algorithm was the number of temperatures visited, which was set to 200 in our case. Note that achieving the optimal case in not guaranteed by the heuristic. Therefore, in order to validate the quality of the achieved planning, it was compared with the solution provided by expert planners.

2.3. Parallel implementation

Nowadays, multicore systems bring about the need for new parallel algorithms and for the parallelization of existing solutions to compute intensive applications. Shared memory programming is the model that suits with this architecture. Therefore, OpenMp [35] was used to implement the parallel C codes. In our case, the computational cost of the algorithm is high because of the large number of plots and the implicit nature of the problem. Note that a practical use of our approach requires the interactive execution with different values of parameters tunned by the user. So, to get a more practical algorithm execution time has to be reduced, and the solution lies in parallelization. Execution times decrease with temperature, mainly because few changes are acknowledged by the algorithm.

Two strategies were initially considered to parallelize simulated annealing: the spatial parallelization and the parallelization of the computation of the objective function. The last one was found not efficient, mainly because of the low computational cost of the objective function since, according to our proposal, the computational cost of the determination of the change in E caused by category change in a single plot was dramatically reduced. However, spatial parallelization was found more productive. It is based on the idea that each thread run the algorithm in a particular geographic zone of the study area. Therefore, the plot map is partitioned into groups of plots that are completely surrounded by plots allocated to the fixed categories, i.e., by plots excluded from the simulation, so that there are no borderline interactions among these zones. Without losing generality, this definition can be relaxed if necessary, and large clusters can be split in smaller ones to optimize load balance. Each isolated group of plots is called a cluster. As an example, in figure 3, four clusters can be identified, P51 and P52 are in one of them, P53, P54 and P55 are in another, P56 and P57 in the third, and P58, P59, P60 and P61 in the last one. The algorithm identifies the clusters from the plot map in the preprocessing stage by using a flooding algorithm by a recursive search of neighbours [36]. In this way, the information used by each parallel thread is not shared by the others,

increasing the locality and affinity of the code.

One of the problems that most affects the performance of our OpenMP parallel code is computational load balance. In order to balance the computational load, the clusters were distributed among the different threads so that the number of plots in each thread was as balanced as possible. Based on the definitions of load metrics in parallel computers [37], for a given distribution of clusters, the balance can be defined as,

$$Balance = \frac{m_P}{M_P} \tag{8}$$

Where M_P is the maximum number of plots assigned to any thread, and m_P is the mean number of plots assigned to the threads. Note that $0 < Balance \leq 1$. $Balance$ is the inverse of the imbalance load [37] and it is the average efficiency across all processes over their maximum [38]. Note that it reflects the busyness degree. To balance the computational load, the clusters are distributed among the threads so that the number of plots is as similar as possible in every thread. A round robin load balancer was used to deal with this issue. If the resulting load balance is poor, that is if $Balance$ is lower than a certain threshold, new clusters can be defined by splitting large clusters.

The execution of each thread is practically independent from the rest of them. The only common information accessed by all the threads is the total area allocated to each category. Yet, this total area is constrained between a minimum and a maximum value. Therefore, changes in the category of a plot that result in a total area for a category exceeding the minimum and maximum values cannot be allowed. Accordingly, this constraint must be continuously checked by using parallel mutual exclusion operations. Some of them also need reduction operations that imply synchronization.

3. Case study

The municipality of Guitiriz, located in Galicia, an autonomous region of NW Spain, was considered as a case study. After the preprocessing stage, the

Category	Minimum area (ha)	Maximum area (ha)	Weight for suitability (α_i)	Weight for compactness (β_i)
Natural space	1900	2300	0.25	0.2
Urban	160	180	0.25	0.4
Agricultural	14000	17000	0.25	0.2
Forestry	5500	7500	0.25	0.2

Table 3: *Parameters assigned to non-fixed categories*

plot map of Guitiriz consisted of 139391 plots, 84216 of which did not have a fixed category, which must be considered in the simulated annealing stage. In the case of Galicia, the fixed categories include water, coast, infrastructure and heritage protection land, whereas the non-fixed categories correspond to agricultural, forestry, natural space and urban land. The suitability for each category was obtained from previous studies [39] and the weights of compactness and suitability for each category were established based on the experience and judgment of planners (see table 3). The range of total area for agriculture and forestry was obtained from the current area of agricultural (15220 ha) and forestry (6319 ha) land, allowing a variation of ±10% and rounding the resulting values, with the exception of the allowable increase for the forestry category, which was set at 20% due to the interest of owners in the afforestation of agricultural land. The area that had to be allocated compulsorily to the natural space category was 1563 ha. However, given the general concern among politicians and society, a minimum increase of 20% and a maximum increase of 50% were established. The current urban area is 76 ha, but planners have estimated a requirement of 172 ha based on the increase in population and the permitted urban density. A variation of only 5% from 172 ha was established for the urban category due to legal requirements.

All performance tests were executed in a system with 2 Intel Xeon E5440 2.83GHz processors with 4 cores each and 16 GBs of shared memory running a Ubuntu 12 distribution.

3.1. Results

Various solutions were generated by using compactness functions based on categories and patches for different combinations of the subobjective weights w

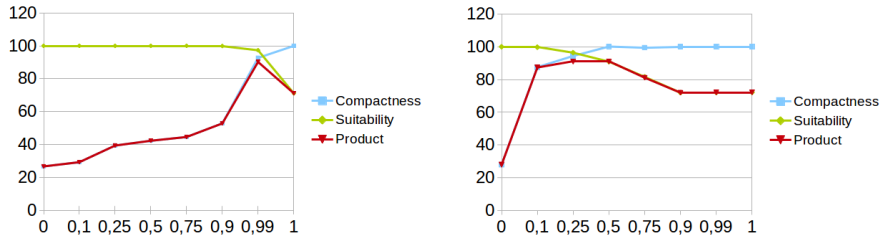


Figure 7: *Fitness in % for compactness, suitability, and their product using the compactness function based on categories (a) and on patches (b)*

and $(1 - w)$. Figure 7(a) shows the fitness achieved for some of these combinations when the compactness function based on categories was used. In the figure, fitness values are expressed as the percentage over the maximum value for that subobjective, obtained when the subobjective was assigned a weight of $w = 1$ for suitability, and $w = 0$ for compactness.

Figure 7(a) shows a strong dependence of fitness values on weights. However, such a dependence was not linear or equal for both subobjectives. The value of the compactness subobjective increased noticeably as its weight increased. Actually, when the compactness subobjective was assigned the maximum weight, the compactness score was 70% higher than when a null weight was assigned to this subobjective. The amount of increase was smaller when w is low, i. e., there was a difference of 15% in the fitness value between not considering compactness and assigning compactness half the weight (0.50), but the difference was much higher for weights above 0.9. The effect of the weight assigned to the suitability subobjective was lower, with a difference between the case of $w = 1$ and the case of $w = 0$ below 30%. Besides, the maximum fitness value was obtained from a weight of $w = 0.9$. Weight of $w = 0.99$ was chosen because a significant increase in compactness was achieved with a barely noticeable decrease in suitability. This case presents the maximum product in figure 7(a).

Figure 7(b) shows the results when the compactness function is based on

patches. Note that a somewhat different behaviour of the objective function for different combinations of weights w and $(1-w)$ is obtained as compared to the behaviour described above for the compactness function based on patches. The general influence of these weights was still stronger for the compactness subobjective than for the suitability subobjective. The fitness value obtained when the compactness subobjective was assigned the maximum weight was more than 70% higher than the value obtained when compactness was assigned a null weight, whereas the difference in the fitness values obtained for the suitability subobjective amounted to less than 30%. The same behaviour was observed for the compactness function based on categories. However, the variation in the fitness values of both subobjectives for intermediate weights differed. In this case, the compactness value increased especially at the beginning. Thus, the achievement levels obtained for the compactness subobjective were 95% for $w = 0.25$ and 100% for $w = 0.5$. The increase in suitability with weight was more linear. The amount of increase was greater at the beginning, until a weight of 0.50 was reached. The case $w = 0.50$ was chosen for parallelization tests because it provided the maximum value for the compactness subobjective and an achievement level of 90% for the suitability subobjective. In fact, this is the situation in which the product of compactness and suitability is maximum.

The maps included in Figure 8, especially the zoomed window, show the increase in compactness with the increase in the weight assigned to compactness. The zoomed window corresponds to an area in which the difference between the cases of $w = 1$ and $w = 0.99$ is clear, particularly for the urban land category. This area has low suitability for urban use because of its location far from the two big settlements of the municipality and, consequently, from any urban infrastructures. The map for $w = 1$ (Figure 8(c)) shows more compact patches than the maps obtained with weight $w = 0$ and $w = 0.5$ (Figures 8(a) and 8(b), respectively). The urban land category was allocated to this area because suitability was not considered. In contrast, in the map obtained using the weight $w = 0.99$ (Figure 8(d)), compactness remained high but urban land was not allocated to this area because of its low suitability for urban use.

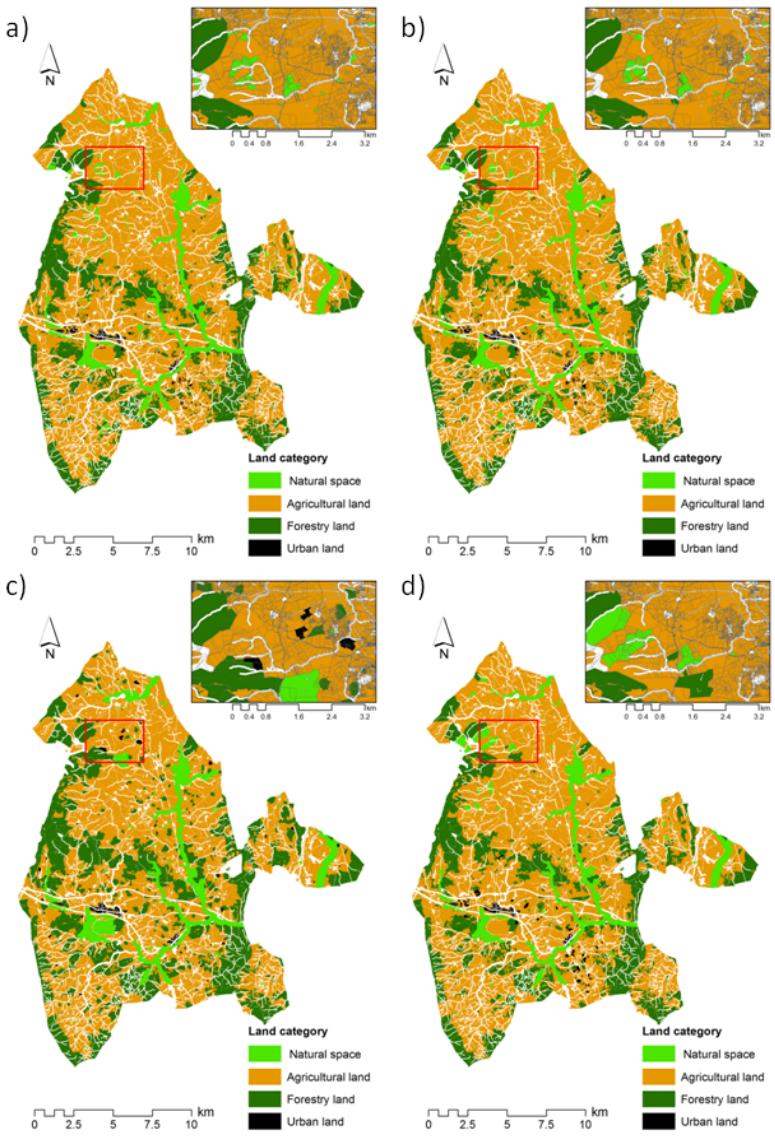


Figure 8: *Maps obtained with the optimization algorithm using the compactness function based on categories for weights a)* $w = 0$*, b)* $w = 0.5$*, c)* $w = 1$*, and d)* $w = 0.99$

The visual comparison of the results from the compactness function based on categories (Figure 8) and those obtained from the compactness function based

on patches (Figure 9) reveals that the compactness function based on patches generates a higher number of smaller patches. It avoids the allocation of isolated plots to a land category, but forms compact small patches that approximate the circular shape. In fact, when $w = 1$, the average area of the 9370 patches obtained when the compactness function is based on categories is 3.13 ha, and 1.23 ha for the 23803 patches obtained when the compactness function is based on patches.

3.2. Efficiency of the parallel implementation

The strategy described in section 2.3 to balance the workload was used to distribute clusters among threads. In particular, our case study presents 84216 plots, and the algorithm achieved a difference of no more than two plots in the number of plots allocated to each thread, except when the number of them was seven, in which case $Balance = 0.903$. Such an exception was caused by the presence of an exceptionally large cluster composed of 13321 plots. Given the size of this cluster, using more than seven threads to balance the load did not make sense when using this approach in this particular situation. As mentioned in section 2.3, when the number of cores causes low performance because of the load imbalance, the user can divide large clusters into smaller ones accordingly.

The speed-up of a parallel code is defined as the ratio to measure how much a parallel algorithm is faster than the sequential counterpart, i.e. sequential execution time divided by parallel execution time. As expected, the speed-up, defined as the ratio to measure how much a parallel algorithm is faster than a corresponding sequential algorithm, obtained with the compactness function based on categories and weight $w = 0.99$ (Figure 10(a)) showed a fairly linear behaviour. The sequential execution times per iteration in the temperature loop are between 24.5 seconds for the highest temperature and 5 seconds for the lowest one. The specific times depend primarily on the number of accepted solutions. The whole program needs about 1910 seconds to run. These execution times are too high for an interactive use of the algorithm. The speed-up obtained with the compactness function based on patches and weight $w = 0.5$

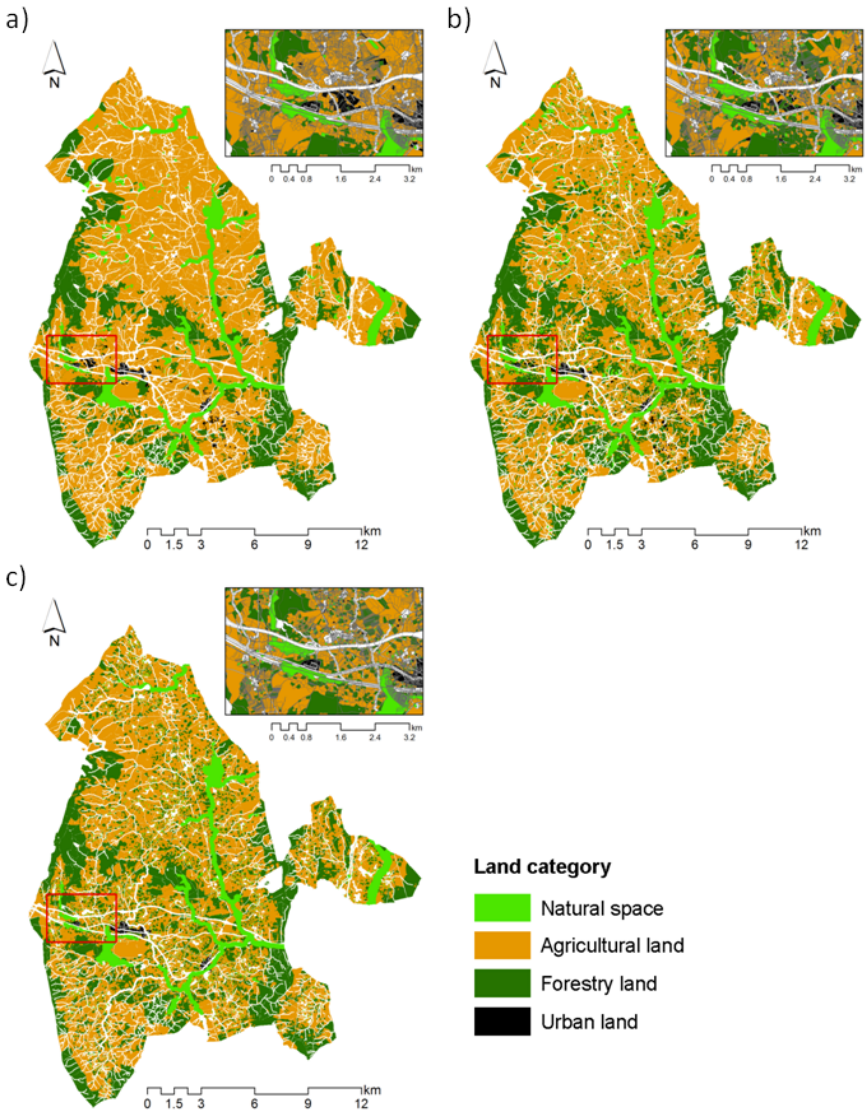


Figure 9: *Maps obtained with the optimization algorithm using the compactness function based on patches for weights a)* $w = 0$*, b)* $w = 0.5$*, and c)* $w = 1$

(Figure 10(c)) was exceptionally high. The reason for this behaviour lies in the structure and management of the storage of data when patches are used. Such a

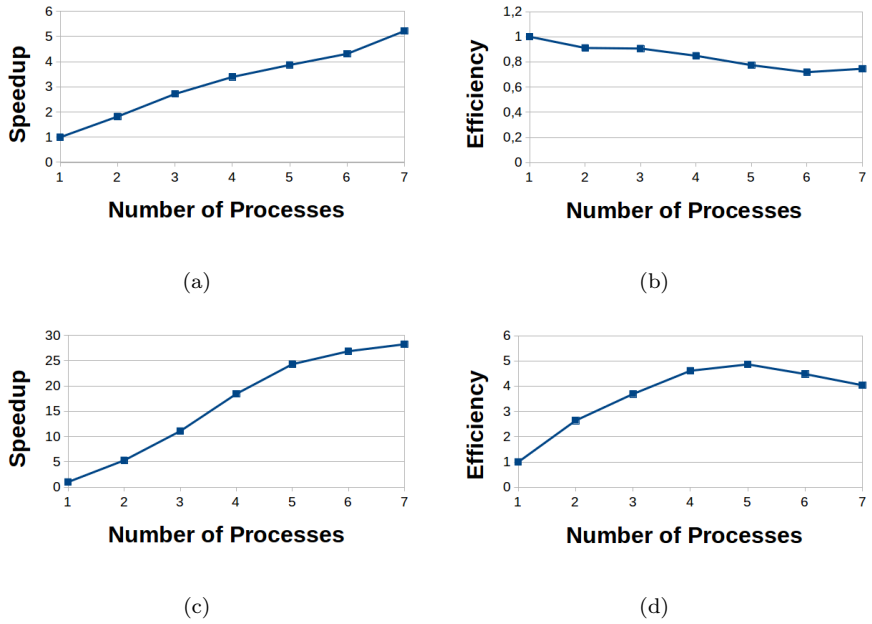


Figure 10: *Speed-up and efficiency by using the compactness function based on categories and* $w = 0.99$ *(a) and (b) and* $w = 0.5$ *(c) and (d) respectively*

superlinear speed-up can be attributed to cache effects. In parallel computing, not only does the number of processors change, but also the size of accumulated caches from different processors [40]. With increased accumulated cache size, more, or even all of the core data sets can fit into caches and memory access time decreases dramatically, which causes an extra speed-up in addition to that from the actual computation. In our particular case, the use of the hash table with more than 80000 entries, each 56 bytes long, is much larger than the L1 cache, and almost over-fits the L2 cache of the multicore system. These results outperform the Java-based message passing implementation of genetic algorithms presented in [19]. Figures 10(b) and 10(d) show the same results in terms of efficiency, that is the number of threads divided by the speedup.

		Solution by planners				
		A	F	NS	U	RS
Algorithm	A	**13504.2**	2340.4	244.4	20.2	228.4
	F	1083.2	**4232.7**	23.0	29.9	131.1
	NS	296.4	33.8	**1562.3**	3.9	3.6
	U	90.2	15.9	0.0	**64.7**	9.3

Table 4: *Coincidence matrix (in ha) between the land zoning map designed by technicians and the ones produced by the algorithm with compactness based on categories*

3.3. Comparison with expert's scenario

In order to evaluate the solutions provided by the algorithm, these solutions have been compared to the land use zoning map designed by planners (Figure 11). Planners delineated the land categories of this map without any support from algorithms or scientific methods and using the same suitability maps. Table 4 shows the overlapping area for each land category between the map designed by planners and the map obtained with the optimization algorithm using the compactness function based on categories and $w = 0.99$. In this table, A means agricultural category, F refers to forestry category, NS to natural space category and U to urban category. Finally, RS means rural settlement category. This category is not included in the algorithm because it is specific to the study area and is delimited by legal restrictions. These results show good matches for the agricultural, forestry and natural space categories, insofar as over 82% of the area allocated by the algorithm to these categories was allocated to the same category by planners. The causes of the worst match for the urban category (36%) were the aesthetic and architectural criteria used by technicians in urban planning, which were not considered in the algorithm. When using the compactness function based on patches and $w_c = 0.5$, the overlapping area between the map designed by planners and the map obtained with the algorithm (Table 5) for forestry, natural space and urban was more than 76%. The decrease in the quality of matches was substantial for the forestry and natural space categories, decreasing to 60% in the forestry category and to 69% in the natural space category.

Table 6 shows the comparison of the values of the subobjective functions

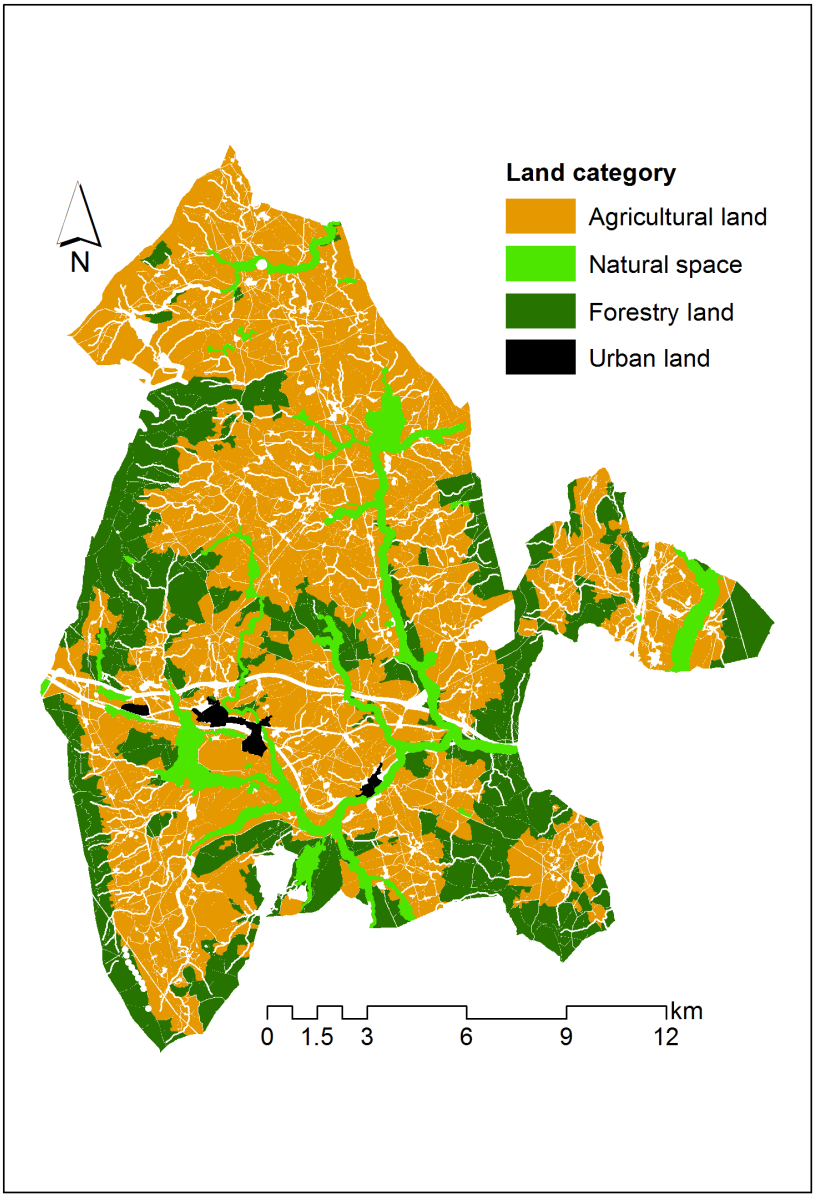


Figure 11: *Land use zoning map designed by planners*

		Solution by planners				
		A	F	NS	U	RS
Algorithm	A	**11682.8**	1892.8	184.7	41.4	198.3
	F	2538.0	**4665.4**	67.2	11.5	155.5
	NS	653.2	56.2	**1577.0**	5.0	8.6
	U	99.9	8.4	0.8	**60.8**	10.1

Table 5: *Coincidence matrix (in ha) between the land zoning map designed by technicians and the ones produced by the algorithm with compactness based on patches*

for the maps obtained with the algorithm and the map designed by planners. In this table, for the algorithm based on categories and $w = 0.99$, whereas for the algorithm based on patches and $w = 0.5$. According to the table, the value of the suitability subobjective is always higher in the algorithmic solutions. Considering that the percentage of suitability was calculated over the maximum value of suitability obtained with the algorithm, the solution of planners was very good for this subobjective. However, this suitability was overcome by the algorithm for weights w lower than 0.25. The comparison of the values of the compactness subobjective reveals that compactness based on patches provides a kind of compactness that is not sought by technicians. In contrast, the map generated by the algorithm when compactness was measured using the function based on categories was similar to the map designed by planners in terms of compactness. Note that the quantification of fitness can be very useful for planners.

In addition, as the compactness metrics used in the objective function are expected to perform better in the algorithm solution, another two spatial metrics were used for the evaluation: contagion index and area-weighted shape index [41]. The contagion index approaches 0 when the patches of the land categories are maximally disaggregated and interspersed and 100 when they are maximally aggregated. The shape index approaches 1 for a square patch and higher scores correspond to more irregular patches. The values of both metrics (shown in Table 6) confirm, firstly that the solution of planners is the most compact one, with values of compactness metrics very similar to those of the algorithm solution using the compactness based on categories and secondly that the compactness

	Algorithm with compactness based on categories	Algorithm with compactness based on patches	Map of planners
Suitability	0.630643	0.582093	0.569660
Suitability (%)	97%	90%	88%
Compactness based on categories	0.002580		0.002536
Compactness based on categories (%)	93%		91%
Compactness based on patches		0.643846	0.028670
Compactness based on patches (%)		100%	4%
Shape index	4.62	4.94	4.49
Contagion index	61.9	58.9	62.5

Table 6: *Values of the subobjective functions for the maps obtained with the algorithm and the map designed by planners*

based on patches generates more disperse patterns.

Results show that the solutions provided by the algorithm improve the overall suitability achieved in the allocation of plots to land categories in comparison with the planners solution. In terms of compactness, the best solution corresponds to the planners map but the output provided by the compactness function based on categories achieves very similar compactness values.

The optimization results also confirm a notable influence of weights on the final configuration of the land use map. As demonstrated, the values of compactness and suitability weights can be easily tuned by trial and error, and the weights for land categories can be clearly established by the planners, while this allocation can be facilitated by means of techniques such as the Analytical Hierarchy Process [42]. A more complex task for planners is the identification of the optimal values for the annealing schedule. These parameters were tuned for the case study and the resulting values are the default settings shown to the user. However, an adaptive method for tuning parameters automatically would improve the outcomes. Finally, the cadastral map configuration also affects the parallelization performance, since the largest cluster determines the maximum number of threads, and thus sets a limit to the speed-up through parallelization.

4. Conclusions

In this paper, we have dealt with the problem of land use spatial allocation. Our proposal is to solve the problem of land category delimitation, which frequently becomes a bottleneck for the land use planning process. A preprocessing algorithm delineates the fixed land categories according to legal and expert criteria. After a preprocessing stage, a new simulated annealing based heuristic is used to efficiently allocate the non-fixed categories. This work generalizes a previous one, and it is also complementary in many aspects.

In the spatial parallel implementation proposed, the geographical area of study is partitioned into a number of so-called clusters that can be processed in parallel. Appropriate mechanisms for sharing the information among the threads have been taken into account. Nowadays, the use of parallel solutions to most applications is justified by the presence of multicore processors in the market. The efficiency of parallel implementation was validated in the case study.

The quality of the results for real situations is comparable to the quality of the results obtained by experts. However, the main advantage of the algorithm does not lie in the increase in the values of the objective function but in the possibility of generating a land use map based on a justified, scientific and transparent procedure in a short time. This possibility, in turn, allows for the generation of a number of alternative solutions by modifying the parameters involved in the algorithm, such as the subobjective weights, the suitability and compactness weights for each land category, the areas of each land category, or even the suitability of maps. Accordingly, the final objective of the algorithm is not to provide the optimal solution but to facilitate and justify the design of the final solution by planners or other stakeholders. The plethora of solutions that can be obtained from different executions of the proposed algorithm provide an important source of invaluable information for users. Such information is particularly important because the quality of these solutions is quantified in terms of fitness.

As future work, one of the most immediate improvements is to study other types of functions for the evaluation of compactness criteria. New definitions of compactness that match the desired spatial distribution of patches of each land category would improve the quality of the results. In addition, considering other spatial criteria such as connectivity can be useful, especially in the case of the natural space category, in which connectivity could be used to design ecological networks. Finally, we plan to adapt the previous genetic algorithm to the conditions of this work to combine both approaches. In addition we will consider the implementation of a hybrid parallel solution that combines message passing and shared memory paradigms.

References

[1] T. A. Arentze, A. W. Borgers, L. Ma, H. J. Timmermans, An agent-based heuristic method for generating land-use plans in urban planning, Environ Plann B 37 (2010) 463–482.

[2] R. G. Cromley, D. M. Hanink, Scale-independent land-use allocation modeling in raster gis, Cartogr Geogr Inf Sci 30 (2003) 343–350.

[3] J. R. Eastman, W. Jin, P. A. Kyem, J. Toledano, Raster procedures for multi-criteria/multi-objective decisions, Photogramm Eng Rem S 61 (1995) 539–547.

[4] J. C. Aerts, E. Eisinger, G. B. Heuvelink, T. Stewart, Using linear integer programming for multi-site land-use allocation, Geogr Anal 35 (2003) 148–169.

[5] J. D. Duh, D. G. Brown, Knowledge-informed pareto simulated annealing for multi-objective spatial allocation, Comput Environ Urban 31 (2007) 253–281.

[6] R. Janssen, M. van Herwijnen, T. J. Stewart, J. C. Aerts, Multiobjective decision support for land-use planning, Environ Plann B 35 (2008) 740–756.

[7] D. J. Nalle, J. L. Arthur, J. Sessions, Designing compact and contiguous reserve networks with a hybrid heuristic algorithm, Forest Sci 48 (2002) 59–68.

[8] T. J. Stewart, R. Janssen, M. van Herwijnen, A genetic algorithm approach to multiobjective land use planning, Comput Oper Res 31 (2004) 2293–2313.

[9] K. B. Matthews, S. Craw, A. R. Sibbald, Implementation of a spatial decision support system for rural land use planning: integrating gis and environmental models with search and optimisation algorithms, Comput Electron Agr 23 (1999) 9–26.

[10] J. C. Aerts, M. van Herwijnen, T. J. Stewart, Using simulated annealing and spatial goal programming for solving a multi site land use allocation problem, Lecture Notes in Computer Science 2632 (2003) 448–463.

[11] J. C. Aerts, G. B. Heuvelink, Using simulated annealing for resource allocation, Int J Geogr Inf Sci 16 (2002) 571–587.

[12] M. Boyland, J. Nelson, F. L. Bunnell, Creating land allocation zones for forest management: a simulated annealing approach, Can J Forest Res 34 (2004) 1669–1682.

[13] E. Martnez-Falero, I. Trueba, A. Cazorla, J. L. Alier, Optimization of spatial allocation of agricultural activities, J Agr Eng Res 69 (1998) 1–13.

[14] I. Sante, M. Boullen, R. Crecente, D. Miranda, Algorithm based on simulated annealing for land-use allocation, Comput Geosci-UK 34 (2008) 259–268.

[15] S. K. Sharma, B. G. Lees, A comparison of simulated annealing and gis based mola for solving the problem of multi-objective land use assessment and allocation, in: The 17th International Conference on Multiple Criteria Decision Analysis, 2004.

[16] M. E. Watts, I. R. Ball, R. R. Stewart, C. J. Klein, K. Wilson, C. Steinback, R. Lourival, L. Kircher, H. P. Possingham, Marxan with zones: software for optimal conservation based land- and sea- use zoning, Environ Modell Softw 24 (2009) 1513–1521.

[17] D. Stevens, S. Dragicevic, K. Rothley, icity: A gis-ca modelling tool for urban planning and decision making, Environ Modell Softw 22 (2007) 761–773.

[18] M. Suarez, I. Sante, F. Rivera, R. Crecente, M. Boullon, J. Porta, J. Parapar, R. Doallo, A parallel algorithm based on simulated annealing for land use zoning plans, in: 2011 International Conference on Parallel and Distributed Processing Techniques and Applications, 2011, pp. 360–366.

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

[20] A. Celmatis, M. Mineter, R. Marciano, High performance computing with geographical data, Parallel Comput 29 (2003) 1275–1279.

[21] M. J. Mineter, S. Dowers, Parallel processing for geographical applications: A layered approach, J Geogr Syst 1 (1999) 61–74.

[22] X. Li, Emergence of bottom-up models as a tool for landscape simulation and planning, andscape Urban Plan 100 (2011) 393–395.

[23] Z. J. Czech, A parallel simulated annealing algorithm as a tool for fitness landscapes exploration, Parallel and Distributed Computing. (2010) 247–271.

[24] E. Onbasoglu, L. Ozdamar, Parallel simulated annealing algorithms in global optimization, J Global Optim 19 (2001) 27–50.

[25] Y. Liu, S. Wang, A scalable parallel genetic algorithm for the generalized assignment problem, Parallel computing 46 (1015) 98–119.

[26] Y. Ding, P. J. Densham, Spatial strategies for parallel spatial modelling., Int J Geogr Inf Sci 10 (1996) 669–698.

[27] S. Schiele, M. Moller, H. Blaar, D. Thurkow, M. Muller-Hannemann, Parallelization strategies to deal with non-localities in the calculation of regional land-surface parameters, Comput Geosci-UK 44 (2012) 1–9.

[28] M. Davis, J. Aquino, JTS Topology Suite. Technical Specifications. Vivid Solutions, 2003. URL: `http://www.vividsolutions.com/jts/jtshome.htm`.

[29] V. Olaya, SEXTANTE Programming Guide, 2009. URL: `http://geostat-course.org/system/files/ProgrammingGuide.pdf`.

[30] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equation of state calculations by fast computing machines, J Chem Phys 21 (1953) 1087–1092.

[31] J. C. Aerts, M. van Herwijnen, R. Janssen, T. J. Stewart, Evaluating spatial design techniques for solving land-use allocation problems, J Environ Plann Man 48 (2005) 21–142.

[32] C. Kai, H. Bo, Z. Qing, W. Shengxiao, Land use allocation optimization towards sustainable development based on genetic algorithm, in: 17th International Conference on Geoinformatics, 2009. Doi 10.1109/GEOINFORMATICS.2009.5292899.

[33] R. S. Montero, E. Bribiesca, State of the art of compactness and circularity measures, in: Int Math Forum, 27, 2009, pp. 1305–1335.

[34] P. Moon-Won, K. Yeong-Dae, A systematic procedure for setting parameters in simulated annealing algorithms, Computers Ops Res 25 (1998) 207–217.

[35] B. Chapman, G. Jost, R. V. D. Pas, Using OpenMP: Portable Shared Memory Parallel Programming, MIT Press, USA, 2007.

[36] L. Vincent, P. Soille, Watersheds in digital spaces: an efficient algorithm based on immersion simulations, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (1991) 583–598.

[37] O. Pearce, T. Gamblin, B. de Supinski, M. Schulz, N. Amato, Quantifying the effectiveness of load balance algorithms, in: 26th ACM international conference on Supercomputing, 2012, pp. 185–194.

[38] K. Huck, J. Labarta, Detailed load balance analysis of large scale parallel applications, in: 39th International Conference on Parallel Processing, 2010, pp. 535–544.

[39] I. Sante, R. Crecente, M. Boullon, D. Miranda, in: In: Geneletti D, Abdullah A (eds.) Spatial Decision Support for Urban and Environmental Planning. A Collection of Case Studies, Arah Pub., 2009, pp. 33–60.

[40] D. P. Helmbold, C. E. McDowell, Modeling speedup(n) greater than n, IEEE Trans Parallel Distrib Syst 1 (1999) 250–256.

[41] K. McGarigal, S. A. Cushman, M. C. Neel, E. Ene, FRAGSTATS: Spatial Pattern Analysis Program for Categorical Maps, University of Massachusetts, 2002. URL: `http://www.umass.edu/landeco/research/fragstats/fragstats.html`.

[42] T. L. Saaty, The Analytic Hierarchy Process, McGraw-Hill, Upper Saddle River, NJ, USA, 1980.