Multi-column Partitioning for Agent-based CA Model

Pablo Cristian Tissera[1], A. Marcela Printista[12] and Marcelo Errecalde[1]

[1] Laboratorio de Investigación y Desarrollo en Inteligencia Computacional
Universidad Nacional de San Luis
[2] CONICET CCT-San Luis-Argentina
{ptissera, mprinti, merreca}@unsl.edu.ar

Abstract.
Computer simulations using Cellular Automata (CA) have been applied with considerable success in different scientific areas, such as chemistry, biochemistry, economy, physics, etc. In this work we use CA in order to specify and implement a simulation model that allows to investigate behavioural dynamics for pedestrians in an emergency evacuation. Two important aspects must be considered when simulating the movement of people: a) estimation of distances from the cells to an exit and b) handling of collisions between individuals. For the first problem, the Dijkstra algorithm was used. In relation to the collisions, we proposed two approaches to solve the movement of people: centralised on a empty cell and distributed in the neighbouring cells. This latter approach leads to the formulation of Agent-based CA Model for pedestrians motion.
Finally, in order to accelerate the simulation and take advantage of modern computer architectures, the paper also presents a parallel implementation which is an adaptation of the traditional Ghost Cell Pattern technique. This implementation will be essential when the model complexity increases due to the incorporation of new features.
We apply our approaches to several environment configurations achieving important reduction of simulation time.

Keywords: Evacuation Simulation, Parallel Cellular Automata, Pedestrian Motion, Agents.

1 Introduction

Shopping centres, schools and dance halls are some examples of buildings commonly used in different daily activities that involve the meeting of a large number of people within a closed area. The designers of these types of building usually attempt to maximise the productivity of the available space, but also it is necessary to consider a suitable planning for assuring people safety when an unusual behaviour of the crowd occurs. One of the most frequent causes of this kind of behaviour is the emergency evacuation due to the threat of fire. In such situation,

a closed area, with a relatively small number of fixed exits, must be evacuated for a large number of people.

In the last years, the interest in models of emergency evacuation processes and pedestrian dynamics has increased. Models used for evaluating the evacuation processes can broadly be categorised in microscopic and macroscopic approaches. The macroscopic approaches are based on differential equations that take into account the similarities with systems previously studied like dynamics of fluids. They are considered as black boxes because they do not allow to analyse its internal procedures. On the other hand, the microscopic approaches allow to consider how the system state evolves during the model running. In this context, the importance of simulation models for pedestrian dynamics in emergency evacuations is out of question [1,2]. Experimental research in the real world has ethical, financial and logical limitations; furthermore, an evacuation exercise also exhibits some of the risks that we could find in a real evacuation. The simulation allows to specify different scenes with a large number of people and environmental features, making easier the study of the complex behaviours that arise when the people interact.

The cellular automata (CA) are discrete dynamic systems that offer an appealing alternative to deal with this kind of situations due to their capacity to develop complex behaviours from a simple set of rules. Basically, these rules allow to specify the new state of a cell based on the state of the neighbouring cells. In this way, it is possible to model complex dynamic systems from the specification of the local dynamics of its components. A further advantage of these systems is the support usually provided for displaying the results in a graphical way, allowing an easier comprehension of the dynamics of the system under study.

When used as simulation tools, the CA have demonstrated to be very useful at the time of constructing artificial scenes, mainly in those domains where other approaches are not suitable.

Taking into account the previous aspects we decided to use the CA as basis of our simulation model and to investigate the pedestrian dynamics in emergency situations. In particular, we concentrate on those cases that involve the forced evacuation of a large number of people due to the threat of the fire, within a building with a specific number of exits.

In section 2 we include a brief introduction to the main concepts of the CA that were used in the simulation model presented in section 3. In section 4 we present the Agent-based CA model for the pedestrian motion. In section 5 multicolumn partitioning technique is explained. In section 6 we describe our work with different instances of the problem at hand and report the performance analysis of each case. This section also includes a few commentaries on the operation of the EVAC and EVAC* simulation systems that support the proposed models. Finally, the section 7 presents the conclusions.

2 Cellular Automata

The *Cellular Automata (CA)* arose as result of the joint work of John Von Neumann and Stanislaw Ulam in studies related to machines with auto-replication capabilities. These systems received special attention when Jhon H. Conway proposed in 1970 an automaton known as the *game of life* and later popularised by Martin Gardner in an article of *Scientific American*. Since then the CA have grown in popularity within of the scientific community and some researchers have even claimed that the CA represent a possible new paradigm of the physics field named *Digital Mechanics* [3].

The CA are mathematical systems with discrete values in space, time and state. In addition to have *auto-replication* and *universal computation* capabilities, the CA can generate extreme ordered behaviours from a total disorder (*auto-organisation* effects). This last property plays a very important role at the time of explaining certain kind of behaviours observed in physical and biological phenomena [3,4]. The CA have been used to create digital models of physics universes that simulate different phenomena as chemical reactions, diffusion processes, hydrodynamic, mechanic, filtration, chaos theory and others. On the other hand, some elemental organisms obey simple local rules that can be described in a direct way by means of CA.

With respect to the structure of a cellular automaton it can be considered as a dynamic system that represents a grid of locally connected finite automata [5]. Each automaton produces an output from several inputs, modifying its state in this process by means of a transition function. The state of a cell of a cellular automaton in a particular generation only depends of the states of its neighbouring cells and the state the cell had in the previous generation.

At the moment, it does not exist an universal consensus about the terminology and definitions related to the CA. For this reason, we will introduce some elemental concepts that will help the reader to a better understanding of this work. Intuitively, we will consider a cellular automaton as a system composed by a array of cells A. Each cell c_i in A represents a *finite automaton* with a set of states Q, an input alphabet Σ and a transition function $\delta : Q \times \Sigma \rightarrow Q$. A particularity of this automaton is that the input alphabet Σ is given by all the possible combination of the *cell states* of the adjacent (neighbouring) cells. If we denote N^{-c_i} to the set of cells that we will consider as neighbour of an arbitrary cell c_i, and $|N^{-c_i}| = n$ is the number of adjacent cells, then the input alphabet will be $\Sigma \equiv Q^n$. Usually, a cell c_i and its adjacent cells are all together considered and represented as a unique set $N = \{c_i\} \cup N^{-c_i}$ which is referenced as the *neighbourhood*. It is important to emphasise that each cell has a copy of the *structure* of the finite automaton, but the initial state of each cell could be different.

We can now to define a *Cellular Automaton* as a 4-tuple $M = \langle A, Q, \delta, N \rangle$ where:

- A is a D-dimensional array, and each component (cell of the array) has associated a *finite automaton*.

– Q is a finite set of states (of the automaton) of a cell
– N is the specification of which cells are included in a neighbourhood, $N \equiv \{c_i\} \cup N^{-c_i}$ such that N^{-c_i} are the adjacent cells to c_i.
– Let $\Sigma \equiv Q^n$ where $n = |N^{-c_i}|$ is the number of adjacent cells to c_i. The *transition function* of states, $\delta : Q \times \Sigma \to Q$, is a mapping such that if $q_i \in Q$ is the state of the cell c_i in the time t and $q_{i+1}, q_{i+2}, \ldots, q_{i+n} \in \Sigma$ are the states of the adjacent cells to c_i, and

$$\delta(q_i, q_{i+1}, q_{i+2}, \ldots, q_{i+n}) = q_i'$$

then q_i' denotes the state of c_i in the time $t + 1$.

It is usual to refer to the cell c_i under consideration as the "central cell" or "cell 0" and to enumerate as "neighbour 1", ..., "neighbour n" the adjacent cells to c_i. The δ function is usually represented in tabular form with $|Q|^{n+1}$ rules:

$$\delta(q_i, q_{i+1}, q_{i+2}, \ldots, q_{i+n}) \to q_i'$$

where $q_i, q_{i+1}, q_{i+2}, \ldots, q_{i+n}$ are the states of the central cell and its neighbours at the time t and q_i' is the state of the central cell at the time $t + 1$. In some cases, it is possible to specify *probabilistic transition rules*, where an arbitrary probability p can be associated to a transition rule. The semantic of this kind of rules establishes that, always that a cell matches the configuration of the specified neighbourhood in a probabilistic rule, the cell will have at time $t + 1$ the new state specified in such rule with probability p.

3 CA Model for Evacuation Simulation

To solve problems concerning with the real world is usually handled with simulation model, which are created to reflect the main features of the real system under consideration. Below, we describe the main features of our proposed model which is based on a CA approach.

Cellular Space: it is a finite bi-dimensional array (grid) with closed boundaries. Each cell of the cellular space represents $40 \times 40\ cm^2$. This is the space usually occupied by a person in a crowd with maximal density [6,7,8]. The dimensions of the cellular space are specified in meters. So, one grid of $10 \times 10\ m^2$ will contain 25 cells by side (see Fig. 1).

States: a cell can be in one of the states of the set $Q = \{W, E, P, O, S, SF, PS\}$ where:

W : External wall cell.
E : Empty cell.
P : Cell with a person.
O : Internal obstacle cell.
SF : Cell with smoke and fire.
PS : Cell with a person and smoke.
S : Cell with smoke.

Neighbourhood: the neighbourhood considered in the model is *Moore's Neighbourhood*, that includes the eight cells surrounding the central cell. With this

choice we aim to provide to each individual in the system with all possible movement directions.
Initial Configuration: before the simulation starts diverse information related to the outer walls, inner obstacles, individuals, combustible locations, cell with fire and arrangement of the exits must be specified.
Virtual clock: taking into account recent studies [1,6], an updating time of 0.3 seconds by time step was specified for our model. This value is the estimated time required by a pedestrian for walking 0.4 m (size of a cell side).

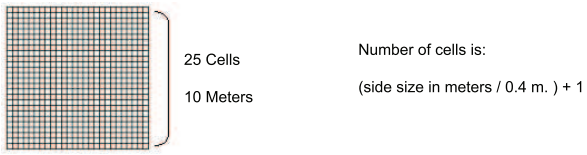

Fig. 1. Cellular space of $10 \times 10\ m^2$

Model Evolution Rules [9]:

1. Rules about the *building:* a cell in state W or O (outer wall or obstacle) will not change its state throughout the simulation.
2. Rules about *smoke propagation:* a cell with smoke[3] at time t, also will have smoke in time $t+1$. If at time t the central cell does not have smoke, but some of its adjacent cells have smoke, the central cell also will have smoke at time $t+1$ with a probability proportional to the number of adjacent cells with smoke. For example, in Fig. 2 the central cell (marked with a X) has four adjacent cells with smoke on a total of eight adjacent cells. In this case, the central cell will have smoke in the next time step with probability $\frac{1}{2}$.
3. Rules about *fire propagation:* these rules are analogous to the rules for smoke propagation. However, they incorporate an additional constraint: a non-zero combustion level of the cell is required.
4. Rules about the *people motion:* a cell without people will have a person in the following cycle if *a*) at least one adjacent cell contains an individual and *b*) the distance from the cell under consideration to an exit is less than the distance from the cell occupied by the individual to the exit. In other cases, the cell does not change its state. Other aspects related to the people motion are described in section 4.

4 Agent-based CA Model for Pedestrians Motion

Intelligent agents have been used successfully in a wide range of applications. In artificial intelligence, an intelligent agent (IA) is an autonomous entity which

[3] In one of the following states: S, SF or PS.

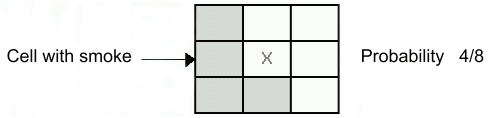

Fig. 2. Probability of smoke propagation.

observes and acts upon an environment (i.e. it is an agent) and directs its activity towards achieving goals [8].

A simple agent program can be defined mathematically as an agent function which maps every possible environment state to a possible action the agent can perform or to a coefficient, feedback element, function or constant that affects eventual actions: $Ag : S^* \rightarrow A$.

In the proposed model we set $S = Q$, $A = \{move, no\text{-}move\}$ and the function Ag represents the agent capability to decide about its desire to move to the cell depending on its environment. For this work we have defined a simple reflex agent, but the model (and the software that implements it) supports many more complex behaviour.

4.1 Calculating Distances and Handling of Collisions

Two important aspects must be considered when simulating the movement of people: *a*) *estimation of distances* from the cells to an exit and *b*) *handling of collisions* between individuals. For the first problem, the Dijkstra's algorithm was used. This algorithm solves the single-source shortest-paths problem on a weighted graph [10]. The cellular space is considered as a graph, where each cell represents a node and all the edges connecting adjacent cells have weight *1*. Cells with state W (wall) or O (obstacle) are not considered to build the graph.[4]. If the building have more than one exit, the distance computation takes into account the exit nearest to the cell (see Fig. 3).

The handling of collisions between the individuals, is one of the most common problems faced in pedestrian dynamics. The conflict arises when two or more people simultaneously attempt to occupy the same physical location belonging to an exit way. To solve this aspect, we changed the approach commonly used in other works. Instead of being the pedestrian who decides which cell to move, the current cell[5] is in charge of choosing between the people in the adjacent cells, which pedestrian will move to the cell.

The obvious question is: Which are the criteria used by the cell in order to select the pedestrian to move in?. The process that we used for solving this

[4] An individual can not cross a wall and therefore he should surround the wall to reach the exit

[5] If the cell is in a state suitable to receive an individual.

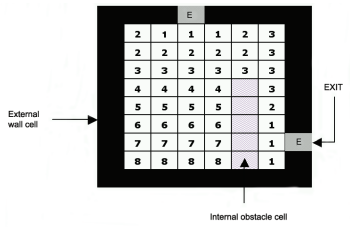

Fig. 3. Estimation of distances to the exits

point is the following: *a*) any person in the neighbourhood whose distance to an exit is shorter than the distance of the cell attempting to receive an individual, will not be considered as candidate (see Fig. 4). *b*) If more than one individual still remains as candidate to occupy the cell, the one with the minor number of damage points (parameter specified in the model) will be selected. *c*) Finally, if the conflict persists, the pedestrian will be selected at random. The decision is concentrated in the empty cell.

Another policy is to distribute the decision between neighbouring agents. This strategy, proposed in the section 4.2, consists of two steps: a step of Ranking of individuals and a further Round of Queries to the ranked list of neighbour. On the request, the agent based on its environment, must decide about its desire to move or not to the cell in question.

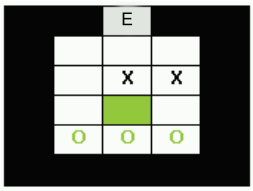

Fig. 4. Shadow cell will choose one person from some cell O, because the cells with an X, are located in a better location (nearer to exit.)

4.2 Agents Decision

As mentioned above, after the Ranking, each agent of the ranked list must decide about its beliefs to move or not to the cell that makes the request.

For this, the agent builds a *beliefs matrix* containing the distances from its neighbour to exits. This information is relevant to the decision that it must take:

If the distance from the requirer cell to the exit is greater than the current distance, then the agent does not move.
If the distance from the requirer cell to the exit is less than the current distance, then the agent replies its desire of moving.
If both distances are the same, the agent checks from its neighborhood, cells that offer better placement:
– If the number of empty cells around the agent with a better position is 0 or 1, the agent will respond positively to the move with a 50% probability.
– If the number of empty cells around the agent with a better position is 2, the agent will respond positively to the move with a 25% probability
– If the number of empty cells around the agent with a better position is greater than 3, the agent says no wish to move. With increasing number of empty positions around a cell, the desire to move decreases.

The above percentage values were obtained empirically, using those that gave better results for evacuation time and average distance traveled by each agent to the exit.

5 Multi-column Partitioning

The CA are discrete systems that evolve over time. As a general rule, the evolution of CA is performed by repeated update of the complete set of cells, where the new state of each cell depends on the existing state of its neighborhood. CA simulation techniques used in this work, are too slow if the space to simulate is large or complex.

The aim of this work is to search for techniques to accelerate simulations exploiting the parallelism available in current multicomputers.

One of the most common methods used for resolving this problem is the Ghost Cell Pattern [11]. In this technique, the grid is geometrically divided into chunks that are processed by different processors. One challenge with this approach is that the update of points at the periphery of a chunk requires values from neighboring chunks. If a cell and its neighbors are in the same node, the update is easy. On the other hand, when nodes want to update the border cells, they must request the values of the neighboring cells on other nodes. The solution to this problem is to allocate space for a series of ghost cells around the edges of each chunk. For every iteration, each pair of neighbors exchanges a copy of their border and places the received borders in the ghost cell region (see Fig. 5). The ghost cells form a halo around each chunk that contains replicates of the borders of all immediate neighbors. These ghost cells are not updated locally, but provide stencil values when the borders of this chunk are updated.

In our proposal, the cellular space of the automaton is represented by an bi-dimensional array, which contains $X \times Y$ cells. Inside of a cluster based on distributed memory system, the parallel execution using P processors (denoted $p_0, p_1, ...p_{P-1}$) is performed by applying the transition function simultaneously to P chunks in a $SPMD$ way. Due to the particular characteristics of the model,

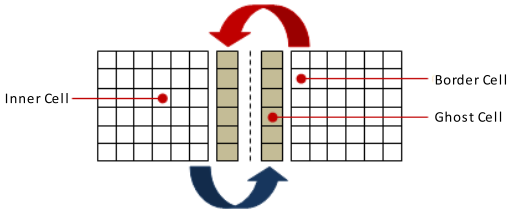

Fig. 5. Border Exchange

the proposed parallelization differs a bit from the traditional approach [12,13]. The Fig. 6 shows the decomposition applied in our proposal where the cellular space is divided in one-dimensional chunks (multi-column).

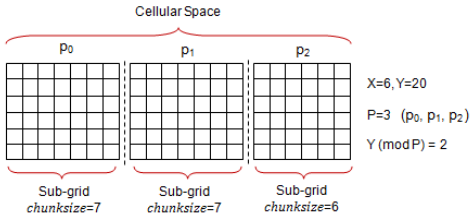

Fig. 6. Multi-column Decomposition: if P and Y are multiples then $chunksize = Y/P$; if P and Y are not multiples and $rank < Y \pmod P$ then $chunksize = Y/P + 1$ else $chunksize = Y/P$

In each time step, the algorithm updates each cell in the lattice. If a cell contains an individual, three things can happen in the next state: (1) the individual may leave the border cell of a chunk and move into the other chunk, (2) it can change of chunk, and (3) it can move from inside a chunk to its border. The Fig. 7 illustrates these movements. Situations 1 and 2 can lead to inconsistencies in the next state.

Suppose in the Fig. 8-left that the cell marked with X is being updated by the process 1. That cell offers a better position than the pedestrian currently possesses (filled circle), so after applying the transition rules it decides to move there. The resulting state would be that the individual has taken the new position, leaving the ghost cell of process 2 outdated, since in the cell still contained the pedestrian who is no longer in that position (Fig. 8-right).

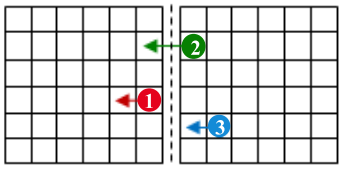

Fig. 7. Possible Movements

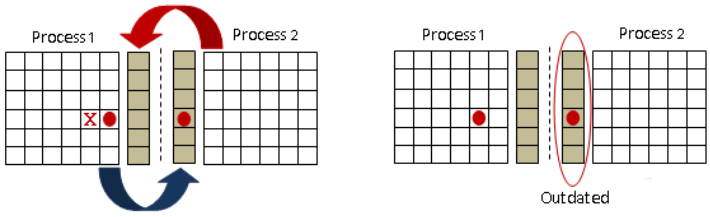

Fig. 8. Status Inconsistency

At first this does not imply any kind of error, but if the individual was placed in a position equidistant from two possible exits, the agent could take a different decision in both processes. The effect of this situation can be seen in Fig.9, with the individual replicated moving toward two possible exits.

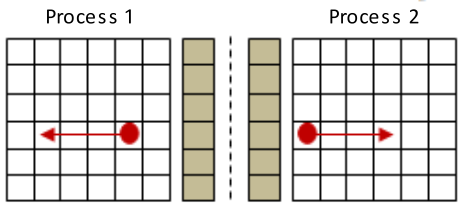

Fig. 9. Replicated Individual

To keep the two processes with the updated status on the decision of an agent by making use of ghost cell pattern technique requires multiple message exchange during the same time step simulation. If we want to model large environments (airports, convention centers, stadiums, etc..) and simulate the evacuation of thousands of people, the communication overhead would be significant, implying a degradation of performance of the model.

The Algorithm 5.1 shows the proposed methodology that alleviates the impact of the interaction of processes against the decision to an agent. The algorithm is illustrated by means of the Figs. 10, 11, 12 and 13.

Algorithm 5.1: EVAC*()

comment: l : leftmost column r : rightmost column

while People to evacuate

do
- **if** $rank < P - 1$
  - **then**
    - Send local $chunk_r$ to process $rank + 1$
    - Receive local $chunk_r$ from process $rank + 1$
- **if** $rank > 0$
  - **then**
    - Receive $chunk_{ghost}$ from process $rank - 1$
    - Evolve $chunk_{ghost}$ and local $chunk_l$ (contiguous columns)
    - Send evolved $chunk_{ghost}$ to process $rank - 1$
- Evolve remaining cells of $chunk$
- Actualize number of people to be evacuated

End.

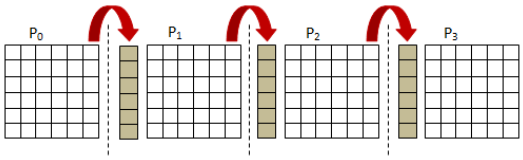

Fig. 10. If ($rank \leq P - 2$) Send rightmost column of $chunk$ to process $rank + 1$

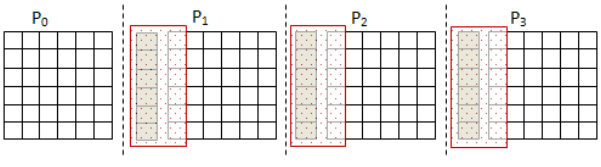

Fig. 11. Evolve ghost and leftmost columns

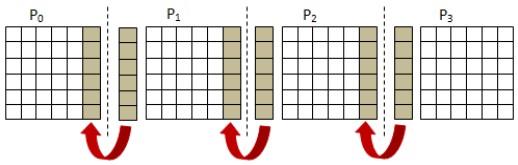

Fig. 12. If $(rank \geq 1)$ Return evolved ghost column to process $rank - 1$

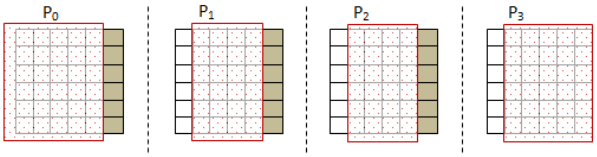

Fig. 13. Evolve remaining cells

6 Simulation

In this section, we present the simulation result of the explained research.

The first group of experiment was carried out with EVAC Simulator [9][14], an integrated simulation system based on cellular automata. EVAC is a system developed in Java that allows the design and simulation of spatial environments in an explicit way. EVAC simulator offers a friendly graphical interface which can be easily used by non expert users.

EVAC can be used in two different modes: *edition* mode and *execution* mode. In edition mode, it is possible to create a new environment or to modify an already existent environment. The general parameters can be set and the main features of the building to be used during the simulation can be graphically specified. In execution mode in turn, an animation of the evacuation process is displayed to the user using the model previously defined in the edition mode. This mode also generates a report with statistical information gathered during the simulation of the evacuation process. These facilities are introduced in EVAC as useful tools that allow the user a better comprehension, design and analysis of the study case under consideration.

In each simulation step, EVAC performs an exhaustive updating of the cellular space, using at this end, the evolution rules described in sections 3 and 4. The cells of the CA are updated at random with the purpose to avoid any sort of bias generated by the order in which the cells are updated. The simulation finishes when all the people have evacuated the building.

The second group of experiment was carried out with EVAC*, an simulation system based on parallel cellular automata. EVAC* is a system developed in C and MPI for passing message and uses the graphical interface (off line) of EVAC.

6.1 Results and analysis

The experiments consider four environment configurations of the building to be evacuated (A,B,C,D), addressed those situations where several exits exists, varying the different occupation densities of the environment (number of individuals) and the size of the exit:

1. A, $10 \times 10\ m^2$, two exits of 1.2 m. each, 150 individuals distributed evenly.
2. B, $20 \times 20\ m^2$, three exits of 1.2 m. each, 650 individuals distributed evenly.
3. C, $40 \times 40\ m^2$, five exits of 1.2 m. each, 1350 individuals distributed evenly.
4. D, $80 \times 50\ m^2$, five exits of 2.4 m. each, and 1850 individuals distributed evenly.

With the purpose of obtaining acceptable statistical data, the results shown below correspond to the average of 100 independent replications of each experiment. The corresponding confidence intervals were obtained, for the total evacuation time (TET) (seconds) , mean evacuation time (MET) (seconds) and mean traveled distance (MD) (meters) per individual to the exit.

Looking Tables 1 and 2 we can observe a comparison of both strategies. While the general behaviour of both models has similar features, it is important to note that in most cases, the Agent-based CA Model shows a decrease in both the total evacuation time and the mean evacuation time per person.

Case	TET	MET	MD
A	18.243-20.296	7.208-7.222	9.608-9.632
B	31.474-32.465	14.416-14.421	19.220-19.229
C	85.043-89.976	31.030-31.037	41.372-41.384
D	104.999-108.680	53.765-53.768	71.686-71.692

Table 1. EVAC: CA-based Simulations

Case	TET	MET	MD
A	15.263 - 15.856	5.184 - 5.187	6.911 - 6.917
B	31.003 - 32.016	10.870 - 10.871	14.493 - 14.496
C	80.465 - 83.614	23.427 - 23.429	31.236 - 31.239
D	96.191 - 100.928	41.348 - 41.350	55.131 - 55.134

Table 2. EVAC*: Agent-based CA Simulations

Through the use of visualization module, it was detected in the CA-based Model the individual, on their path to the exit, follows a more sinuous route than the one chosen by the agents. Also, we can observe that in certain time step, some agent decides not to move from its place to await a better position, which causes some other agent in the neighborhood occupies the empty cell. We believe that a strategy based on the perception of each individual (agent) is closer to reality, which leads to a more rational evacuation. To date, we are investigating the addition of higher level behaviours, such as grouping, and collision negotiation, into our overall pedestrian simulation system. These behaviours are being designed to sit on top of presented model.

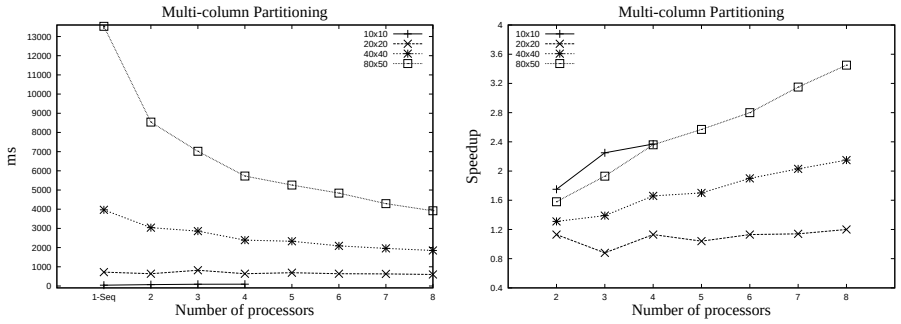

Fig. 14. Parallel Time and Speedup of Evacuation Simulation

For the parallel experiments, we used a cluster equipped with 15 nodes Pentium IV of 3.2 GHz per node with hyperthreading technology. Each node has 1GB RAM and disk of 20 Gb SATA. The nodes are interconnected by a Switch Dlink DGS- 1224T. With both growing environment size and the occupation densities the simulation time increases. As the result of using a partitioning strategy where the cellular space is divided in one-dimensional chunks, we observed that as the degree of parallelism grows, the simulation performance improves.

In addition to overall execution time, we measured the computation times for each environment configuration. Following the applied partitioning strategy, the time spent on communication does not vary significantly between different configurations. So that, all the expectation of performance optimization of simulation is based on reducing the computation time at each time step. But for the experimental configurations shown in this paper, the expected acceleration of the simulation has a threshold that limits its growth. This bound is reached when the percentage of time involved in computing and communication begin to match. In our case, for example, for the 80×50 m^2 setting the limit is reached with 8 processors.

Both cases C and D have five exits and they are considered complex environments. This implies that at the beginning of the evacuation, agents are involved in various discussions aimed at deciding the most appropriate exit to

Number of Processors	2	3	4	5	6	7	8
% Communication	8.06	27.00	31.65	37.72	39.37	39.65	45.99
% Computation	91.93	72.99	68.34	62.28	60.63	60.35	55.18

Table 3. Computation vs Communication

leave the building. Despite the computational work that the discussions involve, the partitioning strategy achieves reasonable accelerations.

7 Conclusions

In this work, we used Cellular Automata for developing and implementing a simulation model of emergency evacuations due to the fire threat. In order to implement the model, we presented the EVAC system which allowed to analyse evacuation processes with different building configurations and experimental conditions.

In spite of the assumptions introduced for obtaining a more simple model, the CA resulted to be very suitable tools for modelling this class of problems achieving in many cases, results very similar to those expected to occur in a real evacuation situation.

CA allowed to generate "local" and "uniform" behaviours that resemble the dynamics observed in real processes of fire and smoke propagation. However, these local features were not suitable for representing certain aspects of people behaviour that require a more global and differentiated perspective. We developing a hybrid model where the dynamics of fire and smoke propagation are modelled by means of CA and for simulating people behaviour we are using goal oriented *intelligent agent*.

While agent-based models afford higher complexity and finer granularity they also require more computational resource. This means for large populations, modeling individuals with complex rules becomes infeasible, typically individuals will instead be modeled as simple particles. We used partitioning technique as a solution to this problem, allowing large scale simulations. The simulation is then divided up with the set of environment and its agents being distributed equally amongst the computer nodes. However, distribution introduced its own challenges. The proposed multi-column partitioning method reduces the communication and synchronization time, while ensuring each agent did not perceive contradictions in the environment.

We have analyzed the behaviour for some typical scenarios. First we worked to adjust the behaviour of agents and improvements were found regarding the choice of the path forward by individuals trying to reach the exit. Then we focused on evaluating the time reduction of the proposed partitioning for the simulation.

Finally, we believe that the EVAC and EVAC* systems are a good start point for analysing and designing preventive safety policies and therefore, we wish to investigate further to optimize both the model and its parallel implementation.

8 Acknowledgements

We thank the National University of San Luis and the ANPCYT for their unstinting support.

References

1. H. Klupfel, M. Schreckenberg, and T. Meyer-Konig. Models for crowd movement and egress simulation. In Serge P. Hoogendoorn, Stefan Luding, Piet H. L. Bovy, Michael Schreckenberg, and Dietrich E. Wolf, editors, *Traffic and Granular Flow 03*, pages 357–372. Springer Berlin Heidelberg, 2005.
2. Nuria Pelechano and Ali Malkawi. Evacuation simulation models: Challenges in modeling high rise building evacuation with cellular automata approaches. *Automation in Construction*, 17(4):377 – 385, 2008.
3. Wolfram Stephen. *Cellular Automata and Complexity*. Addison Wesley, USA, 1994.
4. Stuart A. Kauffman. Emergent properties in random complex automata. *Physica D: Nonlinear Phenomena*, 10(1-2):145 – 156, 1984.
5. Tomas de Camino Beck. Un lenguaje para la especificación de autómatas celulares con aplicaciones biológicas. Master's thesis, Instituto Tecnológico de Costa Rica, 2000.
6. C Burstedde, K Klauck, A Schadschneider, and J Zittartz. Simulation of pedestrian dynamics using a 2-dimensional cellular automaton. *Physica A: Statistical Mechanics and its Applications*, 295(3-4):507525, 2001.
7. Michael Schreckenberg, Tim Meyer-knig, and Hubert Klpfel. Simulating mustering and evacuation processes onboard passenger vessels: Model and applications. In *In The 2nd International Symposium on Human Factors On Board (ISHFOB*, 2001.
8. Timmermans Harry. *Pedestrian Behavior: Data Collection and Applications*. Emerald Group Publishing Limited, 2009.
9. Tissera Cristian. Simulador de evacuaciones basado en autómatas celulares. Informe de tesis de licenciatura UNSL. Julio 2006.
10. Brassard G. and Bratley P. *Fundamentos de Algoritmia*. Prentice Hall, 2000.
11. Fredrik Berg Kjolstad and Marc Snir. Ghost cell pattern. In *Proceedings of the 2010 Workshop on Parallel Programming Patterns*, ParaPLoP '10, pages 4:1–4:9, New York, NY, USA, 2010. ACM.
12. K. Nishinari, A. Kirchner, A. Namazi, and A. Schadschneider. Extended floor field ca model for evacuation dynamics. In *IEICE Transactions on Information and Systems, E87-D*, pages 726–732, 2004.
13. Victor J. Blue and Jeffrey L. Adler. Cellular automata microsimulation for modeling bi-directional pedestrian walkways. *Transportation Research Part B: Methodological*, 35(3):293–312, March 2001.
14. Marcela Printista, Marcelo Errecalde, and Cristian Tissera. Evacuation simulations using cellular automata. *JCS&T*, 7(1):14–20, 2007.