Ecological Informatics 36 (2016) 135–144

Contents lists available at ScienceDirect

Ecological Informatics

journal homepage: www.elsevier.com/locate/ecolinf

Monte Carlo model framework to simulate settlement dynamics

Emmanuel N. Millán[a, b, c, *], Silvana Goirán[b, d], Leonardo Forconesi[b], Julieta N. Aranibar[b, d], Carlos García Garino[c], Eduardo M. Bringa[a, b]

[a] CONICET, Mendoza, Argentina
[b] Facultad de Ciencias Exactas y Naturales, Universidad Nacional de Cuyo, Mendoza, Argentina
[c] ITIC, Universidad Nacional de Cuyo, Argentina
[d] Instituto Argentino de Nivología, Glaciología y Ciencias Ambientales (IANIGLA), CONICET, CCT-Mendoza, Argentina

ARTICLE INFO

Article history:
Received 22 January 2016
Accepted 4 November 2016
Available online 9 November 2016

Keywords:
Settlement dynamics
Spatial distribution
Monte Carlo simulations
Shared memory
Performance analysis

ABSTRACT

We developed an open source Monte Carlo based model to simulate Settlement Dynamics in Drylands (SeDD). The model assigns partial probabilities to each pixel within a grid region, based on several factors that can influence the establishment and subsistence of settlements: groundwater depth, vegetation type, proximity to rivers, paved roads, old river beds, and existing settlements. Partial probabilities are considered to be independent from each other, and therefore multiplied to calculate an overall probability for each pixel. Settlements are assigned by maximum probabilities or randomly, according to pre-established threshold probability values. We also modeled the gradual reduction of vegetation caused by a new settlement in neighboring pixels, decreasing the probabilities related to vegetation type. The final distribution of settlements is given by an average over multiple Monte Carlo simulations. The model is computationally efficient and could be used to rapidly explore different scenarios of settlement dynamics and vegetation degradation in arid environments, and other environmental factors that can be added to the framework without performing changes in the source code.

1. Introduction

Land use in arid and semiarid areas is conditioned by water limitations and vegetation characteristics. If water is not available for the development of irrigated agriculture, livestock production is one of the main economic activities in drylands (Asner et al., 2004; Corvalan et al., 2005). Grazing in these systems may increase the risks of degradation and desertification (Dawelbait and Morari, 2012). Livestock settlements are not located homogeneously in the landscape, but show patterns of aggregation at different spatial scales, affecting vegetation (Goirán et al., 2012). An understanding of the different drivers of settlement distribution and activity is crucial to estimate future effects of environmental and socio-economic changes on arid and semiarid ecosystems. Ecosystem-human interactions, such as human dispersion in a landscape with gradients of availability of natural resources, can be simulated with different approaches. Habitat models simulate plant and animal habitat ranges with a variety of correlation and mechanistic approaches, based on known species distributions or physiological requirements (Guisan and Zimmermann, 2000; Hosseini et al., 2013; Kearney and Porter, 2009). Dispersion of plants has been simulated with spatially explicit, stochastic models, based on environmental conditions relevant for their survival (i.e., abiotic features and habitat type), and dispersion (i.e., corridors such as roads and rivers) (Fennell et al., 2012). Land cover changes in the Amazon, from mature forests to pastures, have been simulated with a model that assigns different land uses based on decision making rules that depend on demographic rates (fertility, migration), institutions, and agricultural prices (cattle and rice) (Evans et al., 2001). Agent-based models (ABM) simulate the behavior of individual "agents" based on environmental, social, and political factors, such as food and water availability, social status and organization (households, villages, tribes), marriage, and agricultural practices (Crabtree and Kohler, 2012; Kohler et al., 2012). ABM have been used to simulate the growth, migration and change of past societies, like the Anasazi (also called Pueblo). Households were defined as agents, with attributes such as age, location, grain stocks, death age, and nutritional needs. The landscape for the simulation contained information on maize crop yield, surface and ground water, soil type, soil degradation, wood, forage, and hunting animals (Axtell et al., 2002; Dean et al., 2000; Kohler et al., 2012). A similar approach, including soil erosion, was used to study prehistoric

* Corresponding author at: FCEN, Universidad Nacional de Cuyo, Padre Jorge Contreras 1300, Parque General San Martín, CP M5502JMA, Mendoza, Argentina.
E-mail address: emmanueln@gmail.com (E. Millán).

http://dx.doi.org/10.1016/j.ecoinf.2016.11.002
1574-9541/

settlements around the world (Barton et al., 2010; Michael Barton and Ullah, 2010). The multi-agent model was also used to simulate spatio-temporal dynamics for human-landscape systems (Le et al., 2010, 2008). These models may simulate complex and numerous processes, but in cases where data to validate different components of model behavior are not available, it may be difficult to evaluate whether the added complexity contributes significantly to understand real ecosystems.

The objective of this study is to develop a model framework that simulates the spatial distribution of human settlements in drylands and their effect on vegetation, based on environmental conditions. We use the Monte Carlo (MC) method (Chan, 2013) to simulate the dynamics of the settlements. For human occupation of the land, we consider habitat aptitudes as in niche models, adding the stochastic characteristics of human decisions, and a knowledge of ecosystem functioning and services obtained by observations or cultural inheritance. The model of Settlement Dynamics in Drylands (SeDD) presented here is based on six environmental factors, which we consider important in arid areas: surface and ground water availability, vegetation type, existing settlements, access routes, and old river beds. These factors provide different services to settlers, such as water provision for humans and livestock, forest products for construction and forage, transport and communication with existing settlements and other regions, and initial labor, materials, and water during the construction period. The model assumes that places with higher availability of water and forest resources have a higher probability of being settled, independently of social structure, and assigns settlements in places where probabilities (representing suitability) are higher. However, a number of settlements may be established stochastically, in randomly selected sites, to take into account social factors not included directly in the model. This component of the model aims to simulate pressures to establish settlements in highly unfavorable sites. Settlement-vegetation interactions are simulated by gradually decreasing vegetation around settlements, up to a distance based on previous studies (Goirán et al., 2012; Karnieli et al., 2008; Ringrose et al., 1996). Our model differs from typical plant and animal dispersion models because it assumes that settlers have a prior knowledge of the environment in the entire region, simulating environmentally-based human informed decisions.

The simplicity of the model may allow testing, with high computational efficiency, the relative importance of different environmental factors on settlement distribution in drylands. In this work the framework is tested with a parallel implementation using shared memory in multi-core CPUs in up to 16 cores and has also been ported to run on GPUs in a previous work (Millán et al., 2016). The model may also be useful to estimate future settlement distributions and associated degradation that may result from changes in water availability, access routes, policies and land ownership. The framework can be adapted to support more environmental factors since the code is open source and available at https://sites.google.com/site/simafweb/proyectos/simulacion-de-poblaciones.

This work is organized as follows: Section 2 describes the model, code implementations, and the numerical experiments. Next, in Section 3 the results of the experiments are discussed along with a multi-core performance evaluation of the framework. Finally, the conclusions are presented in Section 4.

2. Materials and methods

2.1. Description of the model

The SeDD model divides the region of interest into a regular grid, made of square pixels, and estimates the "probability" of each pixel of being settled. Probabilities are given by the combination of six factors, assumed to affect settlement establishment and functioning in arid areas: groundwater depth, minimum distance to paved roads, rivers, and existing settlements, presence of old river beds, and pixel vegetation type. In arid areas, livestock settlements generally rely on surface and groundwater provided by the owners. Low groundwater depths and close distances to rivers may affect the accessibility to owners. Roads and river beds may provide easier access to commercialization centers. River beds also provide access to accumulated rain water and better access to groundwater. Different vegetation types may diferentially affect settlement establishment and livestock subsistence. These factors are represented in the model in six different layers, with each pixel having a partial probability value for each factor, given by its characteristics and potential services to settlers. For example, a minimum distance to rivers and roads is considered to favor settlements, so maximum partial probability values are assigned to pixels near these features. Certain types of vegetation may provide more services to settlers than others, so they will have higher partial probabilities given by vegetation type. Total probabilities for each pixel are calculated by multiplying partial probabilities given by each of the six environmental factors, assuming that they act independently. The model then assigns a new settlement in pixels with maximum probability if a threshold probability value is surpassed, or at random if the threshold is not reached anywhere. This stochastic feature is included to take into account social behaviors that cannot be modeled with the environmental factors. After a settlement is established, the partial probabilities given by "vegetation type" in the neighborhood of this settlement are gradually decrease within a given radius, representing vegetation consumption by the livestock of a new settlement. Because of the stochastic nature of the model, multiple simulations are required for a given set of input parameters or initial conditions, and finally, the simulation results are then averaged. The cases presented here deal with a square grid, but irregular regions, including non-convex topologies, can be easily accounted for by inscribing the desired area into a larger square grid, and taking undesired pixels off by assigning them zero probability for being settled. These "boundary" conditions are given by a "mask" grid including information on which pixels have zero probability. Because this is still a regular grid, pixels can be simply indexed, or identified by their (i,j) coordinates within the grid, where "i" represents the row number and "j" the corresponding column.

A flowchart showing how the simulations work is presented in Fig. 1, and explained below:

(a) The model first reads the input files and necessary parameters, including rivers, roads, etc. Partial probabilities are evaluated in each pixel for each environmental factor, based on initial values of input grids and parameters.
(b) The probability associated with vegetation type is decreased around existing settlements in each time step.
(c) Then, the model calculates the total probabilities (P) for each pixel in the grid, by multiplying partial probabilities of each environmental factor in each pixel. The code then performs a search for the maximum value of P, $Pmax$. If there is more than one pixel with the same $Pmax$, the model randomly chooses one of them.
(d) If $Pmax > Pset$ (threshold of total probability P to place a settlement by high probability), the model assigns a new settlement in the pixel with $Pmax$.
(e) If $Pmax < Pset$, the model may still place a settlement at a random location. First a random number is used to select a single pixel from the grid and, then, a second random number ($Pran$) is generated from a uniform distribution in $(0, 1)$ to decide if a settlement will be established in that pixel. This decision is based on a pre-set rejection threshold ($Pthresh$). For instance, for a rejection rate of 70%, there will be a settlement in the randomly chosen pixel only if $Pran$ is greater than $Pthresh = 0.7$. The rejection threshold is related to pressures which

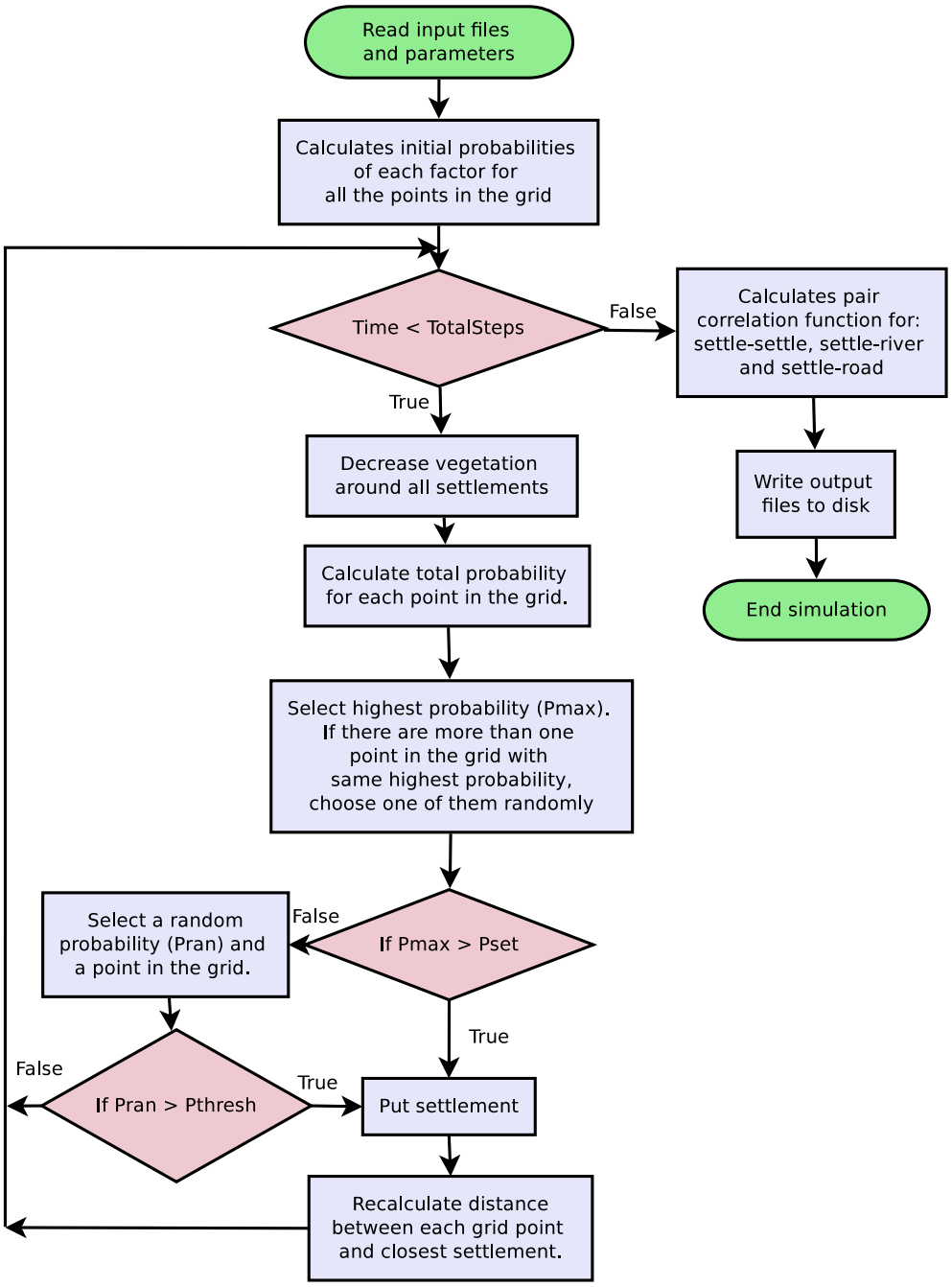

Fig. 1. Flow diagram of the model.

force settlements in places where environmental conditions could be unfavorable.

(f) Partial probabilities for distance from existing settlements are recalculated at each time step, considering newly established settlements.

(g) Steps (b)–(f) are repeated for each time step until the end of the simulated time is reached, i.e. by running a number *TotalSteps* of iterations. Then, the model calculates pair correlation functions for the distance between settlements and rivers, settlements and road, and between existing settlements.

(h) The user may transform the resulting grid of final vegetation values into a final vegetation map with a GIS software or similar, assigning vegetation type classes to given ranges of vegetation values obtained after degradation due to livestock grassing and human intervention.

2.1.1. Required input

To execute simulations, the framework requires a set of input grids or maps and input parameters. The maps of the simulated area have to be prepared with a GIS software (Geographic Information System) like ArcGIS, Quantum GIS or GRASS, and exported to ASCII format. The input parameters can be passed to the framework as parameters in the command line or inside a text file. The input parameters are first read from the input file, and then from the command line, overriding the values present inside the input file. The following is a description of the required input grids and the input parameters.

Input grids. The landscape information needed by the model to calculate partial probabilities is read from seven input grid files: initial distribution of settlements, existing paved roads, existing rivers,

initial vegetation (five types of vegetation are currently included, which are given different vegetation probabilities), water table depth (classified into 3 depth classes), old river beds, and a mask (pixels with *total probability* = 0), which defines the limits of the study area, and can be irregularly-shaped. The mask area corresponds to other productive systems, such as irrigated oases, where livestock settlements cannot be placed, or defined geographic features, such as mountains or rivers, which reduce or prevent animal movements and significant interactions between settlements located in the two sides of these barriers. The framework supports more environmental factors that can be provided by the users. The only requirement to add new factors is to build the input maps with a GIS tool and to add the factor parameters (distance interval and probabilities). The type of environmental factors supported by the model are discussed in the section "Environmental factors".

Input parameters. In this section we discuss the input parameters which define the size of the grid, the number of time steps, the maximum number of settlements that could be placed, and the probabilities of each class for a given factor. Also, a summary of derived variables is given in the following list.

(a) *pixelsize* in km. This value is used to calculate input and output variables, such as distances from settlements to rivers, roads, and other settlements.
(b) *Length* (in pixels). Number of pixels in a side of the simulated square grid (*Length* × *Length*).
(c) *Pset*, being the minimum probability value to establish a settlement in pixels with maximum probability (*Pmax*). It determines the minimum environmental requirements to establish a settlement by maximum probability.
(d) The rejection threshold, *Pthresh*, determines whether a randomly selected pixel (chosen when *Pmax* < *Pset*) is settled.
The temporal dynamics of the simulation is controlled by the following time related parameters.
(e) Total simulated time, in steps, *TotalSteps*.
(f) Minimum time steps that the simulation waits until placing a new settlement, in steps, *TimeMinSet*.
(g) Maximum number of settlements that could be established in a given time interval (in this case the total simulated time), *SettlementMax*.

Environmental factors. To perform the simulation, the model needs environmental factors, which have to be passed to the simulation inside the formatted input file. For each factor, an input map is also needed, and the filenames of these maps are related to the name of the factor. The parameter *matricesdir* can be used to specify from which directory the model can read the input maps. Input maps, together with other input factors, are used to assign partial probabilities to grid points. Partial probabilities can take values from 0.0 to 1.0.

The model supports four types of environmental factors:

- Type 1: distance-related factors that consist of ranges of distances with an associated probability for each range, such as distance from a given pixel to the closest paved road or river. These factors have the following syntax: the "factor;" label, name of the factor (without spaces), type of factor (1 in this case), number of distance ranges, values of each distance range (in km), followed by the probability for each range. For instance:

```
factor; road; 1 3 1 5 12 1.0 0.6 0.1
```

This example creates a factor named "road", type of factor "1", with three distance ranges (in km): [1, 5), [5, 12) and [12, ∞), each range with a probability of 1.0, 0.6 and 0.1, respectively. The same type of factor can be used for the distance to the river, changing the distance range and probabilities.
- Type 2: this type can be used to assign a probability to an attribute of each pixel in the grid, such as the depth of groundwater or if a given pixel can be settled. This is useful to define irregular grids, terrain height, etc. These factors have the following syntax: the "factor;" label, name of the factor (without spaces), type of factor (2 in this case), number of categories of the given attribute, values of each category, followed by the probability of each category. For instance:

```
factor; groundwater; 2 3 1 2 3 1.0 0.7 0.1
factor; mask; 2 2 0 1 0.0 1.0
```

The first example creates a factor named "groundwater", type of factor "2", with three categories: 1, 2 and 3; each category with a probability of 1.0, 0.7 and 0.1, respectively. The values of the categories are read from the input map that has to be constructed separately, for instance with a GIS tool. The second example creates a factor named "mask", with two categories, 0 and 1, with probabilities 0.0 and 1.0 respectively. The input map "mask" has values of 0 for pixel in the grid where settlements cannot be placed, with the probability 0.0 which cancels the rest of the factors when they are multiplied.
- Type 3: this type of factor can be used to assign different probabilities to pixels in the grid according to an input map, and change that probability every step of the simulation. We use this type of factor to simulate the different types of vegetation present in the simulated area and to decrease it in each step around settlements due to the effect of livestock or human interaction with the vegetation. Different rates for degradation can be given to this type of factor, each rate has to be associated with a distance range (in km). This factor has the following syntax: the "factor;" label, name of the factor (without spaces), type of factor (3 in this case), number of categories of the given attribute, values of each category, followed by the probability of each category; next, the number of distance ranges, with the values of each distance in km and finally, the rate of change which has to be applied in each step of the simulation. For instance:

```
factor; vegetation; 3 5 1 2 3 4 5 1.0 0.9 0.3
0.1 0.3 2 2 5 0.015 0.010
```

This example declares a factor named "vegetation", with a type 3, including five attributes that are expected to be found by the model in the input map file, each with a probability assigned (1.0, 0.9, 0.3, 0.3 and 0.1); next, the number of distances to apply the degradation of vegetation, in this case "2" ranges are used, with values of [0, 2) and [2, 5), with rates 0.015 for the first range and 0.01 for the second range.
- Type 4: this type of factor is used to store the information of the initial settlements in the grid, and to store the newly created settlements during the simulation. This type of factor could be used to represent other kind of entities than livestock settlements. The syntax for this factor is the following: the "factor;" label, name of the factor (without spaces), type of factor (4 in this case), number of distance ranges between settlement (or other entities) that can be placed with a given probability, values of each distance range (in km), followed by the probability for each range. For instance:

```
factor; settlement; 4 2 2 6 1.0 0.1
```

The previous example shows the syntax for the type 4 factor, the name of the factor is "settlement", with a type 4, and two distance ranges, $[2, 6)$ and $[6, \infty)$, the first with a probability of 1.0 and the second with a probability of 0.1. These range values can be used to control clustering of settlements around "mother" settlements. The range distance could also be used to have an exclusion area for new settlements around existing settlements by choosing the first distance value >0.

2.1.2. Simulation output

To analyze the results from each simulation, the framework returns several output files, detailed in the list below.

- Grid of simulated settlements, indicating the position within the grid of old and newly established settlements. The grid indicates by different characters whether a new settlement was established randomly, in an unfavorable pixel, or by maximum probability.
- Grid of vegetation values. This grid presents final probability values (0 to 1) given by vegetation, resulting from the degradation of vegetation around settlements, at a rate given by the input parameters. This grid can be transformed into a vegetation map, assigning vegetation types to different ranges of probabilities resulted from the degradation of the vegetation around settlements. Decreasing vegetation values represent decreasing vegetation cover and grass abundance around settlements, as commonly observed in arid ecosystems (Goirán et al., 2012; Karnieli et al., 2008; Ringrose et al., 1996).
- Pair correlation histograms. As a validation tool, the model generates pair correlation histograms that describe settlement spatial distribution. In order to compare all histograms from the same or different simulations they were normalized to have unit area, transforming each bin value to a proportion of the total histogram area. There are several histograms built by default at the end of each run:

    1. *settlement-road* distances, indicating the number of settlements at different distances from the paved road.
    2. *settlement-river* distances, indicating the number of settlements at different distances from rivers. For each settlement, we calculate the closest distance to a road/river, and add that distance to the corresponding bin in a histogram. This approach allows us to rapidly identify possible tendencies like clustering close to rivers and roads.
    3. *settlement-settlement* distances, showing settlements density according to the distance between them. For these histograms, the model calculates pair-correlation functions $g(r)$ as done for atom simulations, where $g(r)$ determines thermodynamical properties (Allen et al., 1987). $g(r)$ is also called the radial distribution function (RDF) and is defined as follows: for an average density of atoms, $g(r)$ gives the average "local" density at a distance r from a given atom. To calculate the *settlement-settlement* correlation, the model selects one settlement, builds concentric radial bins, i.e. ring-shaped bins of width dr, and counts the number of settlements falling in each bin. This is repeated for all settlements, and the histogram is normalized with the area of the corresponding bins, to be consistent with an average density of the system. This approach can rapidly identify exclusion regions, which for atoms correspond to lack of overlap due to their characteristic "size", and can also identify possible short and long range ordering, with short range ordering associated in this case with clustering of settlements likely associated with family ties. We note that this approach is similar to the ecological point pattern analysis of Wiegand et al. (1999) and Wiegand and Moloney (2004). A similar approach was used by Winter-Livneh et al. (Winter-Livneh et al., 2010), who applied spatial analysis (Moran's I autocorrelation and Ripley's K-function) and a general linear model to study the ancient Chalcolithic site distribution pattern in the Northern Negev.

2.1.3. Calculating average of simulation results and residuals

For a given set of parameters, the output of the simulations changes in different runs due to the stochastic contributions of the model. Therefore, we run a number N of simulations for a given set of parameters, and calculate the average value for each of three output histograms $\langle H_N \rangle$: *settlement-settlement*, *settlement-road*, and *settlement-river*. The number n of radial bins (distance classes) in the histogram, of width dr, should be such that $n \times dr \approx 1/4$ of the side length of the region studied, to avoid boundary artifacts. Then, we calculated the residuals between a simulated (average of N simulations) and a reference case (i.e., in this case, the reference case is the average of $N - 1$ simulations). The residuals of two histograms of correlation functions for *settlement-settlement*, *settlement-river*, and *settlement-road* distances, versus distance with distance bins Δr are calculated with Eq. (1).

$$\sum_{i=1}^{n} (H'_i - H_i)^2 \tag{1}$$

where H'_i is the value (fraction of settlements located at distance i) of histogram $\langle H' \rangle$ at bin i for the simulated distribution, and H_i is the value of histogram $\langle H \rangle$ at bin i for the reference distribution. When the residuals change less than 10% we stop increasing N. This approach can be used to compare simulated histograms with real data, considering the histograms of the real settlement distribution as a reference case.

2.2. Model evaluation using simplified initial conditions

In order to test the model behavior, we considered several simple initial conditions, where the output should follow simple trends. These conditions separately include each of the different factors that determine partial probabilities. The values for the input parameters are detailed in Section 3. The exact values in the initial input are not crucial here, because any reasonable set of parameters (based on hypothesized behavior) would give qualitatively similar results. In addition, in order to test model stability, 100 simulations were run with the environmental variables and parameter values detailed in Section 3 for case **(i)**.

2.3. Computational considerations

The code only requires the GCC compiler, the NVIDIA *nvcc* compiler (the CPU and GPU model are integrated in the same source code), and standard compilation tools found in any GNU/Linux distribution. We do not use external libraries that would require additional work to make the simulation run. The simulation is run from the command line. Input parameters have default values placed in the code, which are used if the program does not receive new input parameters. To obtain averages and residuals from multiple simulations, we created a script to run the simulation for multiple test cases with possibly different input files. This script, which is written in Bash (Ramey and Fox, 2003), runs the simulation N times with any number of input files. For each input file it creates a directory with

all the output files of each run, then saves in a single *.csv* file each of the pair correlation histograms for each simulation.

The code takes around 2 s for a typical run with six environmental factors, in a single core of an AMD FX-8350 4 GHz, for a grid size of 150 × 150 pixels (with a pixel size of 0.75 km) and during 2000 steps, using only 4 MB of memory. For a region of 100 × 100 km, with a grid size of 0.1 km, using matrices of 1000 × 1000 pixels is still possible to run the code without incurring in memory issues, since it uses only 125 MB of memory. The above timing is for a single run ($N = 1$), but about $N \sim 100$ simulations are needed to obtain appropriate statistics for one set of parameters. The current simulation code works using one CPU core or multiple cores using Shared Memory with OpenMP (Dagum and Menon, 1998). We also created a script to run N different cases in multiple cores of a single CPU.

A parameter sweep would be needed to test the stability of the results to small changes in the known parameters, or to optimize certain unknown parameters, minimizing the overall error. Details on the implementation of such a parameter sweep for a particular problem will be given in another, more in depth study. A typical parameter sweep might require millions of runs to sample different parameter sets, because a single set will require N repetitions to obtain reasonable statistics, with $N \sim O - (100)$. Therefore, we implemented a computational script, in the Ruby language (www.ruby-lang.org/en/) to speed execution. For instance, if we have to optimize m variables, and each variable can have z different, discrete, values, the number of runs needed for the parameter sweep will be $N \times z \times m$. The Ruby script runs N simulations for each set of input parameters and calculates the residuals between each case and a reference case, saving the results in an output file. The script supports multiple CPU cores in a single workstation, queuing one independent process with a given input data in each core, which can provide a large speed-up in code execution in multi-core machines.

Improving the speed of the simulation would allow a much broader parameter sweep, and it could minimize errors thanks to improved parameter selection. We have already ported the code to CUDA (NVIDIA) and obtained good speedups compared with the serial CPU code (Millán et al., 2016). In this work we identified which functions of the serial code were consuming the most simulation time. Three functions account for ~95% of the simulation time: the first calculates the total probability for each pixel in the grid, the second calculates the minimum distance between all pixels with the road, river and settlements, and finally the third is in charge of re-calculate the distance between each pixel every time a new settlement is added to the grid. These three functions are good candidates to be parallelized with OpenMP to improve performance, due to the fact that the calculations done in each pixel of the grid (a probability or a distance) are independent of the calculations of the rest of the pixels. In the next section we test the framework with the simplified initial conditions and test the performance of the parallel code running in two multi-core computers.

3. Discussion of results

3.1. Simple case simulations

The settlement distributions for different cases (a to i) can be seen in Fig. 2, with a pixel size of 0.75 km and a *Length* of 150 × 150 pixels. Figs. 3 and 4 show the pair correlation functions of settlement-settlement, and settlement road distances, respectively, for the cases shown in Fig. 2.

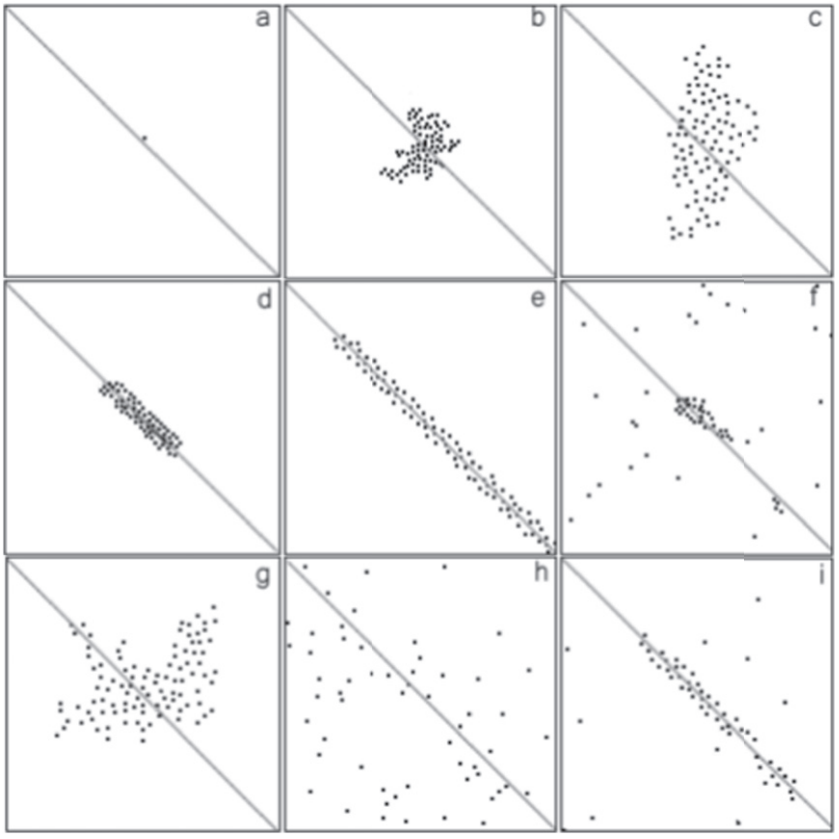

Fig. 2. Simulations with simple initial conditions to test the model, see text for more details.

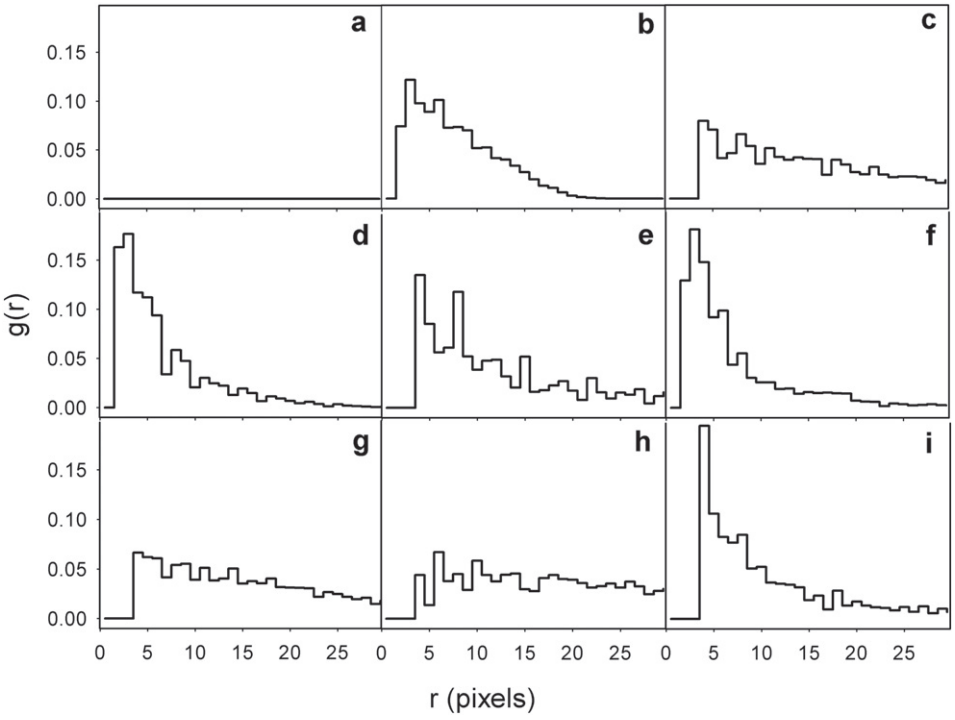

Fig. 3. Pair correlation histograms of *settlement-settlement* distribution, for the same simulations as in Fig. 2. *r* is distance to settlement in pixels.

As an initial test, we made a grid with a road across the diagonal, with one initial settlement close to that road (Figs. 2a, 3a and 4a). When we considered the distance between settlements as the only driving factor for settlement establishment (by disabling all environmental factors), with a maximum distance class of up to 1.8 km, the settlements are set in a cluster around the first settlement (Figs. 2b; 3b, and 4b). In Fig. 2c we only changed the distance between settlements, with an interval distance of [2, 3.6) km. The resulting settlement cluster is larger and the distance among settlements increased (Fig. 3c).

In the cases presented in Fig. 2d and e, we added the distance from the settlement to the road as a driving factor, and the settlements are

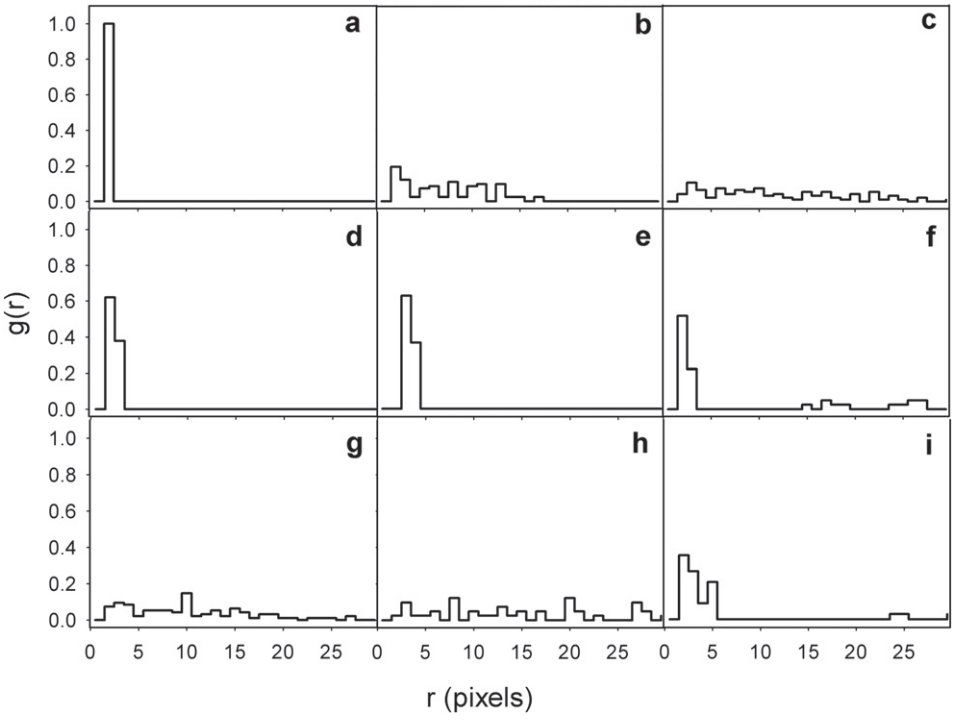

Fig. 4. Pair correlation histograms of *settlement-road* distribution, for the same simulations shown in Fig. 2. *r* is distance to road in pixels.

set alongside the road from both sides, while the distance between settlements is maintained. In the case **(d)** the distance from the settlements to the road is also included as a driving factor. With the following values: distance interval [0.75,1.8) with a probability 1.0; distance intervals from road: [0,3), [3,5) and $[5,\infty)$ km, with probabilities 1.0, 0.5, and 0.1, respectively.

In Fig. 2e the settlements could not be established near each other, because the minimum distance between them was set to 2 km, and the maximum distance between settlements was also modified to 3.6 km. The settlements are more dispersed along the road compared with Fig. 3d, and the distance between them has increased (Figs. 3e and 4e).

In the next simulation, Fig. 2f, we used the same input from case **(d)**, adding some settlements at random, with $Pset = 0.5$ and $Pthresh = 0.1$.

The simulation in case **(g)** (Fig. 2g) added the same type of vegetation in all the grid and vegetation reduction rates around settlements, also taking into account the distance between settlements, with a distance interval of [0,4) km with a probability of 1.0, and $[4,\infty)$ km with probability of 0.1.

In case **(h)** (Fig. 2h), the simulation only took into account vegetation types and vegetation reduction rate, in this case the settlements were not bounded by the distance between them or to the road (Figs. 3h and 4h). Fig. 5 shows the degradation of vegetation around settlements, for the simulation cases **(g)** and **(h)**.

The simulation in case **(i)** (Fig. 2i) considered the distance between settlements, the distance from the road and a random placement of settlements, as controlling factors. The settlements could be placed next to the road and with a minimum distance between them of 0.75 km and a maximum distance of 3.6 km, and the intervals from road distance were [0,3) km, [3,10) km and $[10,\infty)$ km.

We executed additional simulations to test the model behavior. We placed a river across the middle of the grid, in vertical and horizontal positions, and both cases gave equivalent histogram output, as expected. In another simulation, when we placed vegetation only in the upper middle of the grid, all settlements were placed in that area, according to the model structure. If we half the values of the variables *Pset* and *Pthresh* compared with the values they had in the previous simulation, the number of settlements approximately doubled in both cases (due to the randomness of the simulations the number of settlements could not be exactly doubled). In a simulation when the settlements were placed at random with no environmental factors involved, the results in the *settlements-settlement* histogram show that the settlements could be placed anywhere in the grid. This gives just a step function with an exclusion of length h (pixel size) before the step since *settlement-settlement* cannot be smaller than h.

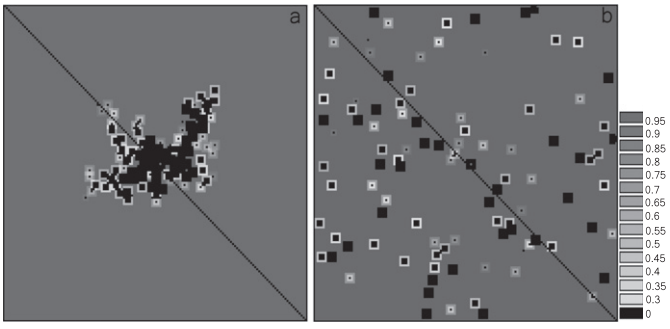

Fig. 5. **(a)** Degradation of vegetation for simulation of Fig. 2g, which shows the degradation around aggregated settlements. **(b)** Degradation of vegetation for simulation of Fig. 2h, showing degradation in sparse settlements. Color scale indicates final vegetation, resulting from decreasing the initial probability assigned to the vegetation around settlements according to defined rate. Black pixels indicate severe degradation around settlements (0.3 or lower probability). A value of 1.0 indicates no degradation.

The number N of simulations required to get stable results, considering the environmental factors in Fig. 2i, is approximately 60, as shown in Fig. 6. However, this should be controlled for different parameter sets, to ensure that a "steady state" value of residuals is reached for a given N.

3.2. Framework limitations and considerations

The computational tool presented here could offer the possibility to test different strategies to improve ecosystem management, including the addition of water provision, roads, protected woodlands, and management plans. Without such long term planning and modeling, supported and complemented by field monitoring, there is an increasing risk of environmental degradation in arid environments. The simulation code is relative small and simple, and it could be easily expanded. New environmental factors can be added by adding the input maps and the required input parameters. The input grids could be elaborated from topographic, hydrogeological, vegetation and settlement maps, often available from research and government institutions for other regions. For instance, elevation maps can be obtained using SRTM-DEM, freely available from NASA (http://www2.jpl.nasa.gov/srtm).

We note that boundary conditions have to be considered with care. Here we disregard any influence of the sites outside the simulated area, which could be unrealistic in certain cases. One future addition to the model could include a "buffer" boundary region, allowing settlements and conditions outside the simulated area to affect those inside the area.

Another caveat in the model is that the simulated settlement establishments will not give a "realistic" time evolution. However, without information about the times at which settlements were started, we cannot asses the usefulness of our model to reproduce actual, "realistic" time evolution. Neglecting settlements random processes, there would be at most one settlement established per allowed step, provided the probability at some pixel in the grid is larger than a pre-set, fix threshold value (*Pset*). A more "realistic" situation might include a time dependent threshold, responding to different environmental pressures. Another way to change the settlement dynamics would be to include an additional random process: when all probabilities have been calculated, add a settlement at a maximum probability site or into another (no so favorable) site, with the decision being taken based on a random number. This would be somewhat similar to a Metropolis Monte Carlo simulation (Allen et al., 1987) and might provide a more realistic time evolution.

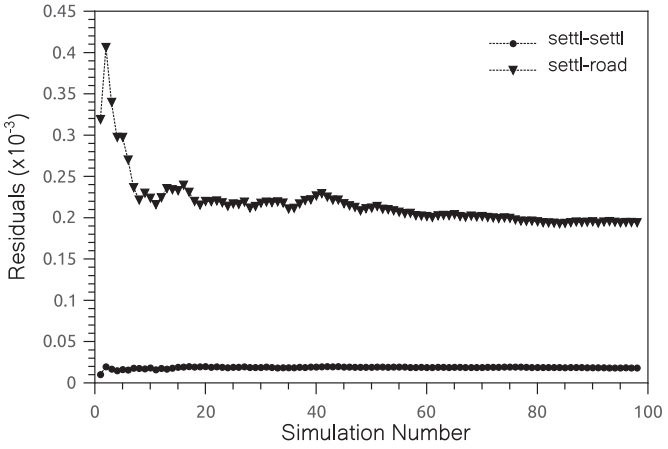

Fig. 6. Average *settlement-settlement* and *settlement-road* residuals vs simulation number, N, for simulations with the parameters for case (i) (Fig. 2i).

The time between new settlements is also fixed and global. All of this might not reflect accurately the social dynamics related to settlements. For instance, if a settlement is established at a given point by a young family, the time to establish a new settlement nearby would be related to the family growth, i.e. 20 years, and not to a global time. In addition, several family members might migrate and settle almost consecutively near this "mother" settlement. The social facet of settlement dynamics would be better described by a collection of "agents", as is often used in studies of epidemic dynamics (Macal and North, 2010) and evolution of ancient societies (Crabtree and Kohler, 2012), instead of the evolution of grid points, but this is beyond the scope of this work.

3.3. Computational benchmarks

We perform a strong scaling with OpenMP in two multi-core computers: an AMD FX-8350 4 GHz 8 cores with 32 GB of DDR3 RAM, and a node of a cluster with four AMD Opteron 6376 processors with 16 cores each running at 2.3 GHz (64 cores in total) with 128 GB of DDR3 RAM. Both multi-core computers have the same software: Slackware Linux 14.1, linux kernel 3.10.17 and GCC 4.8.2. The two OpenMP scheduling configurations, dynamic and static, were tested, achieving better results with the later. The executed simulations were performed with the simplified initial conditions configuration, for seven values of *Length* = (128, 256, 512, 1024, 2048, 4096, 8192), with a grid size of *Length*×*Length*, during 1000 steps. The simulations included three environmental factors: distance between settlements, distance to the road and vegetation degradation. Figs. 7 and 8 show the parallel efficiency calculated with the average of the wallclock time of five simulations, with an average of standard deviation of less than ~3%.

In Fig. 7 it can be seen the results of the strong scaling efficiency up to eight cores (AMD FX-8350 processor). The efficiency decreases considerably to a ~50% for the biggest grid size (8192^2 cells) with a speedup of ~4× comparing eight CPU cores to one CPU core. For the smallest grid size (128^2 cells), the efficiency is ~30% with a speedup of 2.3× comparing the eight CPU cores with one CPU core.

The same simulations were executed in one AMD Opteron 6376 processors, in up to 16 cores in one node of a cluster, the results can be seen in Fig. 8. The efficiency decreases to a ~40% in 16 cores with a ~6.5× of speedup compared with one CPU core (with grid size 8192^2 cells).

In a previous work (Millán et al., 2016) we ported the framework to CUDA and performed benchmarks with three NVIDIA GPUs

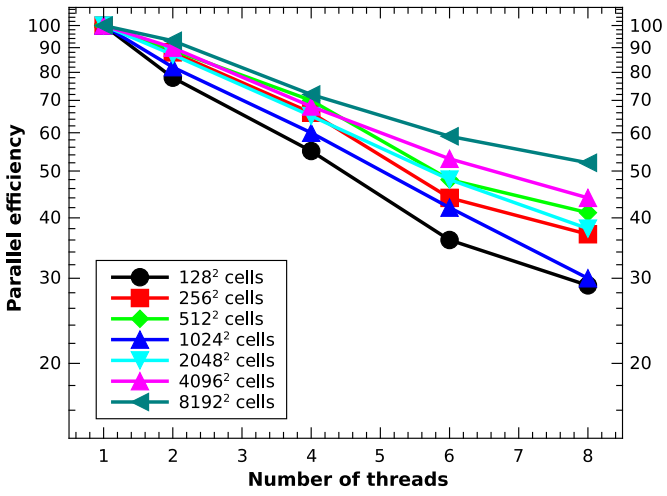

Fig. 7. Strong scaling parallel efficiency, simulations executed in an AMD FX-8350 CPU with eight cores.

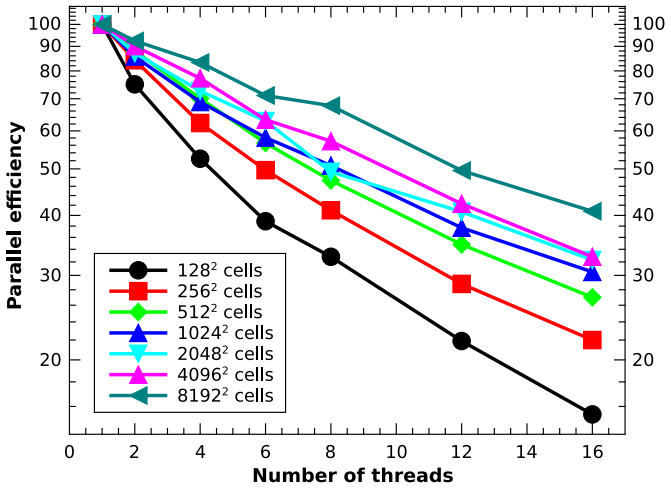

Fig. 8. Strong scaling parallel efficiency, simulations executed in an AMD Opteron 6376 CPU with 16 cores.

comparing the performance with the serial simulation. Here, we compare the results obtained in one GPU with the framework executed in parallel with OpenMP. A NVIDIA Tesla C2050 GPU (released in 2011) results in a ~8× speedup over the serial simulation executed in the AMD FX-8350 CPU (released in 2012), and a ~2.2× compared with the OpenMP execution in the same CPU using eight cores (*Length* = 4096 during 1000 steps). The same benchmarks were executed in the AMD Opteron 6376 processor (released in 2012) and in the same GPU. The framework executes ~11 times faster in the GPU than in one CPU core of the Opteron 6376, and runs ~2 times faster than 16 CPU cores. For this kind of simulation the GPUs have better performance than CPUs, although, the developing time for the GPU code is considerably greater than implementing OpenMP on the CPU serial code, which, in this case, consist in adding four OpenMP *pragmas* in the most time consuming functions of the simulation.

4. Conclusions

The model developed in this work simulates settlement establishment in arid areas, and the gradual reduction of vegetation around established settlements. The simple test cases presented here yielded the expected results, indicating that the model is indeed acting as desired. The drivers of settlement establishment are based on water availability, accessibility, and vegetation, but other factors could be included in the model. The model assumes that settlers know the conditions of the entire simulated area, and could settle in any place they consider appropriate. The decision to settle in a given place depends on the combination of different factors: the probability to settle is calculated by multiplying the partial probabilities associated to each factor, and on a stochastic component, which can be given different weights. Given the stochastic component of the model, multiple simulations should be run and averaged to obtain the final settlement distribution. Simulations under different environmental conditions, or drivers, can be compared among them or with field data using histograms of spatial distribution. With this approach, the model simulates settlement dynamics based on a knowledge of environmental factors in the region, adding stochastic contributions, characteristic of human decisions. This model would be a useful addition to the assessment of future impact of existing settlements in drylands, becoming a planning tool when considering new roads, or aqueducts.

Benchmarks performed with the parallel shared memory implementation does not result in a good efficiency for more than eight

CPU cores. An efficiency of 40% is obtained executing the simulation in 16 cores (AMD Opteron 6376). The GPU solution presented in Millán et al. (2016) results in better speedups, although the development time was considerably greater than implementing OpenMP in the serial CPU code.

Acknowledgments

We thank funding from a PFDT grant for S.G., a PICT grant from the ANPCyT (2011-2703), a PID grant (2010-2014) from SeCTyP (U.N. Cuyo), and from CONICET, Argentina.

References

Allen, M.P., Tildesley, D.J., et al. 1987. Computer Simulation of Liquids. Clarendon Press, Oxford.

Asner, G.P., Elmore, A.J., Olander, L.P., Martin, R.E., Harris, A.T., 2004. Grazing systems, ecosystem responses, and global change. Annu. Rev. Environ. Resourc. 29 (1), 261–299. http://dx.doi.org/10.1146/annurev.energy.29.062403.102142.

Axtell, R.L., Epstein, J.M., Dean, J.S., Gumerman, G.J., Swedlund, A.C., Harburger, J., Chakravarty, S., Hammond, R., Parker, J., Parker, M., 2002. Population growth and collapse in a multiagent model of the Kayenta Anasazi in long house valley. Proc. Natl. Acad. Sci. 99 (Supplement 3), 7275–7279. http://dx.doi.org/10.1073/pnas.092080799.

Barton, C.M., Ullah, I.I., Bergin, S., 2010. Land use, water and Mediterranean landscapes: modelling long-term dynamics of complex socio-ecological systems. Philos. Transact. A Math. Phys. Eng. Sci. 368 (1931), 5275–5297. http://dx.doi.org/10.1098/rsta.2010.0193.

Chan, V.W.K., 2013. Theory and applications of Monte Carlo simulations. InTech http://dx.doi.org/10.5772/45892.

Corvalan, C., Hales, S., McMichael, A.J., 2005. Ecosystems and Human Well-being: Health Synthesis. World Health Organization.

Crabtree, S.A., Kohler, T.A., 2012. Modelling across millennia: interdisciplinary paths to ancient socio-ecological systems. Ecol. Model. 241, 2–4. http://dx.doi.org/10.1016/j.ecolmodel.2012.02.023.

Dagum, L., Menon, R., 1998. OpenMP: an industry standard API for shared-memory programming. IEEE Comput. Sci. Eng. 5 (1), 46–55. http://dx.doi.org/10.1109/99.660313.

Dawelbait, M., Morari, F., 2012. Monitoring desertification in a Savannah region in Sudan using landsat images and spectral mixture analysis. J. Arid Environ. 80, 45–55. http://dx.doi.org/10.1016/j.jaridenv.2011.12.011.

Dean, J.S., Gumerman, G.J., Epstein, J.M., Axtell, R.L., Swedlund, A.C., Parker, M.T., McCarroll, S., 2000. Understanding Anasazi Culture Change Through Agent-based, Modeling, Dynamics in Human and Primate Societies: Agent-based Modeling of Social and Spatial Processes. Oxford University Press, New York., pp. 179–205.

Evans, T.P., Manire, A., F., de Castro, Brondizio, E., S., McCracken, 2001. A dynamic model of household decision-making and parcel level landcover change in the eastern Amazon. Ecol. Model. 143 (1-2), 95–113. http://dx.doi.org/10.1016/s0304-3800(01)00357-x.

Fennell, M., Murphy, J.E., Armstrong, C., Gallagher, T., Osborne, B., 2012. Plant spread simulator: a model for simulating large-scale directed dispersal processes across heterogeneous environments. Ecol. Model. 230, 1–10. http://dx.doi.org/10.1016/j.ecolmodel.2012.01.008.

Goirán, S., Aranibar, J., Gomez, M., 2012. Heterogeneous spatial distribution of traditional livestock settlements and their effects on vegetation cover in arid groundwater coupled ecosystems in the Monte Desert (Argentina). J. Arid Environ. 87, 188–197. http://dx.doi.org/10.1016/j.jaridenv.2012.07.011.

Guisan, A., Zimmermann, N.E., 2000. Predictive habitat distribution models in ecology. Ecol. Model. 135 (2-3), 147–186. http://dx.doi.org/10.1016/s0304-3800(00)00354-9.

Hosseini, S., Kappas, M., Zare Chahouki, M., Gerold, G., Erasmi, S., Rafiei Emam, A., 2013. Modelling potential habitats for *Artemisia sieberi* and *Artemisia aucheri* in Poshtkouh area, central Iran using the maximum entropy model and geostatistics. Eco. Inform. 18, 61–68. http://dx.doi.org/10.1016/j.ecoinf.2013.05.002.

Karnieli, A., Gilad, U., Ponzet, M., Svoray, T., Mirzadinov, R., Fedorina, O., 2008. Assessing land-cover change and degradation in the central Asian deserts using satellite image processing and geostatistical methods. J. Arid Environ. 72 (11), 2093–2105. http://dx.doi.org/10.1016/j.jaridenv.2008.07.009.

Kearney, M., Porter, W., 2009. Mechanistic niche modelling: combining physiological and spatial data to predict species' ranges. Ecol. Lett. 12 (4), 334–350. http://dx.doi.org/10.1111/j.1461-0248.2008.01277.x.

Kohler, T.A., Bocinsky, R.K., Cockburn, D., Crabtree, S.A., Varien, M.D., Kolm, K.E., Smith, S., Ortman, S.G., Kobti, Z., 2012. Modelling prehispanic Pueblo societies in their ecosystems. Ecol. Model. 241, 30–41. http://dx.doi.org/10.1016/j.ecolmodel.2012.01.002.

Le, Q.B., Park, S.J., Vlek, P.L., 2010. Land use dynamic simulator (LUDAS): a multi-agent system model for simulating spatio-temporal dynamics of coupled human-landscape system. Eco. Inform. 5 (3), 203–221. http://dx.doi.org/10.1016/j.ecoinf.2010.02.001.

Le, Q.B., Park, S.J., Vlek, P.L., Cremers, A.B., 2008. Land-use dynamic simulator (LUDAS): a multi-agent system model for simulating spatio-temporal dynamics of coupled human-landscape system. I. Structure and theoretical specification. Eco. Inform. 3 (2), 135–153. http://dx.doi.org/10.1016/j.ecoinf.2008.04.003.

Macal, C.M., North, M.J., 2010. Tutorial on agent-based modelling and simulation. J. Simul. 4 (3), 151–162.

Michael Barton, H.M.C., Ullah, I., 2010. Computational modeling and neolithic socioecological, dynamics: a case study from southwest Asia. Am. Antiq. 75 (2), 364–386.

Millán, E.N., Goirán, S.B., Piccoli, M.F., García Garino, C., Aranibar, J.N., Bringa, E.M., 2016. Monte Carlo simulations of settlement dynamics in GPUs. Clust. Comput. http://dx.doi.org/10.1007/s10586-015-0501-5.

NVIDIA Nvidia Cuda C Programming Guide 7.0.

Ramey, C., Fox, B., 2003. Bash Reference Manual Network Theory Limited.

Ringrose, S., Vanderpost, C., Matheson, W., 1996. The use of integrated remotely sensed and GIS data to determine causes of vegetation cover change in southern Botswana. Appl. Geogr. 16 (3), 225–242. http://dx.doi.org/10.1016/0143-6228(96)00005-7.

Wiegand, T., Moloney, K.A., 2004. Rings, circles, and null-models for point pattern analysis in ecology. Oikos 104 (2), 209–229. http://dx.doi.org/10.1111/j.0030-1299.2004.12497.x.

Wiegand, T., Moloney, K.A., Naves, J., Knauer, F., 1999. Finding the missing link between landscape structure and population dynamics: a spatially explicit perspective. Am. Nat. 154 (6), 605–627. http://dx.doi.org/10.1086/303272.

Winter-Livneh, R., Svoray, T., Gilead, I., 2010. Settlement patterns, social complexity and agricultural strategies during the Chalcolithic period in the Northern Negev, Israel. J. Archaeol. Sci. 37 (2), 284–294. http://dx.doi.org/10.1016/j.jas.2009.09.039.