Crowd Dynamics Modeling and Collision Avoidance with OpenMP

Albert Gutierrez-Milla
Francisco Borges
Remo Suppi
Emilio Luque

Department of Computer Architecture and
Operating Systems
Universitat Autónoma de Barcelona
Bellaterra, 08193
Barcelona, Spain

ABSTRACT

This paper deals with the problem of simulating crowd evacuations in multicore architectures. Manage evacuations is an important issue that involves lives, and in the case of crowds, thousands of lives. We present our model, able to hold thousands of agents walking through the scenario avoiding dynamic and static obstacles. We test how the model behaves with agents falling down, becoming an obstacle. Simulators are widely used in the area of crowd evacuations. In the present work we introduce a crowd model and a HPC simulation tool. Thread level parallelism has a model similar to human behavior where every person is taking decoupled decision. People have the same behavior and they take different decisions depending on the other agents. We used OpenMP to program a multicore architecture. The experimentation shows the scalable performance of the model and how taking decoupled decisions has an impact in the simulator performance.

1 INTRODUCTION

Evacuation planning is a key factor when big events are prepared. Crowd events are getting more and more usual and prevention is a key factor in order to avoid disasters. Bad prevention, planning or management has driven events to disasters such as Madrid Arena disaster (5 deaths), the Hillsborough Stadium in Sheffield (96 deaths) or the Love Parade in Duisburg (21 deaths). It shows the importance of god evacuation planning and the need to have previous knowledge to host big events. Human behavior is not always predictable, and evacuations are complex cases were there are a lot of cases. Crowd evacuations is a special case inside evacuations where a big population is involved (hundreds or thousands). In the particular case of crowd evacuations the complexity increases. Bodies get harmed and they become dynamic obstacles, panic stampedes, domino effect (Helbing and Mukerji 2012), Crowds have special issues as "crowd turbulences" (Moussad, Helbing, and Theraulaz 2011), stop and go waves, etc.

Drills are approaches were the repeatability and the characteristics of the agents cannot be controlled. Fire drills lack of repeatability and control of the human emotions that is why simulators are an increasingly getting the interest of the people who take decisions. Referring to the complexity of evacuations Helbing said: "Despite the sometimes more or less chaotic appearance of individual pedestrian behavior, some regularities can be found" (Helbing and Molnar 1998) which extracts patterns of motion of the agents. There are several approaches that solve the evacuations problem such as: social force, cellular automata, fluid dynamic models, lattice gas models, etc (Zheng, Zhong, and Liu 2009). We use an agent based model (ABM) approach, which has been widely used in computational sociology (Macy and Willer 2002),

biological systems (Grimm, Revilla, Berger, Jeltsch, Mooij, Railsback, and DeAngelis 2005) or complex systems. ABM and the self organization emerging from the population behavior and can implement the models described above adapting the theoretical model to the implementation. ABM has an emergent behavior approach were the global output of the systems comes from the rules dictated to every agent. Agents interaction needs to consider collision avoidance where communication among the agents determine the own behavior. Extreme densities is one of the most important topics, because of that we are interested in models that allow densities, instead of grid-based models. And we will use the Velocity Obstacles model which implements the free navigation among agents without collisions and the multiagent cases.

Simulators are used as DSS (Decision Support System) in areas such as clinics (Musen, Middleton, and Greenes 2014), forests (Varma, Ferguson, and Wild 2000) allow to retrieve information to the decision making committee or person according to a model. Different simulators have different properties and contributions. Software such as MicroSaint or Oasys MassMotion focus on the visualization and integration of buildings. Some simulators focus on fire and complex situations useful for reconstruction and fire fighters as FDS + EVAC (Korhonen, Hostikka, Helivaara, and Ehtamo 2005).Crowd evacuation simulators have performance needs over normal evacuations. In the present work we present a crowd evacuations model able to support thousands of agents. In the model decision are decoupled and every agent takes decisions according to its environment, which is more similar to the reality than a centralized decision making system. The goal of this work is to provide a scalable software breaking data dependencies and make it suitable for multicore architectures like GPUs. Multicores have gained popularity in desktop and even mobile architectures. The paper presents the results that show the scalability of our HPC tool and the linear speedup.

The rest of this paper is organized as follow. Section 2 describes the related work on Crowd models and collisions avoidance. In Section 3 we present our model and the formalization. The simulation process is described in section 4 as well as the algorithm run by every kernel. We present experimental results and validation in Section 5 and the work is concluded in Section 6.

2 RELATED WORK

Crowd modeling has a big interest in the area of computer graphics (Pelechano, Allbeck, and Badler 2007). Other areas involve building security construction, pedestrian as transportation or evacuation in events. Modeling the interactions can be solved by different areas such as robotics or video games.

2.1 Crowd models

Agent Based Modeling (ABM) is a modeling technique to simulate the actions and interactions of autonomous agents. The interactions and governing rules of the agents are defined in a micro level (known at the beginning) and their interaction emerge a macro-behavior (unknown at the beginning). Agent interact with each other sharing information .

Crowd behavior has been approached on several ways but social force has been shown as the most significant contribution to the field. Previous work based on the social force has shown significant results and has been validated against real study cases. Some implementation of the social force use ABM to implement. Other are based on Cellular Automaton.

2.2 Collision avoidance

Free movemnt of agents through the space drive to a first problem: agent's overlapping throught the space. in the Fig. 1 an exemple of this concept can be seen. Collision avoidance algorithms solve this problem using different approaches divided, mainly, in decoupled and centralized. An important problem to solve when the agent is to avoid collision and move freely around the space without getting stuck. Several research present approaches with graphs and potential field. Grid partitioning methods simplify the collision avoidance problem. The density difference is based on empty or filled grids. Positioning based on discrete coordinates freely moving around space. Robotics has made a great contribution to this field. VO

(Velocity obstacles) was introduced by Fiorini and Schiller (Fiorini and Shillero 1998). A review of this contribution was presented by Van den Berg: RVO was designed to solve the oscillation problem inherent to the principles of VO (Guy, Lin, and Manocha 2011).

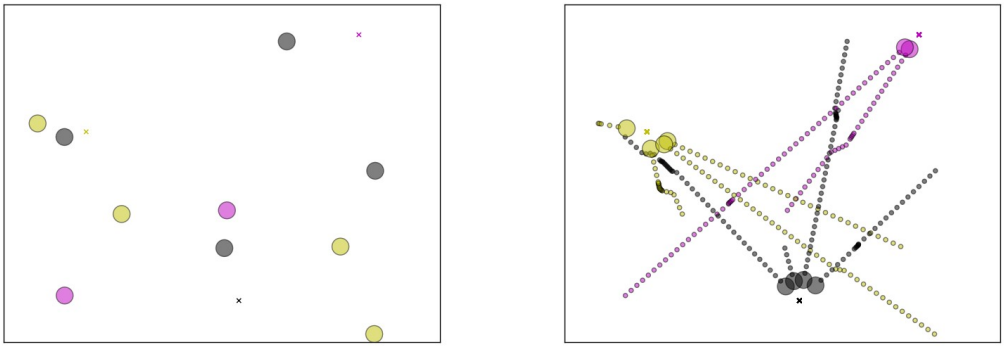

Figure 1: (a) Initial positions. The cross indicates the goal by color. (b) Movement of the agents.

Potential field is a centralized collision avoidance algorithm. It uses a grid filled with values that determines which is the next step for every agent to move. It has been widely used and simplifies the collision avoidance problem increasing the memory complexity. It merges collision avoidance and path finding following a shortest path approach.The cons are that it increases the space complexity and decreases the realism, and the pros is that is more computational efficient because the access to the neighbors regions and check if they are filed is a direct access. It also merges collision avoidance and path planning.

VO has been used by several works in multiagent crowds (Guy, Chhugani, Kim, Satish, Lin, Manocha, and Dubey 2009). RVO is an improvement of VO where it calculates the average of both velocity vectors. RVO assumes that all the agents make a similar collision avoidance reasoning, and this will drive to a guaranteed safe and free-oscillation. Videogames industry is highly interested in performance and artificial intelligence for video game bots. NVIDIA is one of the nowadays great partners of this research. Abi Bleiweiss (Bleiweiss 2009) published a CUDA implementation of the RVO algorithm. The collision avoidance algorithm is based on RVO (Berg, Lin, and Manocha 2008).

In the present work we used an ABM approach to implement our model. We are based on the RVO technique for collision avoidance among the agents. In our particular case we have crowded scenarios. The architecture used for this purpose is a multicore as it has been used in other research in video games.

3 CROWD MODEL

Crowds model has been an study for more than 40 years proposing theoretical models of the human behavior. Computational simulation has gained an increasing interest in recent years. In this section we present our local dynamics and collision avoidance algorithms and our approach. In this paper we do not consider the path finding problem, which is a set of intermediate goals.

3.1 Crowd dynamics

The crowd movement has been modeled as the interactions between agents. These interactions have been defined as the sum of forces, velocity vectors, union of occupied areas, etc. An important field is artificial intelligence, 3D visualization and video games. Video games industry plays an important role in crowd dynamics research where crowd behavior has been gaining importance in recent years

The characteristics of scape panic described by Helbing include: *People show a tendency towards mass behavior, that is, to do what other people do* and *Alternative exits are often overlooked or not efficiently used*

in escape situations. Both points are related to a tend of mass behavior.Pressures come from agglomeration and interactions come from pushing. Group behavior emerges when a group of people share the same objective. In our model we describe a goal for every agent. Agent are influenced by other agents. Some agents have a exit set, and other do not. Agents are influenced by other agents with more knowledge.

All the times are independents and the past time is not analyzed. So the only steps calculated is from $t \rightarrow t + 1$. The agent is modeled by a circle that represents the area. Current position, the goal, and speed retrieves the speed vector v that determines. We can represent densities. We model the knowledge (k) that the agent has about the exit. The knowledge is a value in the interval [0,1]. Agents compare their knowledge to other agents knowledge. When their knowledge is behind a threshold their speed decreases. When agents follow an agent with more knowledge their knowledge increases.

As mentioned above in the social force model, a set of social forces determine the influence in every agent in the social force model. Current position, the goal and the speed gives the vector τ. This vector is transposed from -60 to 60 as showed in Fig. 2. As well as in RVO all agents always chose by default the same side. v is recalculated from v_{max}. Injured agents have a $v_{max} = 0$. δt is the simulation time step. The con is that positions are not accessed directly as in a grid representation. Every agent iterates over all the other positions. Other models[Avi] precomputed the global path in the GPU. In this model we do not include path finding algorithm. The atributes defined for every agent are the following:

- Speed: s
- Diameter: d
- Angle: φ
- Position: P_i
- Goal: G_i

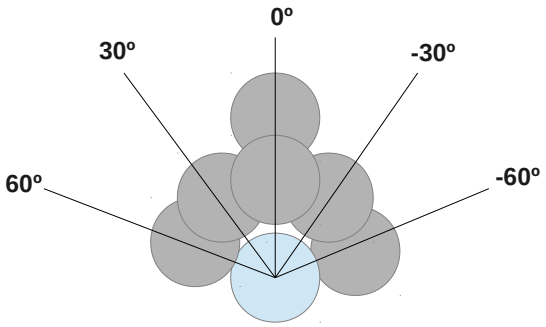

Figure 2: Model representation with the agent in blue and the ”future position” areas in gray.

3.2 Collision avoidance

Free movement through the space involves the first problem: avoid agent from intersecting their areas. As seen previously, Velocity Obstacles is a technique used in robotics, video games and agent based simulations. Collision avoidance is one of the most costly computationally because is a combinatorial problem based on the comparison between agent’s positions. The scene contains static and dynamic agents and static obstacles as walls.

Because the space is a geometry we will use Minkowski notation for the mathematical model. Agents try to walk at their desired speed. Our collision avoidance algorithm is based on concepts introduced by Velocity Obstacles (VO) used by Fiorini and Shiller (Fiorini and Shillero 1998) and RVO. VO has been

used in many application such as robotics, video games or crowds. The equation that describes VO is as follows:

$$VO_B^A(v_B) = \{\ v_A \mid \lambda(p_A, v_A - v_B) \cap B \oplus -A\ 6{=}\emptyset\}$$

RVO assumes that opposite agents will have the same reasoning. RVO is based on a set of simple rules. RVO is a collision avoidance model that solves the oscillation problem. The model needs to be adapted to a SIMD computational model to fit in the GPU. Each agent individually computes the avoidance. This makes more complex the computation because two agents can calculate that they are moving to the same position in time t and at time $t + 1$ there will be a inconsistency state. Because of this we calculate the normal vector of every agent and consider the neighbors. We do this without gathering the information. The only moment when an agent knows the new position is in the next time stop.

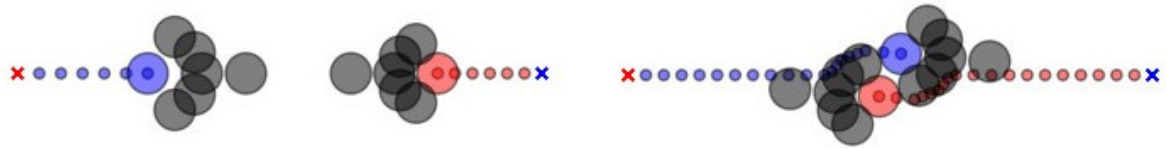

Figure 3: Collision avoidance

Here we consider this approach for cases of closed spaces and we don't solve the path finding problem. RVO has a highly parallel approach that makes it suitable for SIMD architectures such as GPGPUs. All agent take decisions independently. In our particular case these rules will be applied mostly in the fluid phase, but the clogging phase will be most of the time. We created a simple method with a set of points that discretize the surrounding space of the agent. In the case of multi agent navigation were the union of all RVO fill all the space the agent's speed is penalized according the v_{pref} and the collision time. In Fig. 3 we show an example of collision avoidance. When the collision is imminent the speed is reduced to zero in order to avoid oscillations. Evacuation can be divided in two phases: fluid phase and jammed phase. This method allows agents to move collision free.

4 SIMULATION

We implemented the model described for a high performance architecture. We model one agent as a structure with attributes that will be what defines the movement and interaction. Agents are modeled as circle and the scenario is represented by a bi-dimensional map. Agents are spread randomly among the space avoiding initial inconsistencies as collisions.

4.1 Agent simulation step

Simulation time discretizes the continuity of time to adapt it to machine. The time step is determined by δt that directly affects to the performance of the simulation time and the number of simulation steps if its too low and affects to the simulation reality if its too high and drives to inconsistencies. The procedure executed by every agents is described in Algorithm 1. Initially the environment formed by the map and the exits.

4.2 System input

Populations differ on sizes, ages and tolerance to stress. Because of this we model differently each agent. This allows the simulations to be more representative of the real variation. In the previously described

Algorithm 1 Evacuation simulation

```
1:  procedure SIMULATION STEP
2:      initMap()
3:      initExits()
4:      initAgents()
5:      p ← n
6:      space ← ∅
7:      for all a ∈Agent do
8:          space ← space ∪VO(a)
9:      if space ∪U <> ∅ then
10:         move()
11:     else
12:         v ← penalyze(v)
13:         move()
```

model, people have different diameters, velocities and stress. These variables can be distributed among the population using a normal distribution. Because it is different the evacuation of a kindergarten, a high school or a retirement home, the parameters that define the distribution are a parameter of the input. The number of individuals are part of the system input, The population is randomly distributed through the space. The system input consists on: agents attributes, map (optional) and exits. There is a description of these normal distributions using the mean, standard deviation and minimum and maximal values, because distributions tend to $-$inf and inf. The building or space represents the boundaries of the scenario. It is considered a static obstacle meanwhile the agent are dynamic obstacles. The map is represented by a boolean matrix that represents occupied spaces by obstacles or walls and free spaces.

4.3 OpenMP and parallelization

There are several techniques to parallelize the compute in a agent based model. It strongly depends on the interactions between the agents. Space partitioning is a good options when the agents are fixed on the space or when there is no a big number of agents switching between areas. In the first case the division is static and in the second is dynamic because the load unbalance has to be managed. In crowd evacuations all the agents are commanded by the same rules. Because of that the simulation can be parallelized by agent. This allows to follow an SIMD model that is suitable in multicore and manycore architectures.

OpenMP is an API to program multicore architectures. It uses a shared memory model to communicate between threads. We chose this API because it provides a powerful and simple programming interface. A problem presented when the API is used in our problem is the consistency. The next position of two agent cannot intersect in time $t + 1$. When the agents are iterated sequentially inconsistencies can be avoided easily because all the agents can be checked. But when the new position of the agents are computed in parallel there are unknown new position: the positions calculated by other threads. The model needs to consider the next position and is that what is evaluated, and, in the last case, the position of the agent.

5 EXPERIMENTAL RESULTS

The experiments have been carried sing population of 3.000, 6.000 and 12.000 agentsAgents move towards a bi dimensional space represented by coordinates (x,y) stored in an array of structs (AoS). The model was implemented using C and OpenMP. The simulations were carried out using 1, 2, 4, 8, 16, 32 and 64 cores. Experimental times are the average of 100 simulations for every case. The binary was compiled using the maximum optimization level provided by the GCC compiler (O3). The cluster specifications are described in Table 1.

Table 1: Cluster specifications.

Cluster	Details
32 IBM x3550 Nodes	2 x Dual-Core Intel(R) Xeon(R) CPU 5160 @ 3GHz 4MB L2 (2x2) 12 GB Fully Buffered DIMM 667 MHz gcc 4.4.7 OpenMP 3.0

The boundaries are not represented as a static agent, but it is considered an area in the obstacle avoidance process. The agents are spread among the bi dimensional space in a random way checking that the space of two agent do not interact and there are no inconsistence states. Goals that represent the gates of the square are stored in an array of coordinates. Goals are assigned initially to the agents as well as their knowledge.

5.1 Scenarios

We executed several scenarios in order to test our model and simulator. These scenarios check different aspects of the behavior of the agents:

- Agents crossing: a simple case where agents cross the scenario to achieve the opposite position.
- Harmed agents becoming obstacles.
- Evacuation of a group of agents.

Normally, crowd simulations are interested in a big number of agents moving freely thorough the space as in the case of video games or graphics. In the following case the agents are positioned in random coordinates and have a random goal. This allows to test how the agents cross the scenario moving in a crowded environment. In the Fig. 4 the simulation shows how the movement of the agents is determined by the future positions of the neighbor agents. They move, and in case of no possible future positions they stop, waiting for a new free space available.

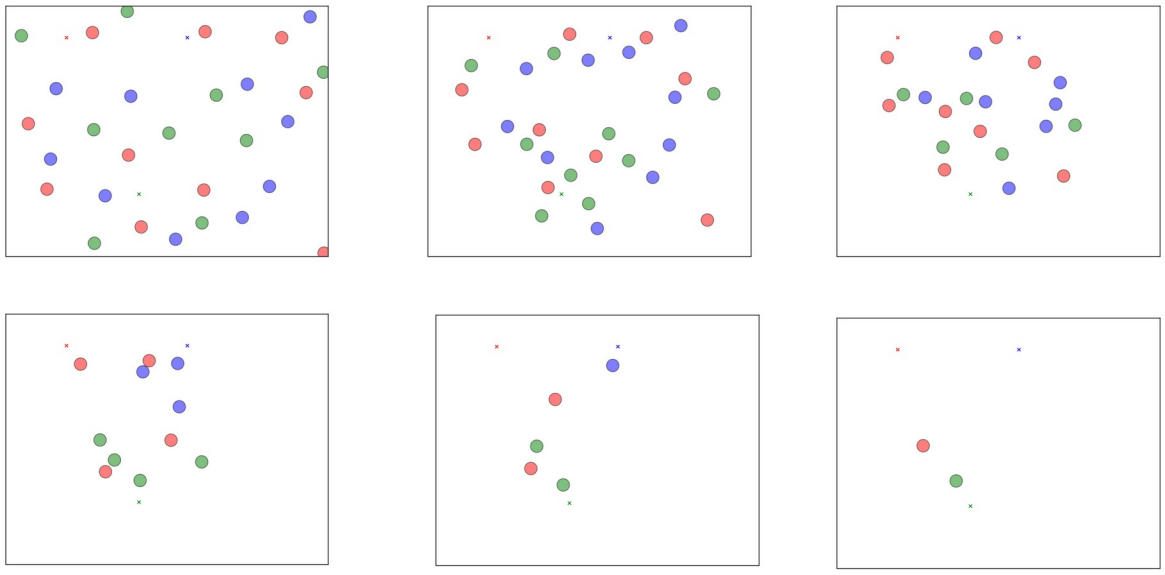

Figure 4: Agents crossing the scenario

Agents getting harmed in a crowd evacuation become static obstacles. We test how dynamic bodies become static obstacles when they get harmed. To test our model, we choose randomly and agent and we set its speed to 0 as seen in Fig. 5 (a) where the black agent is dead and other people avoid it. This agent is harmed in the middle of the simulation and the other agents consider it an agent with speed 0.

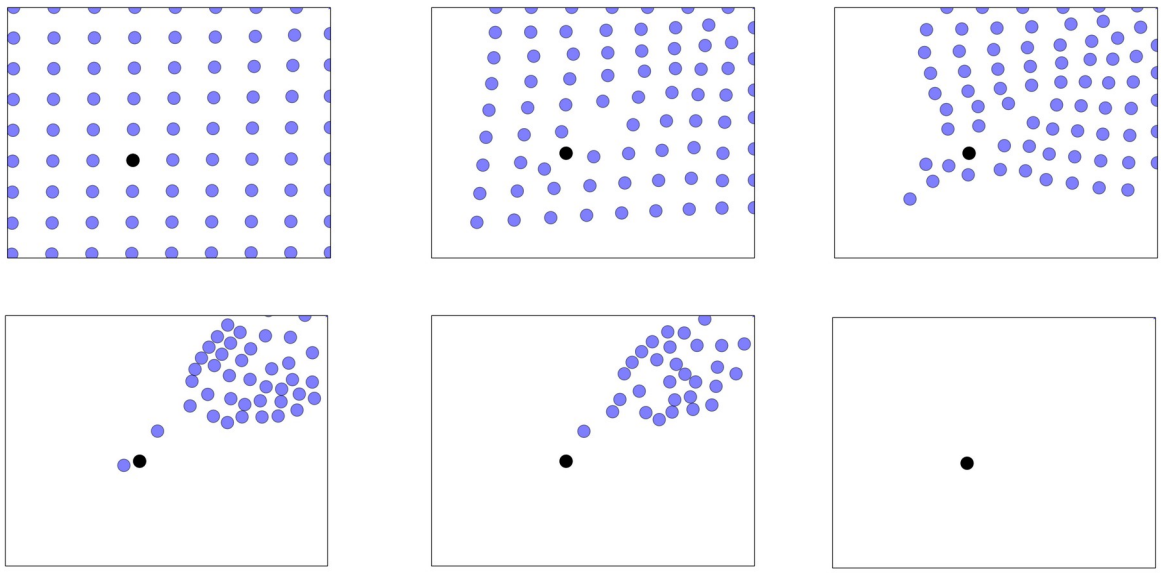

Figure 5: Harmed agents scenario. Agents reaching the goal are removed from the scenario.

In the last example we show our interest case: crowd evacuations. Agents have different exits as shown in the Fig. 6. Agents form bottlenecks around emergency exits and the evacuation slows down.

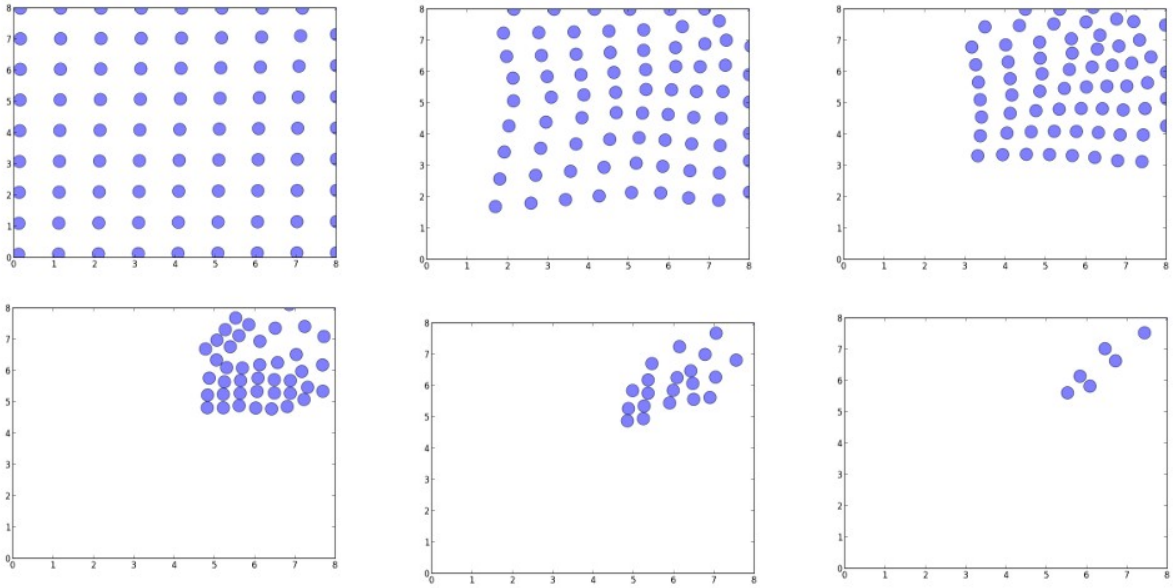

Figure 6: Crowd evacuation

In Fig. 7 are shown the total time of execution for the populations previously mentioned agents. The data is stored in arrays and accessed by all the threads. This is the average time of 50 different executions for every number of agents. The execution time decreases. The red line depicts the linear or ideal speedup

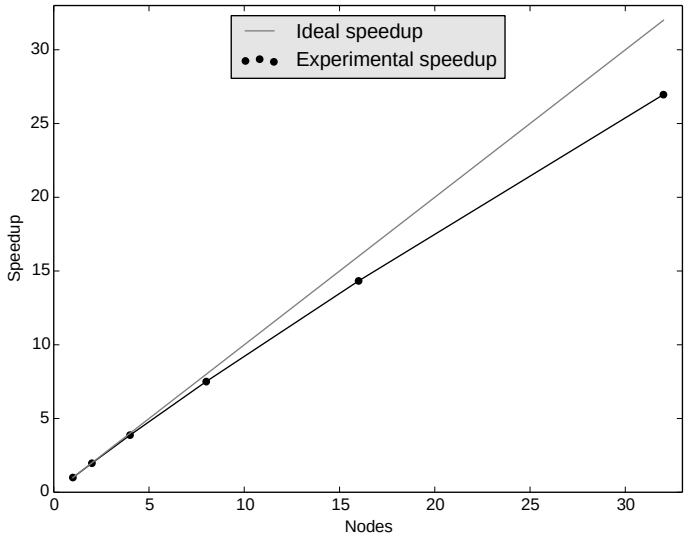

Figure 7: Execution time

6 CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

From the research that has been carried out, it is possible to conclude that:

- The proposed method can be readily used in practice.
- The model avoids collision and supports crowd clogging.
- Shared memory parallelization is a suitable computational model for crowd evacuations parallelization.
- The algorithm scales efficiently as we increase the number of cores.

6.2 Future work

In our future research we intend to concentrate on the realism of the simulation, ther performance and validation techniques. Future work will involve:

- Add a path finding intelligence where agents make mistakes and take bad decisions.
- Reproduce high density situations such as crowd waves.
- Compare other hardware architectures with our algorithm.
- Implement validation techniques for the model.
- Look for performance improvements in the simulator

ACKNOWLEDGEMENT

This research has been supported by the MINECO (MICINN) Spain under contract TIN2011-24384.

REFERENCES

Berg, J. V. D., M. Lin, and D. Manocha. 2008. "Reciprocal velocity obstacles for real-time multi-agent navigation". In *Robotics and Automation*, 1928–1935. ICRA.

Bleiweiss, A. 2009. "Multi agent navigation on the gpu". *GDC09 Game Developers Conference*.

Fiorini, P., and Z. Shillero. 1998. "Motion planning in dynamic environments using velocity obstacles". *The International Journal of Robotics Research* 17.7:760–772.

Grimm, V., E. Revilla, U. Berger, F. Jeltsch, W. M. Mooij, S. F. Railsback, and D. L. DeAngelis. 2005. "Pattern-oriented modeling of agent-based complex systems: lessons from ecology". *Science* 310(5750):987–991.

Guy, J. V. D. B. S. J., M. Lin, and D. Manocha. 2011. "Reciprocal n-body collision avoidance". *Robotics research*:3–19.

Guy, S. J., J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey. 2009. "Clearpath: highly parallel collision avoidance for multi-agent simulation". In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 177–187. Eurographics Association.

Helbing, D., and P. Molnar. 1998. "Self-organization phenomena in pedestrian crowds". *arXiv preprint cond-mat* 9806152.

Helbing, D., and P. Mukerji. 2012. "Crowd disasters as systemic failures: analysis of the Love Parade disaster". *EPJ Data Science* 1.1:1–40.

Korhonen, T., S. Hostikka, S. Helivaara, and H. Ehtamo. 2005. "FDS+ Evac: an agent based fire evacuation model". *Pedestrian and Evacuation Dynamics*:109–120.

Macy, M. W., and R. Willer. 2002. "From factors to actors: Computational sociology and agent-based modeling". *Annual review of sociology*:143–166.

Moussad, M., D. Helbing, and G. Theraulaz. 2011. "How simple rules determine pedestrian behavior and crowd disasters". In *Proceedings of the National Academy of Sciences*, 6884–6888.

Musen, M. A., B. Middleton, and R. A. Greenes. 2014. "Clinical decision-support systems". *Biomedical informatics* 17.7:643–674.

Pelechano, N., J. M. Allbeck, and N. I. Badler. 2007. "Controlling individual agents in high-density crowd simulation". In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 99–108. Eurographics Association.

Varma, V. K., I. Ferguson, and I. Wild. 2000. "Decision support system for the sustainable forest management". *Forest ecology and management* 128.1:49–55.

Zheng, X., T. Zhong, and M. Liu. 2009. "Modeling crowd evacuation of a building based on seven methodological approaches". *Building and Environment* 44.3:437–445.