OpenMP Implementation in the Characterization of an Urban Growth Model Cellular Automaton

Alvaro Peraza Garzón
Instituto Tecnológico de Mazatlán
Universidad Autónoma de Sinaloa
Sinaloa, México

René Rodríguez Zamora
Universidad Autónoma de Sinaloa
Instituto Tecnológico de Mazatlán
Sinaloa, México

Wenseslao Plata Rocha
Facultad de Ciencias de la Tierra y el Espacio
Universidad Autónoma de Sinaloa
Sinaloa, México

***Abstract*—This paper presents the implementation of a parallelization strategy using the OpenMP library, while developing a simulation tool based on a cellular automaton (CA) to run urban growth simulations. The characterization of an urban growth model CA is shown and it consists of a digitization process of the land use in order to get all the necessary elements for the CA to work. During the first simulation tests we noticed high processing times due to large quantity of calculations needed to perform one single simulation, in order to minimize this we implemented a parallelization strategy using the fork-join model in order to optimize the use of available hardware. The results obtained show a significant improvement in execution times in function of the number of available cores and map sizes, as a future work, it is planned to implement artificial neural networks in order to generate more complex urban growth scenarios.**

Keywords*—Cellular automata; parallel programming; simulation models; OpenMP; urban growth

I. Introduction

The evolution in the land use of the territory is a fundamental element in our society, since it manifests different variables that affect our daily life, for example, accessibility to different points of interest within the city, slopes of the land, etc. This evolution has gained interest mainly fueled by the different environmental problems especially those in urban areas [1]. Thanks to the advances in the computing field and the development of important analytical tools such as Geographic Information Systems (GIS) or simulation models, the study of the changes taking place in metropolitan areas has been promoted [2]. The analysis of the environmental alterations that result from these changes and the development of new planning instruments, has caused that different disciplines, specifically the Artificial Intelligence (AI), approaches from a computer and mathematical point of view to give alternative solutions to this problem [3].

Numerous modeling tools have emerged in recent years. In the case of urban growth, the models based on cellular automata (CA) are the most widely used [4]. Regression models, artificial neural networks (ANNs), multi-criteria evaluation techniques (MCE), and still incipient, agent-based models (ABM) can also be found.

The CA based models are oriented fundamentally towards the representation of the attributes of a given geographic region in a two-dimensional lattice, in which a neighborhood radius is defined and a certain rule of evolution is applied in order to define the behavior of the CA. With the use of these models it has been possible to generate territorial scenarios prospectively [5]. To generate these scenarios, a characterization of a CA is needed, this has different components, such as the size of the study area, maps of urban uses, map scales, neighborhood radius, evolution rules, slopes, and others geographical factors [6].

The developing of a CA based simulation tool to generate territorial scenarios prospectively in order to implement future simulation techniques, bring us to address some challenges. One of them was, the huge amount of mathematical operations needed in one single simulation, because the complexity of the algorithm to do such operations results to be exponential.

One key calculus in the whole simulation process is, the transition potentials (TPs) of each cell in the map, these TPs show the probability of a cell to change from one state to another. The amount of these TPs have a direct impact on the computation cost needed to perform the mathematical calculations.

To optimize these calculations, we enhanced sequential algorithms with parallelization strategies in order to maximize computational hardware. The library OpenMP (Open Multi-Processing), widely used in parallel programming, helps to implement a parallel strategy called fork-join. This allows to take advantage of hardware resources for the execution of processes in shared memory [7].

The present work aims to implement the fork-join strategy to speed up the necessary TPs calculations and to compare the results against the first sequential algorithm used in the simulation.

The base maps for the experiments where generated from the study area of Culiacan, México. Being the faster growing city in the State of Sinaloa, we plan to use the simulation tool to understand the dynamics of the urban changes and to forecast for planning urban development as a future work.

The remainder of this paper is structured as follows. All material and methods such as, the study area, digitation process, CA model and OpenMP are defined in Section II. Calculus of transition potentials for each pixel using the fork-join model are explained in Section III. Also proposed

implementations and experiments are analyzed. Finally, the study is concluded with future research directions.

II. Material and Methods

A. Study Area and its Digitization

The municipality of Culiacan is located in the central region of the State of Sinaloa (Fig. 1), forming part of the northwest of Mexico. The corresponding coordinates are: 24° 48'15 "N (north latitude) and 107° 25'52" O (longitude west), with an altitude of 54 meters above sea level. The city of Culiacan concentrates 81% of the population of the municipality that in the last 20 years has registered a very significant territorial and demographic growth, according to the last census of the National Institute of Statistics and Geography (INEGI), with population of around 800,000 inhabitants. In 1980 the city had an urban area of 5,163 hectares, by 1990 it increased to 7,377 hectares and by 2001 there were 9,800 hectares. This growth occurred in a disorderly, that is, anarchic way under the protection of political leadership resulting in the city currently having more than 275 neighborhoods, most of them formed in common lands, ecological reserved areas, places without feasibility of utilities due to its topographic composition [8].

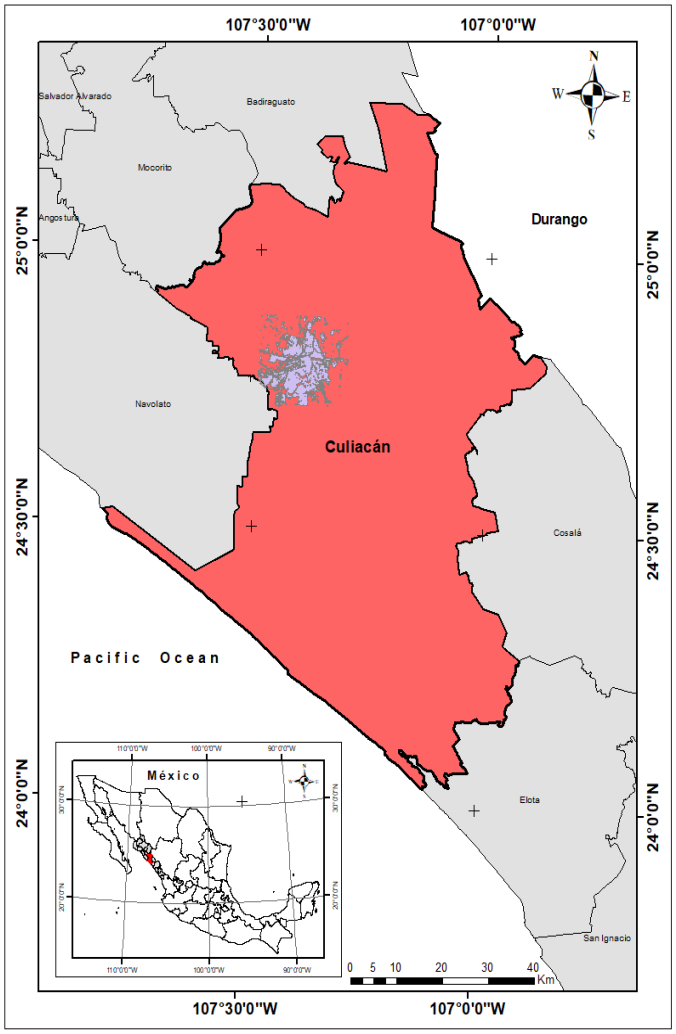

Fig. 1. Culiacán Sinaloa, México.

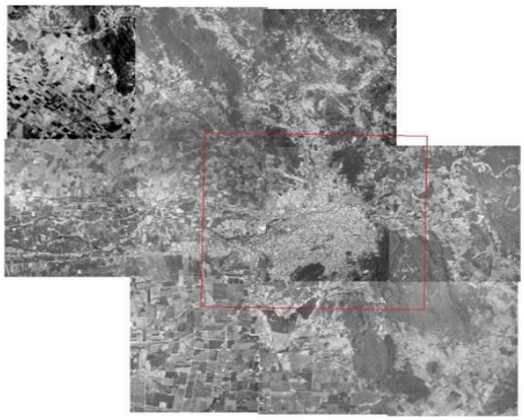

Fig. 2. Urban area of Culiacán 1997 and the study polygon (in red).

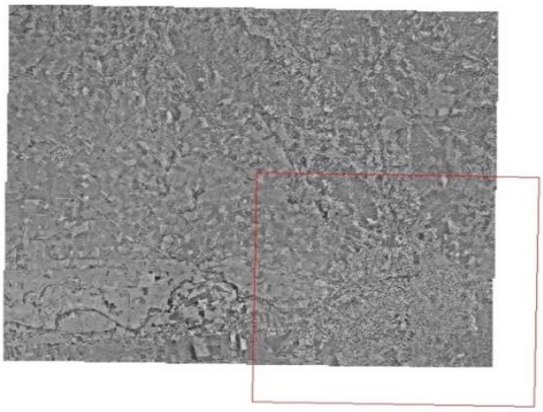

Fig. 3. Urban area of Culiacán 2004 and the study polygon (in red).

The digitization of the study area consisted in the generation of vector cartography over an orthophoto mosaic of the study area (Fig. 2 and 3). We worked with orthophotos (GeoTIFF) in the urban area of 1997 on a scale of 1: 20000, and in 2004 on a scale of 1: 10000, projected in WGS 84 / UTM 13N. The Geographic Information Systems (GIS) used were ArcMap® for vector maps, and IDRISI Selva ® for raster maps.

The digitization process (Fig. 4), urban land uses where classified in order to generate vector maps for each of them.

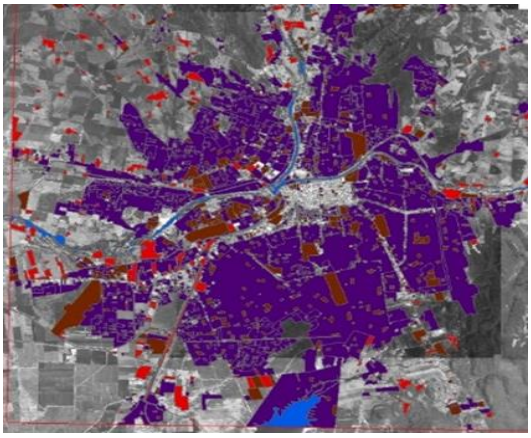

Fig. 4. Process of digitization of the urban area on the orthophoto of 1997.

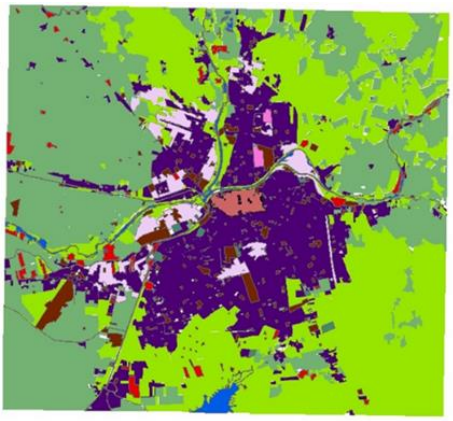

Fig. 5. Raster map 1997 Culiacan city.

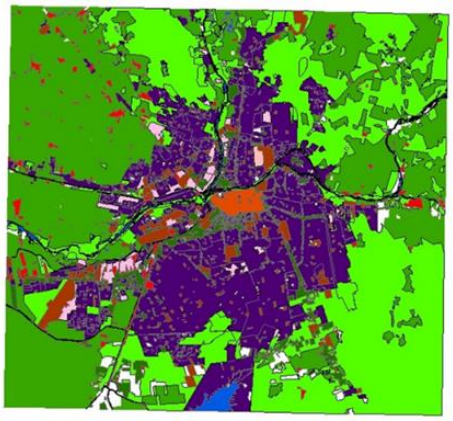

Fig. 6. Raster map 2004 Culiacan city.

TABLE I. URBAN USE CLASSES

Urban Land Use	Value in Raster Map
Residential	1
Commerce	2
Industrial	3

All generated maps from the study area were converted from vector to raster format, with the option "to raster image" placed in the module IDRISI Database Workshop.

The resulting raster maps (Fig. 5 and 6) and the classification of urban uses (Table I). Raster maps are the input for the CA model.

To manage map files we used GDAL (Geospatial Data Abstraction Library). This is a library of free use for the reading and writing of geospatial data providing low-level functions that allow the manipulation of raster files.

B. Celular Automaton Model and TP

The fundamental idea in CA Models is that the state of a cell at any given time depends on the state of the cells within its neighborhood in the previous time step, based on a set of transition rules [9]. The CA model used in this investigation is the one proposed by R. White (2), is a constrained cellular automata for high-resolution modelling of urban land-use dynamics [10].

As previously mentioned (Section I), the CA models are oriented towards the representation of the attributes of a given geographic region in a two-dimensional lattice, raster maps provides these data format to the CA.

A raster map can be represented formally by an array of real values. This matrix is represented as $A = \{a_{ij}\}$ of order mxn such that $0 \leq i \leq m, 0 \leq j \leq n$ where each element $A = [a_{ij}] \in \mathbb{R}$.

A neighborhood filter matrix (1) is required to analyze each element $A = [a_{ij}]$, this neighborhood is formally represented $B = \{b_{ii}\}$ of order nxn such that $0 \leq i \leq n$ where each element $B = [b_{ii}] \in \mathbb{Z}$.

$$B = \begin{bmatrix} b_{i-1,j-1} & b_{i-1,j} & b_{i-1,j+1} \\ b_{i,j-1} & b_{ij} & b_{i,j+1} \\ b_{i+1,j-1} & b_{i+1,j} & b_{i+1,j+1} \end{bmatrix}_{3x3\ neighbour} \quad (1)$$

The neighborhood filter is used to calculate the transition potential from state h to j for each element $A = [a_{ij}]$. The calculation methodology is detailed below:

$$\boldsymbol{P_{hj}} = vs_j a_j\left(1 + \sum_{k,i,d} m_{kd}\, I_{id}\right) + H_j \quad (2)$$

Where,

P_{hj}: Is the transition potential of state h to state j.

v : Stochastic perturbation term. $v = 1 + [-ln(random)]^x$.

($0 < random < 1$), and x allows you to adjust the size of the disturbance

s_j: represents the suitability of the state of the cell.

a_j: Euclidean distance from the cell to the nearest road.

m_{kd}: Calibration matrix, contains the weights of each cell as a function of its state k and distance d.

$$I_{id} = \begin{cases} 1, & i = k \\ 0, & i \neq k \end{cases}$$

i is the index of the cell in the current neighborhood, k

The transition potential P_{hj} of each cell A_{ij} is calculated only if the suitability of the objective state $s_j > 0$. That is, for each cell (pixel) in the map, its transition potential will be calculated except for those in which its suitability is equal to zero. For the neighborhood calculation, the calibration matrix m_{kd} gives each neighbor cell b_{ii} a weight based on its state and distance (subscript d formula 1) concerning the analyzed cell a_{ij}. The nearby neighbor cells will generally have a higher weight, positive values are taken for an attractive effect and negative for repulsive effect, these values tend to decrease as the distance increases between the analyzed cell and its neighbor, this is called Distance Decay Effect. When analyzing the neighbor cells, the I_{id} component helps to filter (multiplying by 1) cells with the same state.

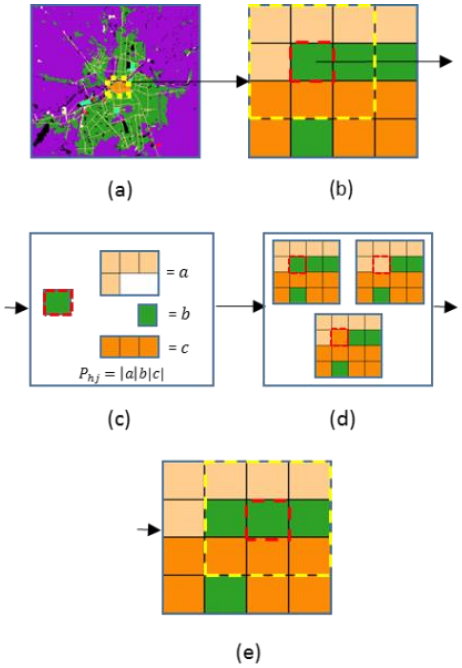

Fig. 7. Transition potential process.

Visually, in Fig. 7(a) we have the urban land use map, in Fig. 7(b), the neighbor is set to 3x3 around the analyzed cell. Fig. 7(c) calculates the transition potential P_{hj} of each cell from its current state h to a desired state j, the higher is selected. For this case we set as the higher to urban use. Fig. 7(d) analyzed cell change its value to the higher urban use. Fig. 7(e) shows observation window moves to the next cell.

An epoch has been completed when the last cell of the map is calculated. A simulation may require one or more epochs. If we take into account that this calculation must be done for each pixel of the map, we find a problem of computational complexity $O(2^n)$, this means, larger size of the input maps would increase the execution time of the simulation exponentially.

To handle this complexity, it was necessary to define a strategy to streamline the calculations, and this has been achieved with the development of programming modules in which parallel programming models are used by the OpenMP library.

C. Openmp the Fork-Join Model

OpenMP is a shared memory application programming interface, provides functions to facilitate shared memory parallel programming, and it is intended to be suitable for implementation on SMP architectures, OpenMP is based on the fork-join model [11], [12].

Under fork-join model, a program starts with a single execution thread, this is named as the initial thread. When a parallel directive (#pragma omp parallel**)** is executed in a current thread, it will create a group of threads (fork) called, parallel region. In this region, every thread can collaborate with the other threads. At the end of the directive, the parallel region terminates (join), and the initial thread is the only which continues. Fork-join model shown in Fig. 8 [11].

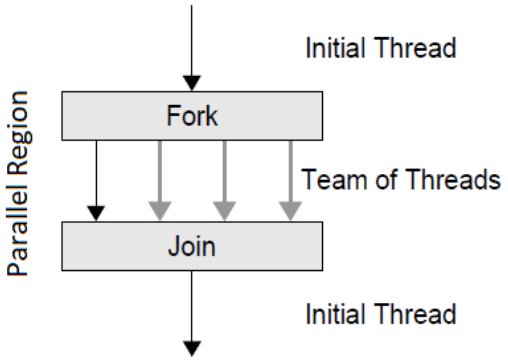

Fig. 8. Fork-join model.

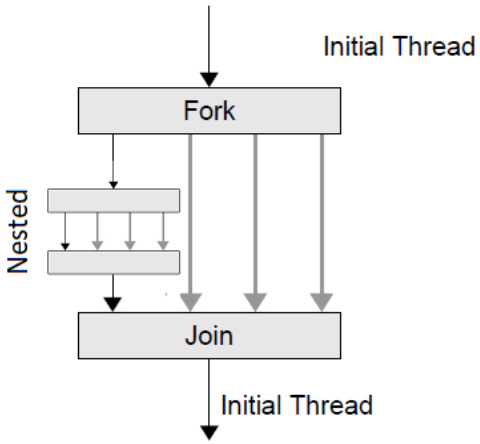

Fig. 9. Fork-join model with nested parallel region.

In addition, if required, OpenMP has the ability to create a parallel region inside another (nested), therefore, it is possible to divide a task as much as necessary and as much as the hardware allows it to, as shown in the Fig. 9.

III. Proposed Algorithm and Experiments

To create an OpenMP program from a sequential one, we must first to identify sequence of instructions that may be executed concurrently by more than one processor [11].

We identified the calculus of transition potentials P_{hj}, as the portion of sequential code which can be parallelized in order to do the mathematical operations using more than one processors' core.

Fig. 10 shows a schematic of the implementation of the strategy to carry out the calculation of two urban uses. In the raster map, an observation window is defined, that window is analyzed in two cores, the transition potentials are calculated (x_1 and x_2), one per core, the higher is selected and assigned as a new value to the cell.

For n urban uses, the basic idea is, for each cell A_{ij} we must calculate their P_{hj} from the current state h to objective state j where $j = n$ urban uses. Fig. 11 illustrates the process.

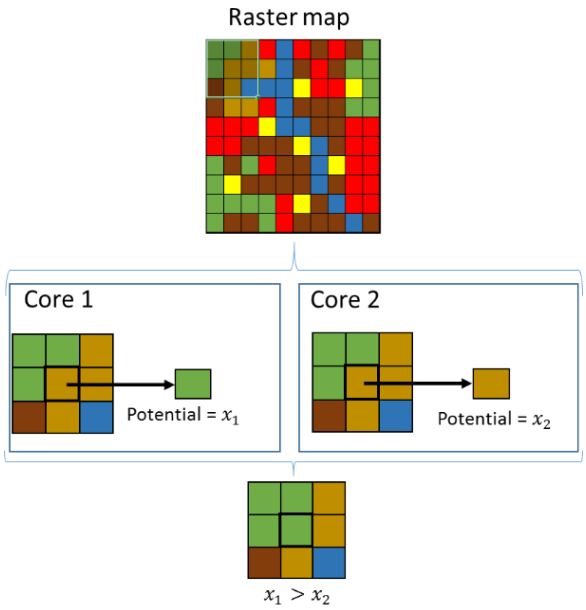

Fig. 10. Calculation of the transition potential (one per core) of a cell to two possible uses.

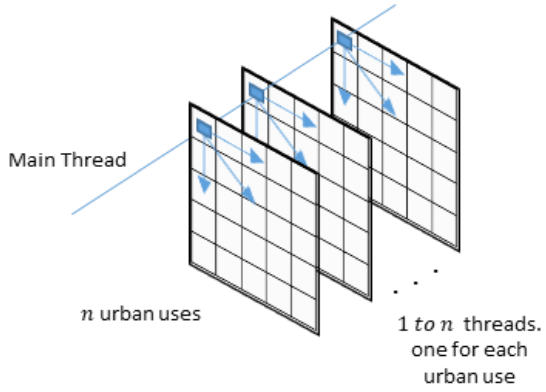

Fig. 11. Calculation of the transition potential for n urban uses.

```
1: 0 to epoch;
2: (0,0) to mapSize (nxm);
 {
3: pixel = cell(nxm);
4: nCores = nDinamicUses;
5: omp_set_num_threads(nCores);
6: #pragma omp parallel
        {
7:              j = omp_get_thread_num();
8:                      vectorP  [j]=calculateP
(pixel,j);
        }
9:   newMap(nxm) = hPotential (vectorP);
}
```

Fig. 12. Algorithm to calculate transition potentials.

The proposed algorithm (Fig. 12) to calculate the transition potentials is: 1) The epochs are defined. 2) The loop is set from the first cell to the last one. 3) The value of the current pixel is obtained. 4) The number of processor cores to be used is established (based on the number of dynamical uses). 5) The directive omp_set_num_threads(nCores) is used to establish the number of threads and the quantity of processor cores to use. 6) The parallel region #pragma omp parallel is initialized. 7) The thread number j = omp_get_thread_num() is identified. 8) The result of the potential of the analyzed cell is assigned to the potential vector, once the calculation has been completed in all cores. 9) The highest calculated potential is assigned to the analyzed cell.

The implementation was performed on an HP ProLiant ML350 G6 server, 12 GB RAM, 2 Intel Xeon E5645 processors (2.40 GHz), Linux Centos 6.9 operating system.

A. *Experiments*

Three conditions were considered for the experiments: 1) resolutions of raster maps from the study area. 2) Number of epochs for each resolution. 3) Times from sequential and parallel algorithm.

Table II summarizes the maps used.

Times from these 3 sets of resolutions were measured using the sequential algorithm and the one with the fork-join model.

TABLE II. RASTER maps RESOLUTION

resolution		**pixel size**
cols	**rows**	**(meters)**
397	366	50
9,925	9,150	25
19,850	18,300	1

TABLE III. RUNNING TIMES FOR EACH RESOLUTION

pixel resolution		**pixel size**	**Execution times (minutes)**	
cols	**rows**	**(meters)**	**Sequential**	**OpenMP**
397	366	50	0.08	0.02
9,925	9,150	25	53.24	13.01
19,850	18,300	1	214.1	54.69

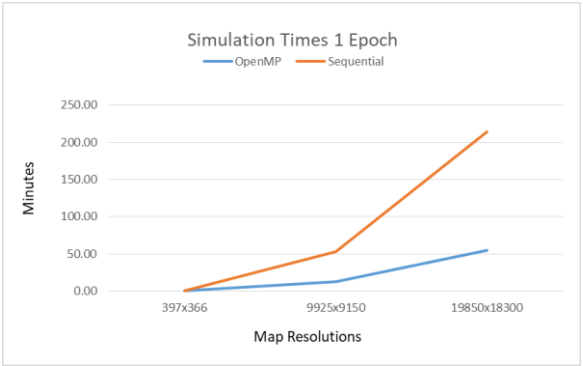

Fig. 13. Simulation times with different resolutions, 1 Epoch each.

TABLE IV. RUNNING TIMES FOR EACH RESOLUTION

pixel resolution		pixel size	Execution times (minutes)	
cols	rows	(meters)	Sequential	OpenMP
397	366	50	0.46	0.11
9,925	9,150	25	230.23	58.91
19,850	18,300	1	929.18	245.42

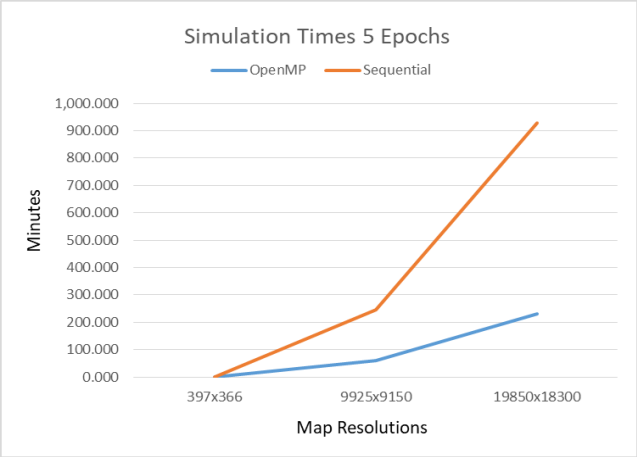

Fig. 14. Simulation times with different resolutions, 5 epochs each.

Table III shows the firs results for 1 epoch, times are shown in minutes.

It is evident the correlation in the maps sizes and the running times, as shown in Fig. 13 and 14, simulation times grow as we incremented the map sizes.

Table IV shows the first results for 5 epoch, times are shown in minutes.

B. Discussion

Execution times using sequential and parallel algorithms increase along with the maps size, but no linearly. As we expected, sequential method is the one that takes the most time to complete the calculations. OpenMP helped to reduce in almost 4 times the execution times.

At lower resolutions, there is no big difference due to the minimum execution times. Resolutions around 500^2 pixels should not represent big challenge when working with small number of urban uses, for our experiments we use 3.

Since most simulations require from 3 to 8 urban uses and maps sizes from 1000^2 to 5000^2 pixels, it is necessary to implement a strategies to enhance calculations in the development of this kind of tools.

The number or epochs is critical in this, depending of the kind and configuration, every simulation needs several epochs. As shown in Table IV, for maps sizes lower than 1000^2 pixels, we expect times under 59 minutes in our future simulations.

IV. CONCLUSION AND FUTURE WORK

OpenMP provides mechanisms that help to reduce execution times when implemented in a simulation model based on a cellular automaton obtaining improvements of up to 4 times.

Since numerous simulations must be performed in order to achieve different tasks such as calibrating the simulation model, perform sensitivity analysis or tests with different urban uses, OpenMP must be considered as a very interesting option when implementing algorithms for this area.

Future work: First, our CA based simulation tool became faster after the implementation of the parallelization strategy to calculate transition potentials, now we need to continue testing more resolutions and urban uses. Second, we are considering implementing CUDA along with artificial neural networks in order to improve the forecast of urban growth.

REFERENCES

[1] W. Plata-Rocha, M. Gómez-Delgado, y J. Bosque-Sendra, “Simulating urban growth scenarios using GIS and multicriteria analysis techniques: A case study of the Madrid region, Spain”, Environ. Plan. B Plan. Des., vol. 38, pp. 1012–1031, 2011.

[2] C. G. Ralha, C. G. Abreu, C. G. C. Coelho, A. Zaghetto, B. Macchiavello, y R. B. Machado, “A multi-agent model system for land-use change simulation”, Environ. Model. Softw., vol. 42, pp. 30–46, 2013.

[3] E. F. Lambin, B. L. Turner, H. J. Geist, S. B. Agbola, A. Angelsen, J. W. Bruce, O. T. Coomes, R. Dirzo, G. Fischer, C. Folke, P. S. George, K. Homewood, J. Imbernon, R. Leemans, X. Li, E. F. Moran, M. Mortimore, P. S. Ramakrishnan, J. F. Richards, H. Skånes, W. Steffen, G. D. Stone, U. Svedin, T. a. Veldkamp, C. Vogel, y J. Xu, “The causes of land-use and land-cover change: Moving beyond the myths”, Glob. Environ. Chang., vol. 11, pp. 261–269, 2001.

[4] F. Aguilera Benavente, W. Plata Rocha, y J. Bosque Sendra, “Diseño y simulación de escenarios de demanda de suelo urbano en ámbitos metropolitanos”, Rev. Int. sostenibilidad, Tecnol. y humanismo, pp. 57–80, 2009.

[5] F. Aguilera Benavente, L. M. Valenzuela Montes, J. A. Soria Lara, M. Gómez Delgado, y W. Plata Rocha, “Escenarios Y Modelos De Simulación Como Instrumento En La Planificación Territorial Y Metropolitana”, Ser. Geográfica, vol. 17, pp. 11–28, 2011.

[6] R. White y G. Engelen, “High-resolution integrated modelling of the spatial dynamics of urban and regional systems”, Comput. Environ. Urban Syst., vol. 24, pp. 383–400, 2000.

[7] R. Chandra, Parallel Programming in OpenMP. 2001.

[8] J. A. Inzunza, “La Planeación Urbana en el Municipio Mexicano: Culiacán, Un Caso de Estudio”, 2003.

[9] J. I. Barredo y M. Gómez Delgado, “TOWARDS A SET OF IPCC SRES URBAN LAND-USE SCENARIOS : MODELLING URBAN LAND-USE IN THE MADRID REGION European Commission – DG Joint Research Centre Institute for Environment and Sustainability Departament of Geography , University of Alcalá”, pp. 1–16, 2000.

[10] R. White, G. Engelen, y I. Uljee, “The use of constrained cellular automata for high-resolution modelling of urban land-use dynamics”, Environ. Plan. B Plan. Des., vol. 24, núm. 3, pp. 323–343, 1997.

[11] B. Chapman, G. Jost, y R. Van Der Pas, Using OpenMP. The MIT Press, 2008.

[12] T. Mattson y L. Meadows, “Introduction to OpenMP”. [On Line]. Disponible en: http://www.openmp.org/wp-content/uploads/omp-hands-on-SC08.pdf.