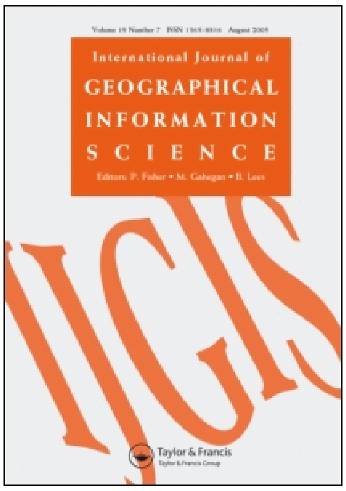

International Journal of Geographical Information Science
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713599799

Parallel cellular automata for large-scale urban simulation using load-balancing techniques

Xia Li [a]; Xiaohu Zhang [b]; Anthony Yeh [b];Xiaoping Liu [a]
[a] School of Geography and Planning, Sun Yat-sen University, Guangzhou, PR China [b] Department of Urban Planning and Design, The University of Hong Kong, Hong Kong, PR China

First published on: 02 March 2010

To cite this Article Li, Xia , Zhang, Xiaohu , Yeh, Anthony andLiu, Xiaoping(2010) 'Parallel cellular automata for large-scale urban simulation using load-balancing techniques', International Journal of Geographical Information Science, 24: 6, 803 — 820, First published on: 02 March 2010 (iFirst)
To link to this Article: DOI: 10.1080/13658810903107464
URL: http://dx.doi.org/10.1080/13658810903107464

International Journal of Geographical Information Science
Vol. 24, No. 6, June 2010, 803–820

Parallel cellular automata for large-scale urban simulation using load-balancing techniques

Xia Li[a]*, Xiaohu Zhang[b], Anthony Yeh[b] and Xiaoping Liu[a]

[a]*School of Geography and Planning, Sun Yat-sen University, Guangzhou, PR China;* [b]*Department of Urban Planning and Design, The University of Hong Kong, Hong Kong, PR China*

(*Received 11 January 2009; final version received 9 June 2009*)

Cellular automata (CA), which are a kind of bottom-up approaches, can be used to simulate urban dynamics and land use changes effectively. Urban simulation usually involves a large set of GIS data in terms of the extent of the study area and the number of spatial factors. The computation capability becomes a bottleneck of implementing CA for simulating large regions. Parallel computing techniques can be applied to CA for solving this kind of hard computation problem. This paper demonstrates that the performance of large-scale urban simulation can be significantly improved by using parallel computation techniques. The proposed urban CA is implemented in a parallel framework that runs on a cluster of PCs. A large region usually consists of heterogeneous or polarized development patterns. This study proposes a line-scanning method of load balance to reduce waiting time between parallel processors. This proposed method has been tested in a fast-growing region, the Pearl River Delta. The experiments indicate that parallel computation techniques with load balance can significantly improve the applicability of CA for simulating the urban development in this large complex region.

Keywords: parallel computing; load-balancing; GIS; cellular automata; urban simulation

1. Introduction

Parallel computing, using multiple computer resources simultaneously to solve one problem, has become an important research direction of computer science, which has achieved significant progress recently. Through applying parallel computing techniques, large computational tasks are usually divided into smaller ones, which can be distributed to a single computer in a workstation cluster and be calculated concurrently. Parallel computing provides a powerful tool for high-performance computing that can alleviate the current bottleneck problems in many geographical applications. Actually, explorations of parallelism in GIS were started about 10 years ago with a special issue in the International Journal of GIS 1996 as a milestone (Densham and Ding 1996, Clematis *et al.* 2003).

Research has indicated that parallel computing is effective for solving large-scale geographical models (Turton and Openshaw 1998), such as air pollution diffusion (Owczarz and Zlatev 2002), combinatorial resource allocation (Morales *et al.* 2000), and visualization and modeling (Vokorokos 2005). Parallel computing is also very attractive

*Corresponding author. Emails: lixia@mail.sysu.edu.cn;lixia@graduate.hku.hk

ISSN 1365-8816 print/ISSN 1362-3087 online
© 2010 Taylor & Francis
DOI: 10.1080/13658810903107464
http://www.informaworld.com

for facilitating high-performance computing with distributed data grids in the emerging cyber infrastructures for geosciences (Zhang *et al.* 2007). For example, parallel computing has been used in the studies of geodynamo (Glatzmaier and Roberts 1995), seismic wave propagation (Komatitsch and Tromp 2002), and mantle convection (Kameyama and Yuen 2006).

The computational capabilities of cellular automata (CA) have been extensively documented since these models were first proposed by von Neumann and Ulam in the 1940s (Bandini *et al.* 2001). These models have been applied to the simulation of fluid dynamics, crystal growth, and biological processes (Goles 1989). Recent studies also show that CA are also very suitable for simulating a variety of complex geographical phenomena, such as lava flows (Barca *et al.* 1993), rock fracturing (Norman *et al.* 1991), weather modeling, wildfire propagation (Clarke *et al.* 1994, Karafyllidis and Thanailakis 1997), vegetation successions (Atkinson *et al.* 1998, Bandini *et al.* 2001), forest dynamics (Lett *et al.* 1999), epidemic propagation (Sirakoulis *et al.* 2000), landscape changes (Soares-Filho *et al.* 2002), settlement development (Deadman *et al.* 1993), and urban dynamics and land use changes (White and Engelen 1993). It is attractive that emerging behaviors and global patterns of these phenomena can be simulated by the local interactions of CA (White and Engelen 1993, Batty and Xie 1994, Wu 2002). The simulation of complex urban dynamics and land use changes is among the most successful examples of CA applications (Clarke *et al.* 1997, Li and Yeh 2004, Pontius and Malanson 2005, Li and Liu 2006).

CA are a kind of discrete models because they operate on discrete cells, states, and time. Discrete cells are defined by dividing the space into many basic processing units so that transition rules can be applied to each cell. The data volume of the simulation is related to cell sizes. The discrete state of each cell is updated by iterations (discrete time). The use of discrete space, states, and time in CA perfectly fits the features of digital computers and raster GIS formats. Transition rules, which are the core of CA, determine state conversion for each cell. Various kinds of spatial variables are usually prepared from GIS for defining transition rules (Li and Yeh 2002). The computation of CA is very intensive for solving practical simulation problems because of using a rich set of spatial data. Simulating natural phenomena usually involves large computational tasks by using fine resolutions of spatial data and time steps, and large numbers of spatial variables (Costanza and Voinov 2003). Traditionally, sequential implementation of CA often encounters the speed bottleneck when simulating large and complex regions.

The regular topologies of CA perfectly fit straightforward parallelization requirements. The locality of interactions of CA is another feature that allows the implementation of these models on parallel computers (Bandini *et al.* 2001). Fixed nondeterministic local rules can be applied to each cell in a parallel way. CA have been considered as one of the first parallel computing abstract models (Cannataro *et al.* 1995).

There have been an increasing number of studies on the introduction of parallel techniques to CA in recent years. Indeed some general-purpose parallel CA models have been developed and brought to real-world application. Among them the most significant systems are CAMEL (Cannataro *et al.* 1995), CAPE (Norman *et al*. 1991), and NEMO (Hutchinson *et al.* 1996). However, these systems cannot be directly used to solve urban simulation effectively. Parallel urban simulation is quite unique in data decomposition because a large region usually has heterogeneous or polarized development patterns. The challenge is to arrange equal workloads among processors according to spatial heterogeneity. Few studies have been carried out on developing urban CA for simulating large complex regions under the parallel environment. The computation time will become a bottleneck if sequential CA is applied to this kind of simulation. The development of urban CA on parallel computers can enhance the capability of CA for solving practical problems.

In this study, a parallel CA model is proposed for facilitating large-scale urban simulation. It is based on the Message Passing Interface (MPI) and load-balancing techniques. These techniques provide powerful tools for developing high-performance parallel and distributed applications of using urban CA. 'Ghost cells' are used to exchange state information across the boundaries between decomposed parts. Moreover, a line-scanning method of load balance is proposed to reduce the waiting time between parallel processors for improving the overall simulation efficiency.

2. Parallel computing for implementing urban cellular automata

2.1. Urban cellular automata

Although urban CA may have different model structures, the general form of these models can be given as follows (Batty and Xie 1994, Li and Yeh 2000):

$$S_{ij}^{t+1} = f\left(S_{ij}^{t}, \Omega\right) \tag{1}$$

where S is a set of possible discrete states at location ij, Ω is the neighborhood of all cells providing input values to the function f, and f is the transition function (transition rules) that determines state conversion from time t to $t + 1$.

Explicit transition rules are usually required for programming urban CA instead of using this general structure. A convenient way to represent transition rules is based on transition probabilities or transition potentials. For example, the following rule-based structure is used to estimate development probabilities (Batty 1997):

IF any cell $\{x \pm 1, y \pm 1\}$ is already developed
THEN $P_{ij} = \sum_{xy\in\Omega} P_{xy}/8$
&
IF $P_{ij} >$ some threshold value
THEN cell ij is developed with some other probability ρ_{ij}

where P_{ij} is the development probability for cell ij and xy represents all the cells that are from the Moore neighborhood Ω including the cell ij itself.

Urban CA usually have a more complex form of transition rules than classical CA. It is because urban CA need to include various spatial variables and constraints for simulating realistic urban dynamics. A convenient way is to use multicriteria evaluation (MCE) techniques to estimate development probabilities. MCE can effectively capture the combined influences of various spatial factors that govern different development regimes in urban simulation (Wu and Webster 1998). Development probabilities are determined by a combined evaluation score r_{ij}. A nonlinear transformation is used to discriminate the simulation patterns by using the following equation:

$$p_{ij}^{t} = \phi\left(r_{ij}^{t}\right) = \exp\left[\alpha\left(\frac{r_{ij}^{t}}{r^{\max}} - 1\right)\right] \tag{2}$$

where p_{ij}^{t} is a dispersion parameter ranging from 0 to 1, r_{ij}^{t} is the combined evaluation score at location ij, and $r^{\max}$ is the maximum value of r_{ij}^{t}.

The composite evaluation score (r^t_{ij}) is calculated based on the following linear equation:

$$r^t_{ij} = a_0 + a_1 x^t_1 + a_2 x^t_2 + \cdots + a_m x^t_m + \cdots + a_M x^t_M \tag{3}$$

where a_0 is the constant, x^t_m is a spatial variable representing a driving force for urban development, and a_m is the parameter (weight) representing the importance of this variable in calculating the development probability.

A modification to this MCE-CA model is to transform it into a logistic form so that the calibration is possible (Wu 2002):

$$p^t_{ij} = \frac{\exp\left(-r^t_{ij}\right)}{1 + \exp\left(-r^t_{ij}\right)} = \frac{1}{1 + \exp\left(-r^t_{ij}\right)} \tag{4}$$

Urban development is also subject to neighboring influences, a series of physical constraints, and some uncertainties. By incorporating all these factors, the above equation is further revised as follows:

$$p^t_{ij} = (1 + (-\ln\gamma)^{\alpha}) \frac{1}{1 + \exp\left(-r^t_{ij}\right)} \times f\left(\Omega^t_{ij}\right) \times \mathrm{con}\left(s^t_{ij}\right) \tag{5}$$

where γ is a stochastic factor ranging from 0 to 1, α is a parameter to control the stochastic degree, $f(\Omega^t_{ij})$ is the development intensity in the neighborhood of Ω_{ij}, and $\mathrm{con}(s^t_{ij})$ is the total constraint score ranging from 0 to 1.

At each iteration, p^t_{ij} is compared with a threshold value to determine whether a non-urbanized cell will be converted into an urbanized cell:

$$S^{t+1}_{ij} = \begin{cases} \text{Converted,} & p^t_{ij} \geq T \\ \text{Non Converted} & p^t_{ij} < T \end{cases} \tag{6}$$

where T is a threshold value.

In this study, parallel urban simulation is illustrated based on the logistic-CA model. Other types of CA can be implemented by using the same architecture because parallel computing can be independent of application models. This flexibility is desirable as it enables the modification of CA models without altering the parallel computing structures.

2.2. *MPI for parallel CA*

Dividing the study area into a number of sub-regions is the first step to implement parallel urban simulation. A distinct feature of parallel CA is that the determination of state conversion for a cell requires the input of its neighborhood's data. Therefore, a decomposed part assigned to a computing unit must have overlapping belts with its neighboring parts. The overlapping is related to the matrix size (e.g., 3 × 3, 5 × 5, 7 × 7, 9 × 9, and 11 × 11) in defining the local influences (interactions).

The state of a central cell at time $t + 1$ is related to the states of this cell and its neighbors at time t. In the implementation of parallel CA, the whole study area needs to decompose into many parts. Cells are distributed by belts or folds over several threads running on

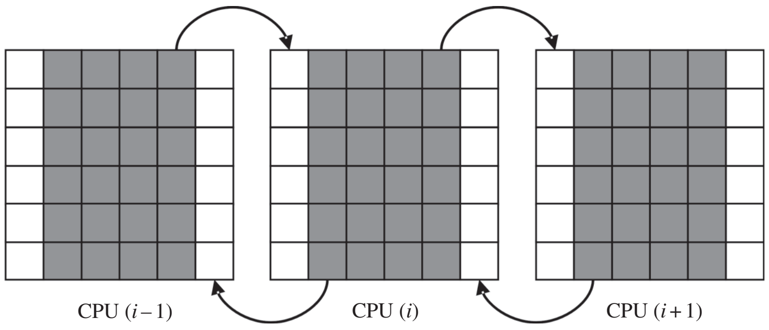

Figure 1. Implementation of ghost-cell techniques.

different computers (Mazzariol *et al.* 2000). Every thread carries out these tasks: (1) sending and receiving the states of neighboring cells and (2) calculating the current state of central cell.

Neighboring information is received during the computation of this step and sent during the computation of the next step. These overlapping belts are called ‘ghost cells’ (Figure 1). In parallel urban simulation, the boundaries must be updated at each time step to address neighborhood influences. This boundary update process involves very intensive communications because there is a large number of ‘ghost cells’. The number is related to the neighborhood size in defining CA. These ‘ghost cells’ should be updated at every time step to exchange the most recent values of the corresponding active cells on neighboring processors.

One of the most common parallel programming environments is based on the MPI for the exchange of information between neighboring processors (Morales *et al.* 2000). A very important goal of MPI is to provide a widely portable and efficient programming library without sacrificing performance (Al-Tawil and Moritz 2001). MPI has usually been adopted in the parallel computing environment (Hempel and Walker 1999). MPI provides the two key aspects of parallel programming: (a) synchronization of processes and (b) read/write access for each processor to the memory of all other processors (Chan *et al.* 2003). MPI offers routines to interchange information between tasks, asynchronous blocking send, asynchronous blocking receive, and non-blocking receive functions (Morales *et al.* 2000).

In this parallel urban CA, MPI is used to implement the boundary update process. The point–point data exchange is accomplished by using the functions of MPI_Send and MPI_Recv. The message exchanges are only performed for these cells lying on partition boundaries. For example, extra caching columns are added at the right edge and left edge of each sub-region to receive and send data to its neighboring processor (Figure 1). The process ID of a sub-region is recorded by MPI_Comm_rank. The process IDs of its left and right neighbors are stored by LEFT_RANK and RIGHT_RANK.

The variables defined for each process are listed as follows:

```
int SIZE = MPI_Comm_size() //get the total number of processes
int rank RANK = MPI_Comm_rank() //get current process ID of a sub-region
int LEFT_RANK = (RANK – 1 + SIZE)%SIZE //get process ID of its Left Neighbor
int RIGHT_RANK = (RANK – 1 + SIZE)%SIZE //get process ID of its Right Neighbor
```

The pseudo codes for the ghost-cell exchanging algorithm are illustrated as follows:

```
IF (RANK(current process) has LEFT_RANK(Left Neighbor)) {
   MPI_Send(Send) ghost cells To LEFT_RANK(Left Neighbor)
   MPI_Recv(Receive) ghost cells From LEFT_RANK(Left Neighbor)
}
IF (RANK(current process) has RIGHT_RANK(Right Neighbor)) {
   MPI_Send(Send) ghost cells To RIGHT_RANK(Right Neighbor)
   MPI_Recv(Receive) ghost cells From RIGHT_RANK(Right Neighbor)
}
```

2.3. Load-balancing techniques for distributing spatial data among processors

There are a number of data decomposition schemes for distributing the spatial data among processors. First, there are two major types of data decomposition: (1) division based on equal areas and (2) division based on equal workloads. The first option is efficient only if the study area is homogenous. In fact, this option cannot lead to equal workloads among processors in most cases because of heterogeneous distribution of geographical phenomena in nature (e.g., the existence of development clusters) (Figure 2). The second option is to address this problem by evenly distributing computation loads among processors.

Each major type can have three sub-types of the decomposition: (1) dividing the study region into *n* horizontal folds (rowwise decomposition); (2) dividing the study region into *n* vertical folds (columnwise decomposition); (3) dividing the study region into *n* rectangular sub-regions (chessboard decomposition) (Quinn 2004) (Figure 3). It is straightforward and easy to implement the first two sub-types. Alternately, the chessboard decomposition, which is relatively complicated, may be more appropriate for capturing the heterogeneity of geographical features in large complex regions than the first two sub-types. It is because the chessboard decomposition concerns not only the horizontal inequality but also the vertical inequality. It is expected that the chessboard decomposition can achieve more efficient parallel computing under the situations of polarized or complex development patterns.

2.3.1. Division based on equal areas

A simple method of data decomposition for implementing parallel CA is to divide a study region into equal areas. This method is ideal if the computation time of each fold is approximately the same. The simulation will be implemented by *p* parallel threads for a region of *m* columns $\times$ *n* rows. In the partitioning by columns, each thread will be in charge

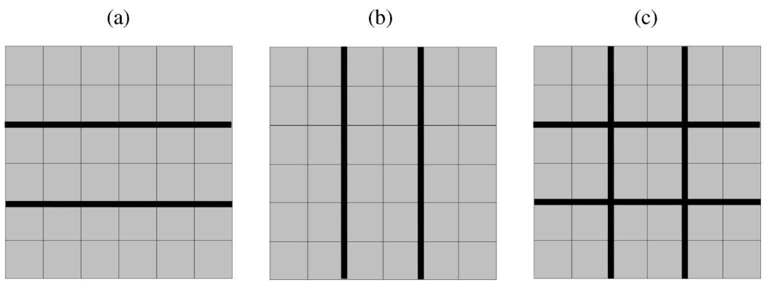

Figure 2. Development clusters in the study areas.

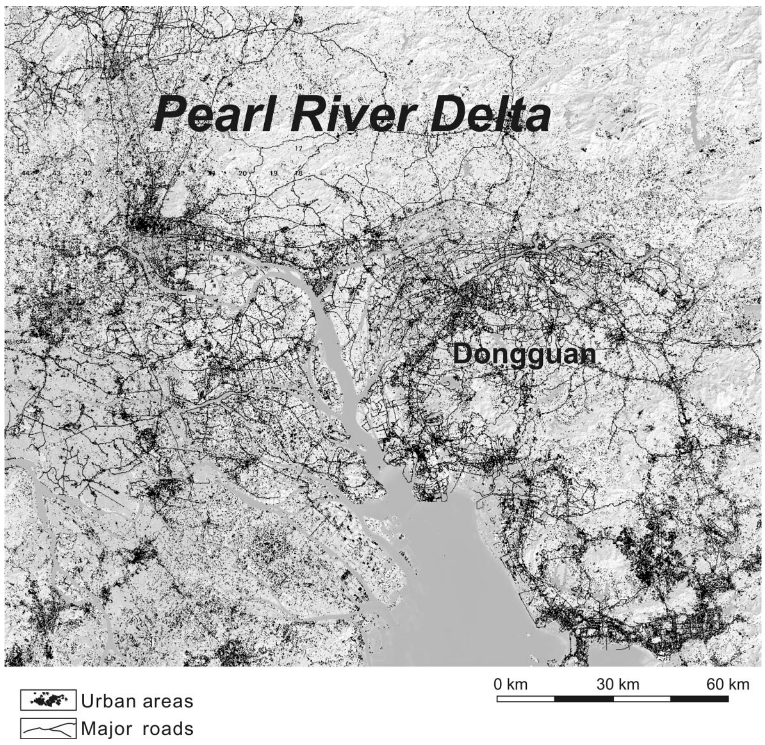

Figure 3. Three methods for data decomposition: (a) divided by rows, (b) divided by columns, and (c) divided by chessboard.

of *m/p* neighboring columns for data processing. In the partitioning by rows, however, each thread will be in charge of *n/p* neighboring columns for data processing.

The partitioning by columns or rows is straightforward and the communications between neighboring processors are quite simple. These two sub-types of data decomposition only consider the communications between the right neighbor and the left neighbor (dividing by columns) or between the upper neighbor and the lower neighbor (dividing by rows). However, the chessboard division involves very complex communications because the data exchange is carried out between eight neighbors (Figure 4).

For each type of decomposition, MPI is used to update the states of cells at the boundary. The boundary update is convenient for the first two sub-types. The decomposed data will be sent to each processor after the region has been divided based on equal areas. The exchange of neighboring information takes place at each iteration of simulation. Two steps of data exchange are required for completing the state update of the boundary cells (Figure 5). For the chessboard division, the update of neighboring information is quite complicated. The shift algorithm is used to reduce the total number of communications (Palmer and Nieplocha 2002). This algorithm first exchanges the boundary data in the horizontal direction and then exchanges the boundary data in the vertical direction (Figure 6).

2.3.2. Division based on equal workloads

A critical issue with the equal-area decomposition is its inefficiency in communications between different processors under complex situations. Some processors have to wait longer

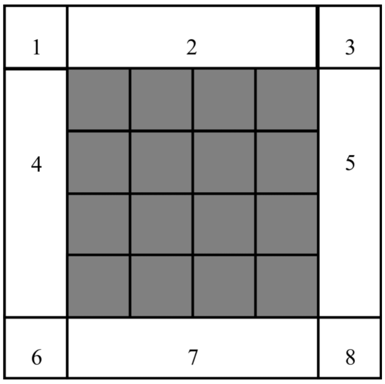

Figure 4. Complex communications for the chessboard decomposition.

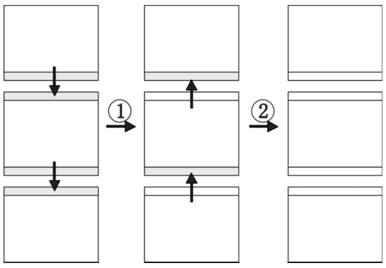

Figure 5. Two steps of data exchange for completing the state update of the boundary cells.

for the inputs from other processors if the workload is not the same. This can result in the increase in the total computation time for the whole simulation. For the equal-area decomposition, the imbalance of workloads between processors is severe if there is inhomogeneous distribution of geographical features, such as the uneven distribution of rivers, steep mountains, and ecological protected lands. The cells that belong to the above land use types almost do not need time to compute because they cannot be urbanized in the simulation. A simple way for equally distributing the computation loads among processors is to remove these unavailable cells in dividing the region. Moreover, there is another type of unavailable cells (already urbanized land) that cannot be determined at the beginning of simulation. The number of urbanized cells will grow as the simulation continues.

The computation time for processing available cells is the same because the transition rules are uniformly applied to these cells. Although the unavailable cells (e.g., rivers, steep mountains, ecological protected lands, and urbanized land) are excluded from the conversion, computation time is still required for identifying these unavailable cells in the programming. The computation time for available cells and unavailable cells can be estimated by experiments. The total numbers of these two types of cells can also be calculated at each iteration.

The total computation time at each iteration is then determined by the following equation:

$$T = n_{av} \times T_{av} + n_{un} \times T_{un} \tag{7}$$

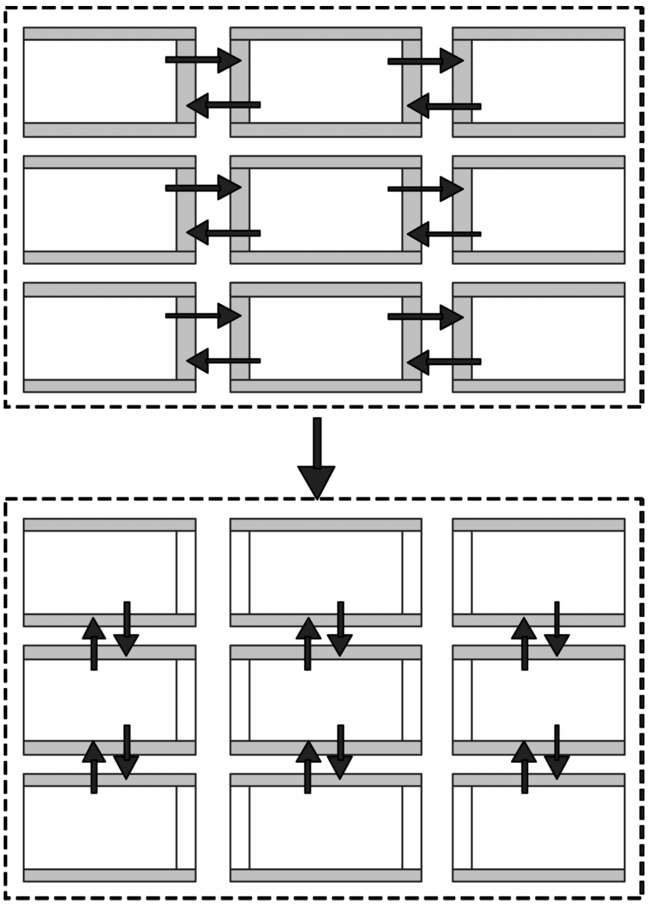
Figure 6. The shift algorithm for updating neighboring information of the chessboard division.

where n_{av} and n_{un} are the total numbers of available (e.g., agricultural land) cells and unavailable (e.g., water, steep hills, and urbanized land) cells. T_{av} and T_{un} are the computation time for available cells and unavailable cells, respectively.

If there are k parallel nodes (processors) for dividing the region into k sub-regions, each sub-region should be assigned with approximate workloads for efficient computation. The optimal computation time from the balanced assignment is expected to be T/k for each processor.

There is a dilemma for keeping the load balance between processors for the whole simulation. The optimal partition for the balance can be found at a particular time. However, this optimal partition will have imbalanced workloads at other time. It is impossible to find a fixed data division that can keep the load balance all the time because of the increase in the number of urbanized cells. The idealized load balance can only be obtained by using dynamic data divisions that require the re-distributing of the spatial data to each processor at each time. However, this dynamic balance cannot yield an optimal solution because the re-distributing itself needs a long waiting time. Therefore, a fixed division has to be adopted although this cannot guarantee the load balance all the time.

A simple way is to determine the optimal division of load balance at the beginning of simulation. This study proposes a line-scanning method to obtain the load balance between processors. For the rowwise decomposition, the line-scanning is carried out to calculate the computation time of each row after the total numbers of these two types of cells (available cells and unavailable cells in terms of urban development) have been obtained. After the total computation time of all the rows has been obtained, the division of k folds can then be determined by ensuing each fold with equal computation time. The same procedure can be used to decompose the data for the columnwise decomposition.

Optimal chessboard division is not straightforward because of its complex structure. This problem is solved by using an approximation procedure that determines a total of $k_r \times k_c$ folds by two steps: (1) obtaining the optimal k_r horizontal folds by using the same procedure for the rowwise decomposition and (2) determining the optimal k_c vertical folds for each horizontal fold using the same way. After these folds have been obtained, fold (i, j) is assigned to k'th processor (where $k' = (i-1) \times k_c + j$, $k_r \times k_c = k$, $1 \le i \le k_r$, $1 \le j \le k_c$, and $1 \le k' \le k$).

3. Application and results

3.1. The study area

This proposed parallel CA was applied to the simulation of a fast-growing region in South China. Experiments were carried out to simulate the urban dynamics for both the Dongguan city and the whole Pearl River Delta (Figure 7). This region witnessed fast urban expansion because of the rapid economic development in the last three decades (Li and Yeh 2004). Urban simulation using CA could provide useful information for urban planning and management in this region. However, data size has been a bottleneck for urban simulation when the study area is large and many spatial variables are involved. In this study, the cell size is 28.5 m for the initial land use layers that are classified from Landsat TM images. There are a total of $2693 \times 1864 = 5{,}019{,}752$ cells for Dongguan and $6084 \times 6756 = 41{,}103{,}504$ cells for the

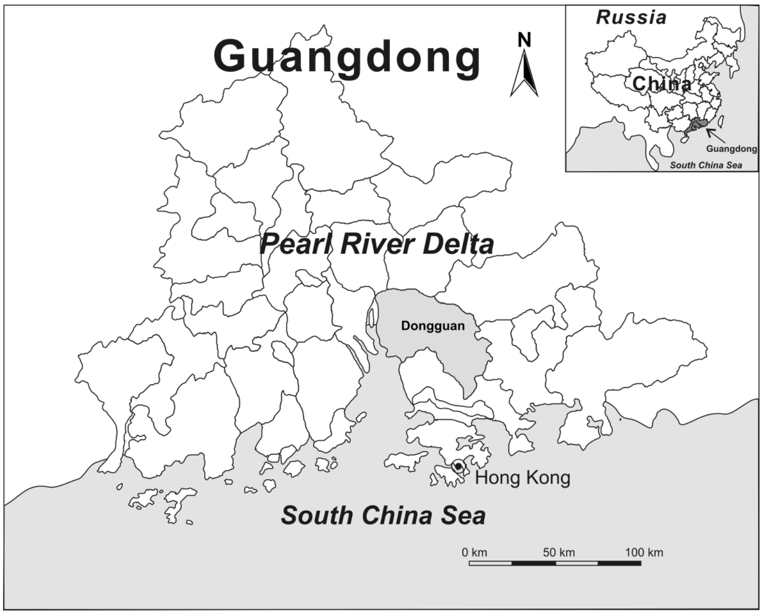

Figure 7. The study area in the Pearl River Delta.

whole region. The computation is intensive by applying transition rules to each cell iteratively for the simulation of urban growth in this large region. Resolution degradation may be used to reduce the data size, but this will cause the loss of many small land use parcels. Parallel computing is expected to alleviate the computation time problem if high-resolution data are used for urban simulation.

In this study, Landsat TM images dated in 1988, 1993, 1997, and 2004 were used to train and verify the proposed parallel CA model. The first two TM images were used to derive the transition rules whereas the remaining images were used to validate the simulation results. The spatial information included land use types, topography, and various kinds of proximity variables (e.g., distances to roads, expressways, railways, and town centers). This parallel model was compared with the sequential model based on the same logistic-CA form. Other types of CA can be implemented by using the same parallel architecture because it is independent of CA model structures.

3.2. Data decomposition schemes and results

The proposed parallel CA was programmed by using C language and MPI in a Linux environment (Fedora6.0). The switch, Tp-Link TL-SF1024V, was used to connect eight computers with the Ethernet speed of 100 Mbps. Each of these computers had a Intel CPU of Core^{TM} 2 Duo (E6600) 2.40 GHz and a 2G memory.

As the rowwise decomposition is just different from the columnwise decomposition in terms of the direction, this study only selects the rowwise decomposition for the experiments. Therefore, there are two sub-types of data decomposition: the rowwise decomposition and the chessboard decomposition. Figure 8 shows these two sub-types of data decomposition: equal areas and equal workloads.

For the implementation of data decomposition of equal workloads, the average computation time for processing an available cell (T_{av}) and an unavailable cell (T_{un}) was determined by using the line-scanning method. These values were obtained by measuring the actual time for processing these two types of cells (Table 1).

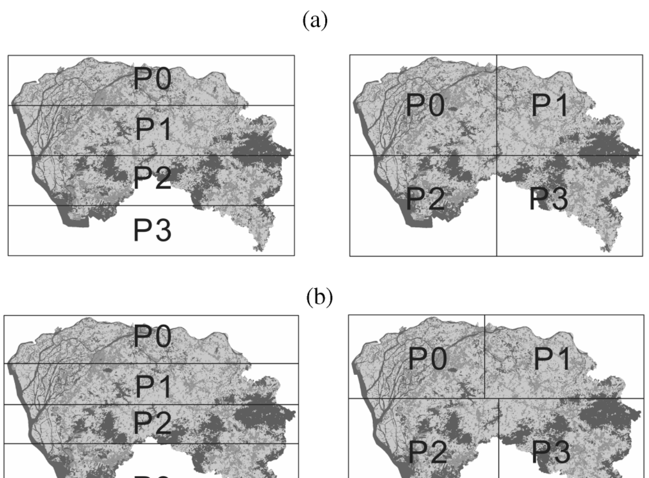

Figure 8. Rowwise decomposition, columnwise decomposition, and chessboard decomposition.

Table 1. The average computation time for processing an available cell (T_{av}) and an unavailable cell (T_{un}).

Neighborhood	T_{av} (s)	T_{un} (s)
Dongguan		
3 × 3	0.003674	0.000127
5 × 5	0.004550	0.000128
7 × 7	0.006082	0.000128
9 × 9	0.007916	0.000127
11 × 11	0.010132	0.000127
The Pearl River Delta		
3 × 3	0.003522	0.000127
5 × 5	0.004637	0.000127
7 × 7	0.006163	0.000128
9 × 9	0.007815	0.000127
11 × 11	0.010058	0.000128

This proposed parallel CA model was then applied to the simulation of Dongguan and the whole Pearl River Delta in the period of 1988, 1993, 1997, and 2004. Landsat TM images were also used to obtain the actual urban development for these years. Figure 9 shows the goodness of fit between the actual and simulated urban development. The overall accuracy was 85.3% according to the cell-by-cell comparison.

The execution time, speedup, and efficiency can be used as the metrics for evaluating the performance of the proposed parallel CA. In parallel computing, speedup refers to how much a parallel algorithm is faster than a corresponding sequential algorithm. The indicator of speedup is calculated as follows:

$$S_k = \frac{T_s}{T_k} \tag{8}$$

where k is the number of processors, T_s is the execution time of the sequential algorithm (one processor), and T_k is the execution time of the parallel algorithm with k processors.

A computing task can be divided into two parts: the part unavailable for parallel computing and the part available for parallel computing. The computing time can be calculated as follows:

$$T_s = T_n + T_p \tag{9}$$

where T_n is the computing time of the part unavailable for parallel computing and T_p is the computing time of the part available for parallel computing.

For k processors, the parallel computing time is given as follows:

$$T_k = T_{nk} + \frac{T_{pk}}{k} + T_{\text{comm}} \tag{10}$$

where T_{comm} is the time for communication.

The efficiency is represented as follows:

$$E_k = \frac{S_k}{k} = \frac{T_s}{kT_k} = \frac{T_{nk} + T_{pk}}{\left(T_{nk} + \frac{T_{pk}}{k} + T_{\text{comm}}\right) \times k} = \frac{T_{nk} + T_{pk}}{T_{nk} \times k + T_{pk} + T_{\text{comm}} \times k} \tag{11}$$

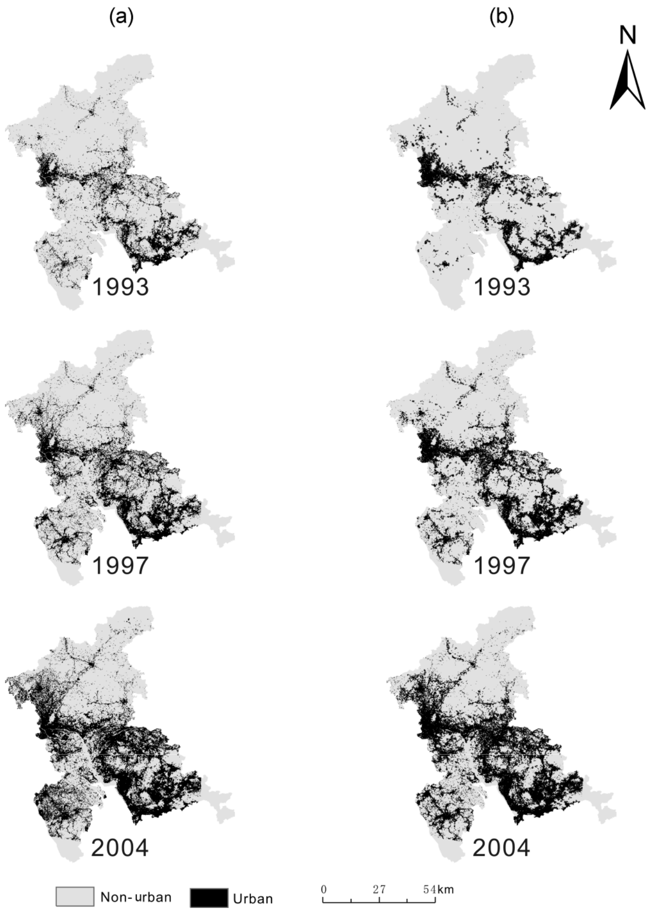

Figure 9. Simulating urban dynamics of the whole Pearl River Delta using parallel CA: (a) actual and (b) simulated.

It is easy to find that the efficiency (E_k) of parallel computing falls within the range of [0,1]. The efficiency will drop as the number of processors (k) increases because of the longer communication time between processors. There are four kinds of data decomposition schemes, the rowwise division of equal areas (RDEA), the rowwise division of equal workloads (RDEL), the chessboard division of equal areas (CDEA), and the chessboard division of equal workloads (CDEL). The execution time of parallel urban simulation is related to the neighborhood size (Figure 10). The use of larger neighborhood size will significantly increase the execution time for these four decomposition schemes.

Figure 11 shows the assessment results for the performance of parallel CA. The neighborhood size is 11×11 cells. RDEL has the least execution time. Its execution time with eight processor is only 13.8% and 17.2% of that with one processor (ordinary CA simulation) for Dongguan and the Pearl River Delta, respectively. RDEA and CDEA have the longest execution time for Dongguan and the Pearl River Delta, respectively. Therefore, data decomposition based on equal areas is not an efficient method for parallel urban simulation. Instead, data decomposition based on equal workloads can help us to reduce the execution time.

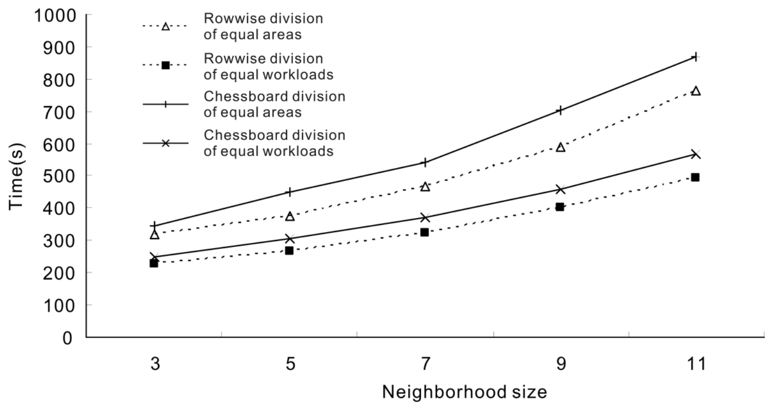

Figure 10. The execution time and the neighborhood size for the urban simulation in the Pearl River Delta.

Similar results are obtained by using the indices of speedup and efficiency. The indicator of speedup clearly shows that the performance of simulating larger areas is poorer because these lines lie more far away from the ideal speedup line (Figure 11). The indicator of efficiency also indicates that the simulation of larger regions has poorer performance. Although the use of more processors can reduce execution time, the efficiency of parallel computing decreases as more processors are used according to the experiments. This result coincides with the Equation (11).

The parameters of *Initialization Time*, *Scatter Time*, *Barrier Time*, *Computation Time*, and *Communication Time* can provide additional information about the performance of the parallel CA. *Initialization Time* indicates the time required in the initialization phase. In this stage, the master node (usually node 0) of the parallel CA retrieves all the data sets that will be processed in running time, whereas other nodes are just waiting in idle mode. *Scatter Time* indicates the time for scattering the data to corresponding nodes. The master node sends the data to each node one by one in this stage. *Barrier Time* refers to the waiting time because there are early and late arrival nodes. When every node receives the corresponding data, the system starts actual parallel computing routine. In this stage, the parallel CA iterates many generations for the simulation of urban dynamics. *Computation Time* is the total time required in the intensive computing phase. *Communication Time* is the sum of the time for communication in all iterations. Among these parameters, *Computation Time* is undoubtedly the most concerned one because its distribution in the parallel nodes has a significant impact on the efficiency.

Figure 12 shows the values of these time parameters by using eight processors for various decomposition schemes. *Computation Time* is unevenly distributed among the nodes for both rowwise and chessboard division. This will lead the parallel CA to work inefficiently. The decomposition by equal workloads induces relatively even distribution of *Computation Time* in each node, thus making effective use of the computation resources. The CDEL seems to be effective in dealing with heterogeneous development patterns, but it cannot produce the results better than RDEL. It is because CDEL needs extra time for communication. This can be clearly found in Figure 12d.

4. Conclusion

Parallel computing has been increasingly used to solve hard computation problems in geosciences. Parallel computing is a powerful tool for handling many geographical problems

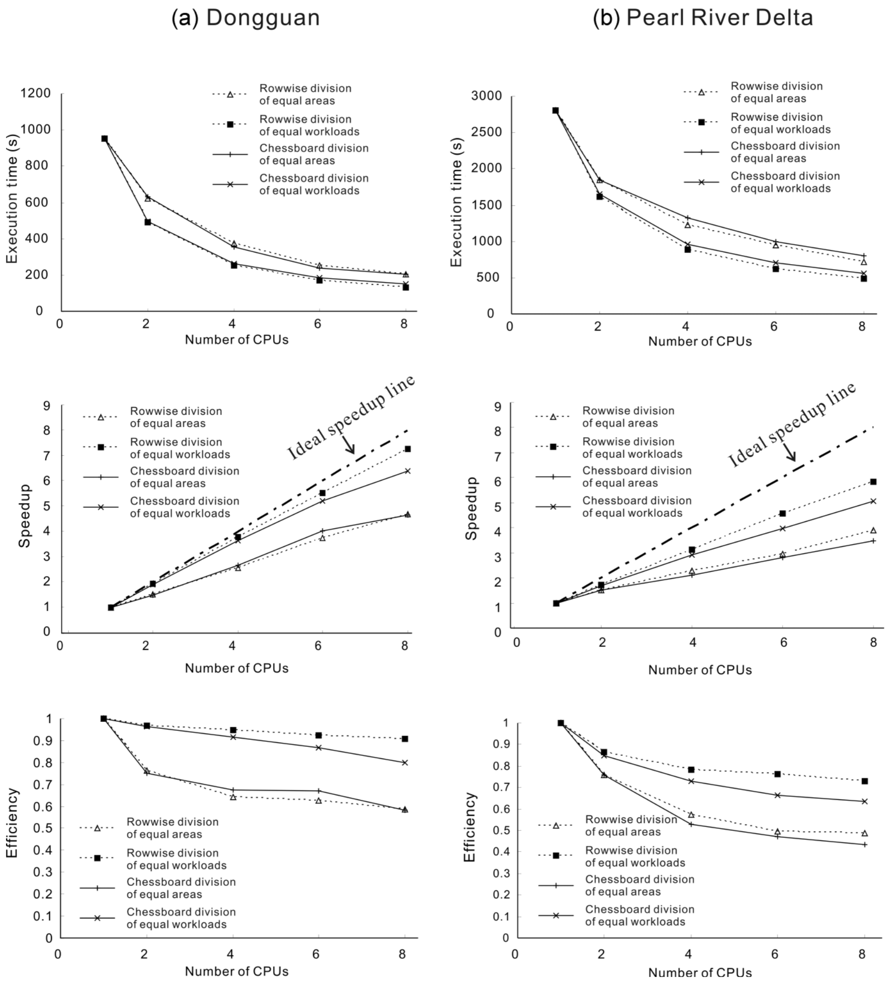

Figure 11. Computation time, speed up, and efficiency of the parallel CA related to the number of CPUs.

that involve large volumes of spatial data. This technique can be used to improve the performance of urban CA for simulating large regions. Urban CA is born to be parallel because transition rules are unanimously applied to each cell in the simulation. This study demonstrates that parallel computing can effectively reduce computation time in urban simulation. Parallel computing is especially useful when various types of GIS data, such as land use, topography, facilities, transportation, and population, are used as the inputs to urban CA.

A number of techniques should be incorporated to improve the performance of parallel urban CA. This study has demonstrated that parallel computing with 'ghost-cell' and load-balancing techniques can significantly reduce the computation time in urban simulation. Unlike classical CA, urban CA are quite unique because workload balance is important for implementing parallel computing effectively. The proper distribution of workloads can affect the efficiency in parallel urban simulation. There are two major methods of distributing spatial data among processors for parallel computing.

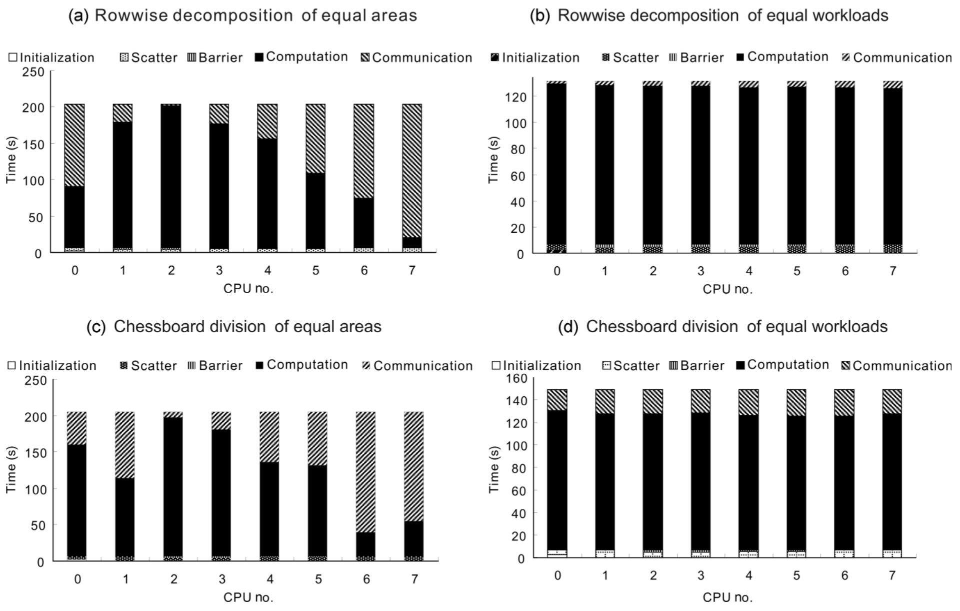

Figure 12. Initialization time, scatter time, barrier time, computation time, and communication time of the eight processors for various decomposition schemes.

Data decomposition can be based on equal areas or equal workloads. The first option is relatively simple for implementation. It can have satisfactory results if the study area is homogeneous. However, it will lead to bias for distributing computation loads among processors if the study area is complex in terms of land use patterns or development patterns. Data decomposition based on equal workloads can provide an effective way to deal with the load-balancing problems. This study proposes a line-scanning method to obtain optimal load balance. Experiments indicate that this method can significantly reduce the waiting time among processors.

Moreover, a study region can be divided into horizontal folds, vertical folds, or rectangular sub-regions (chessboard decomposition). The first two divisions assume that the study area is homogenous in terms of geographical features. In particular, this type of divisions is useful if the growth patterns are approximately evenly spread out in the region. It seems that the chessboard decomposition can have more efficient computation if the region has heterogeneous or polarized development patterns. However, our study indicates that the CDEL cannot produce better results than RDEL because the former needs extra time for communication. The execution time of RDEL with eight processors is only 13.8% and 17.2% of that with one processor (ordinary CA simulation) for Dongguan and the Pearl River Delta, respectively. Therefore, parallel computing can significantly save the time of urban simulation for large regions by properly distributing workloads across processors. The further studies could be focused on the consideration of process parallelism.

Acknowledgements

This study was supported by the Hi-tech Research and Development Program of China (863 Program) (grant no. 2006AA12Z206), the Key National Natural Science Foundation of China (grant no. 40830532), and the National Outstanding Youth Foundation of China (grant no. 40525002).

References

Al-Tawil, K. and Moritz, C.A., 2001. Performance modeling and evaluation of MPI. *Journal of Parallel and Distributed Computing*, 61, 202–223.

Atkinson, D.E., Sawada, M.C., and Gajewski, K., 1998. Emergent spatial structure in vegetation succession. *In: Proceedings of the Annual Meeting of the Canadian Association of Geographer*, Ottawa.

Bandini, S., Mauri, G., and Serra, R., 2001. Cellular automata: from a theoretical parallel computational model to its application to complex systems. *Parallel computing*, 27, 539–553.

Barca, D., *et al.*, 1993. Cellular automata methods for modeling lava flow: simulation of the 1986–1987 Etnean eruption. *In*: C. Kilburn and G. Luongo, eds. *Active Lauas*. London: UCL Press.

Batty, M., 1997. *Growing Cities* (London: Centre for Advanced Spatial Analysis, University College).

Batty, M. and Xie, Y., 1994. From cells to cities. *Environment and Planning B: Planning and Design*, 21, 531–548.

Cannataro, M., *et al.*, 1995. A parallel cellular automata environment on multicomputers for computational science. *Parallel Computing*, 21, 803–823.

Chan, F., Cao, J., and Sun, Y., 2003. High-level abstractions for message-passing parallel programming. *Parallel Computing*, 29, 1589–1621.

Clarke, K.C., Brass, J.A., and Riggan, P.J., 1994. A cellular automata model of wildfire propagation and extinction. *Photogrammetric Engineering & Remote Sensing*, 60, 1355–1367.

Clarke, K.C., Hoppen, S., and Gaydos, L.J., 1997. A self-modifying cellular automaton model of historical urbanization in the San Francisco Bay area. *Environment and Planning B: Planning and Design*, 24, 247–261.

Clematis, A., Mineter, M., and Marciano, R., 2003. High performance computing with geographical data. *Parallel Computing*, 29 (10), 1275–1279.

Costanza, R. and Voinov, A., 2003. Introduction: spatially explicit landscape simulation models. *In*: R. Costanza and A. Voinov, eds. *Landscape simulation modeling: a spatially explicit, dynamic approach*. New York: Springer, 3–20.

Deadman, P.D., Brown, R.D., and Gimblett, H.R., 1993. Modelling rural residential settlement patterns with cellular automata. *Journal of Environmental Management*, 37, 147–160.

Densham, P.J. and Ding, Y., 1996. Spatial strategies for parallel spatial modelling. *International Journal of Geographical Information Science*, 10 (6), 669–698.

Glatzmaier, G.A. and Roberts, P.H., 1995. A 3-dimensional convective dynamo solution with rotating and finitely conducting inner-core and mantle. *Physics of the Earth and Planetary Interiors*, 91 (1–3), 63–75.

Goles, E., 1989. Cellular automata, dynamics and complexity. *In*: P. Manneville, N. Boccara, G.Y. Vichniac, and R. Bidaux, eds. *Cellular automata and modeling of complex physical systems*. Berlin: Springer, 10–20.

Hempel, R. and Walker, D.W., 1999. The emergence of the MPI message passing standard for parallel computing. *Computer Standards & Interfaces*, 21, 51–62.

Hutchinson, D., *et al.*, 1996. Parallel neighbourhood modeling: research summary. *In*: *Proc. SPAA '96*. Italy: Padua, 204.

Kameyama, M. and Yuen, D.A., 2006. 3-D convection studies on the thermal state in the lower mantle with post-perovskite phase transition. *Geophysical Research Letters*, 33, L12S10.

Karafyllidis, I. and Thanailakis, A., 1997. A model for predicting forest fire spreading using cellular automata. *Ecological Modeling*, 99, 87–89.

Komatitsch, D. and Tromp, J., 2002. Spectral-element simulations of global seismic wave propagation – I. Validation. *Geophysical Journal International*, 149 (2), 390–412.

Lett, C., Silber, C., and Barret, N., 1999. Comparison of a cellular automata network and an individual-based model for the simulation of forest dynamics. *Ecological Modeling*, 121, 277–293.

Li, X. and Liu, X.P., 2006. An extended cellular automaton using case-based reasoning for simulating urban. *International Journal of Geographical Information Science*, 20, 1109–1136.

Li, X. and Yeh, A.G.O., 2000. Modelling sustainable urban development by the integration of constrained cellular automata and GIS. *International Journal of Geographical Information Science*, 14 (2), 131–152.

Li, X. and Yeh, A.G.O., 2002. Neural-network-based cellular automata for simulating multiple land use changes using GIS. *International Journal of Geographical Information Science*, 16 (4), 323–343.

Li, X. and Yeh, A.G.O., 2004. Data mining of cellular automata's transition rules. *International Journal of Geographical Information Science*, 18, 723–744.

Mazzariol, M., Gennart, B.A., and Hersch, R.D., 2000. Dynamic load balancing of parallel cellular automata. *Proceedings SPIE Conference, Parallel and Distributed Methods for Image Processing IV*, Volume 4118, San Diego, USA, 21–29.

Morales, D., *et al.*, 2000. Theory and methodology design of parallel algorithms for the single resource allocation problem. *European Journal of Operational Research*, 126, 166–174.

Norman, M.G., *et al.*, 1991. The use of the CAPE environment in the simulation of rock fracturing. *Concurrency: Practice and Experience*, 3 (6), 687–698.

Owczarz, W. and Zlatev, Z., 2002. Parallel matrix computations in air pollution modeling. *Parallel Computing*, 28, 355–368.

Palmer, B. and Nieplocha, J., 2002. Efficient algorithms for ghost cell updates in two classes of MPP architectures. *In*: *Proceedings of the fifteenth international conference on Parallel and Distributed Computing Systems* (PDCS). ACTA Press: Calgary, Canada.

Pontius, G.R. and Malanson, J., 2005. Comparison of the structure and accuracy of two land change models. *International Journal of Geographical Information Science*, 19 (2), 243–265.

Quinn, M.J., 2004. *Parallel programming in C with MPI and OpenMP*. Columbus: McGraw-Hill, 544.

Sirakoulis, G.C., Karafyllidis, I., and Thanailakis, A., 2000. A cellular automaton model for the effects of population movement and vaccination on epidemic propagation. *Ecological Modeling*, 133, 209–223.

Soares-Filho, B.S., Cerqueira, G.C., and Pennachin, C.L., 2002. DINAMICA – a stochastic cellular automata model designed to simulate the landscape dynamics in an Amazonian colonization frontier. *Ecological Modeling*, 154, 217–235.

Turton, I. and Openshaw, S., 1998. High-performance computing and geography: developments, issues, and case studies. *Environment and Planning A*, 30 (10), 1839–1856.

Vokorokos, L., 2005. Parallel computer system utilization in geographic information systems. *IEEE 3rd International Conference on Computational Cybernetics*, Maurícius, April 13–16, Budapest Mad'arsko, 333–338.

White, R. and Engelen, G., 1993. Cellular automata and fractal urban form: a cellular modelling approach to the evolution of urban land-use patterns. *Environment and Planning A*, 25, 1175–1199.

Wu, F., 2002. Calibration of stochastic cellular automata: the application to rural-urban land conversions. *International Journal of Geographical Information Science*, 16, 795–818.

Wu, F. and Webster, C.J., 1998. Simulation of land development through the integration of cellular automata and multicriteria evaluation. *Environment and Planning B: Planning and Design*, 25, 103–126.

Zhang, H., *et al.*, 2007. Toward an automated parallel computing environment for geosciences. *Physics of the Earth and Planetary Interiors*, 163, 2–22.