A Java-based Parallel Genetic Algorithm for the Land Use Planning Problem

Juan Porta, Jorge Parapar, Guillermo L. Taboada, Ramón Doallo
Computer Architecture Group
University of A Coruña, Spain
{juan.porta,jparaparl, taboada,doallo}@udc.es

Francisco F. Rivera
Computer Architecture Group
University of Santiago de Compostela, Spain
ff.rivera@usc.es

Inés Santé, Marcos Suárez, Marcos Boullón, Rafael Crecente
Land Laboratory
University of Santiago de Compostela. Lugo, Spain
{ines.sante,marcos.suarez, marcos.boullon, rafael.crecente}@usc.es

ABSTRACT

In this work, the application of genetic algorithms to the elaboration of land use plans is studied. These plans follow the national legal rules and experts' considerations. Two optimization criteria are applied: aptitude and compactness. As the number of affected plots can be large and, consequently, the execution time of the algorithm can be potentially high, the work is focused on the implementation and analysis of different parallel paradigms: multi-core parallelism, cluster parallelism and the combination of both.

Categories and Subject Descriptors

D.1.3 [**Concurrent Programming**]: Concurrent Programming—*distributed programming, parallel programming*

General Terms

Algorithms, Performance

Keywords

Land use planning, genetic algorithms, parallel programming, distributed programming, MPJ

1. INTRODUCTION

This work try to solve municipal land use planing problems, which can be formulated in terms of an optimization problem in which each piece of land is allocated to the best category according to experts' criteria and law restrictions.

Defining land use patches as the polygons resulting from the union of plots allocated to the same category, two aspects to be optimized are considered: the shape-regularity of the resultant land use patches (compactness), and the land suitability that each plot has for the category allocated to it (aptitude). Also, the process of spatial allocation should take into account the current boundaries of the existing plots.

Therefore, the objective is to maximize these criteria for the search space defined by all the possible combinations of plots to categories, and subjected to given constraints. Genetic algorithms (GA) are clearly one of the candidate methods to solve this optimization problem[2], and they are the ones that are studied in this work. In this scenario, each individual gene represents a label that identifies the possible category allocated to the corresponding plot. So, individuals consist of as many genes as the number of considered plots.

Our case of study is in Galicia, an autonomous region at the Northwest of Spain characterized by the high fragmentation of the land ownership. Thus, the execution of the GA involves a high computational costs.

GECCO'11, July 12–16, 2011, Dublin, Ireland.
ACM 978-1-4503-0690-4/11/07.

2. GENETIC ALGORITHM FOR LAND USE PLANNING

Valid initial individuals are randomly generated. Valid means that they must satisfy the restrictions imposed in the problem. We use the roulette-wheel technique for the selection stage and a two-point crossover with random point-selection was implemented. Users can set the probability with which crossover and mutations are applied.

In every iteration of the genetic loop two individuals are chosen using elitism, so parents and children can be selected to be part of the new population. To validate the quality of the individuals, a fitness function with two criteria to optimize is considered: aptitude, and compactness. Both components are weighted to allow the user to establish the importance of one against the other.

The used compactness function is:

$$Compactness = 4\Pi \sum_{i=1}^{C} w_i \left(\frac{\sum_{j=1}^{N_i} a_{ij}}{\sum_{j=1}^{N_i} p_{ij}^2} \right) \qquad (1)$$

where C is the number of categories, w_i is the weight of the *i-th* category, N_i is the number of plots allocated to the *i-th* category, a_{ij} is the area of the *j-th* plot allocated to the *i-th* category, and p_{ij} is its perimeter.

2.1 Parallel Genetic Algorithm

Three levels of parallelism are considered in terms of the target architecture where the application can be deployed: for multi-cores, for clusters and a hybrid version.

When multi-cores are the target architecture, we focus on the parallelization of the genetic loop of the algorithm, that is, the process of generating new individuals through multiple threads. Each thread generates a set of individuals by

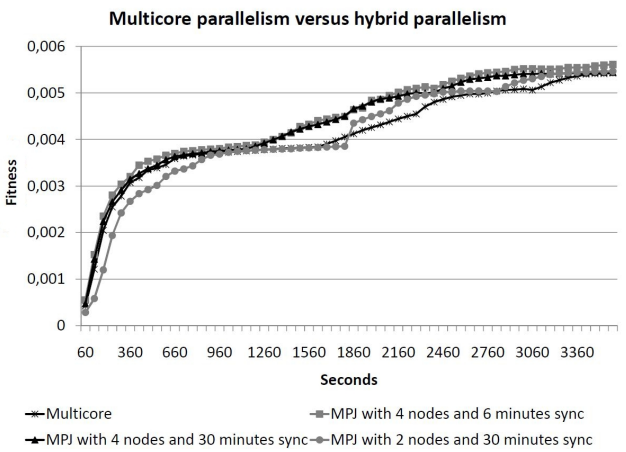

Figure 1: Fitness value for the multi-core version against the hybrid version.

performing the selection, the crossover, the mutations and the replacement operation. As threads share the memory, all of them can read the same population and they can write the next population. With this approach, best individuals are obtained in less time than the sequential one.

As clusters are distributed memory systems, the parallel algorithm has to be able to send and receive messages among processes for the information exchange. Each process runs in a node of the cluster, and a master process is in charge of establishing the communications with the slave ones. The idea is that each slave process executes the whole GA independently from the rest of slaves, using its own population. A synchronization with the master process is performed periodically, sending the best individuals from the slaves to the master. Then, gathering this information, the master generates a new population. This version takes advantage of the fact that each slave process generates its own population at each iteration of the genetic loop, and it executes the whole algorithm until the synchronization occurs. After synchronization, every slave receives the same population but it will evolve in its own way because of the randomness of the genetic operators. In addition, the best individuals obtained from the execution of different slaves join into the new population raising the average fitness and increasing the search space. For the implementation of this message passing paradigm, we use Java Message Passing (MPJ) [1].

As the two previous approaches are orthogonal in the sense that they can be mixed, a hybrid version was implemented to produce a new algorithm that can be executed in a cluster with multi-core nodes taking advantage from both parallel versions.

2.2 Experimental Results

In our case of study, the individuals have 40,498 genes and 4 categories to assign them. Figure 1 shows the fitness of the hybrid version versus the multi-core version. For these tests, 8 threads per node were used (in nodes with two quad-core processors) for both versions of the algorithm and, in the case of the hybrid version, they were executed in two and four nodes. All the executions using the hybrid version provided better results than the execution using the multi-core version. Focusing on the executions of the hybrid

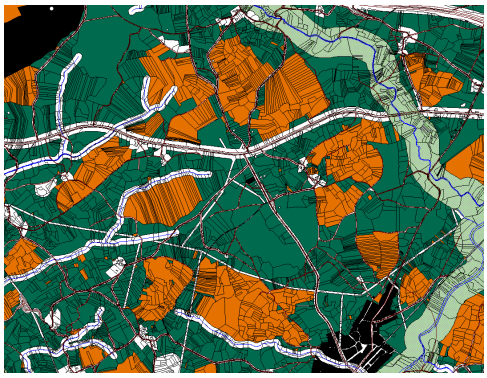

Figure 2: Zoom of an output of the algorithm.

algorithm, better individuals are obtained as the number of nodes increases. However, the increment of fitness is not as remarkable as the one obtained in the multi-core version with respect to the sequential one. Nevertheless, the search space is explored more deeply and the results are usually better. Finally, according to Figure 1, we can conclude that better results are obtained when frequency of the synchronizations is increases (6 and 30 minutes frequency charts are shown). Testing only the multicore version, an almost linear speed-up is obtained. Figure 2 shows a solution provided by the algorithm executed in hybrid mode and only with the compactness. As can be denoted by the different colours, patches of plots form regular polygons as it is expected. White patches correspond to fixed categories, like riversides, roads, or urban areas. Other patches include the rest of the plots, fixed and not-fixed.

3. CONCLUSIONS AND FUTURE WORK

This work shows that parallel GAs are a good choice to deal with land use planning problems with a significant number of plots. Regarding the parameters of the algorithm, big population sizes are the best option in these problems that demand large execution time. With respect to the mutation, we propose an adaptive mutation rate with high values in the first iterations, to avoid local optimums, and decreasing it over the time to refine the search modifying only a few genes. Finally, the development of efficient parallel Java code has demonstrated to be a competitive solution: programmability issues are clearly in favour of Java while acceptable performance has been obtained.

Acknowledgements.

This work has been funded by the Galician Regional Government, Spain. Projects: *08SIN011291PR*, and *Consolidation of Competitive Research Groups* (ref. 2010/06)

4. REFERENCES

[1] B. Carpenter, V. Getov, G. Judd, A. Skjellum, and G. Fox. MPJ: MPI-like message passing for Java. *Concurrency - Practice and Experience*, 12(11):1019–1038, 2000.

[2] K. Matthews, S. Craw, I. MacKenzie, S. Elder, and A. Sibbald. Applying genetic algorithms to land use planning. In *Proceedings of the 18th Workshop of the UK Planning and Scheduling Special Interest Group*, pages 109–115. University of Salford, December 1999.