Multithreading model for evacuations simulation in emergency situations

Pablo CristianTissera[1], A. Marcela Printista[1,2], Emilio Luque[3]

[1] Departamento de Informática, Laboratorio de Investigación y Desarrollo en Inteligencia Computacional. UNSL.
[2] Conicet - CCT San Luis.
[3] Departamento de Arquitectura y Sistemas Operativos Universidad Autónoma de Barcelona - España
{ptissera, mprinti}@unsl.edu.ar
emilio.luqe@uab.es

Abstract.
Evacuation simulations allow to consider preventive measures against possible emergency scenarios. We have developed a simulation model that takes into account not only the characteristics of the environments but also is able to represent social behaviours that would render our models more accurate and realistic. The proposed model has a hybrid structure where the dynamics of fire and smoke propagation are modelled by mean of Cellular Automata and for simulating people's behaviour we use Intelligent Agents. In this paper, a behaviour in panic situation is added to the existing ones. Moreover, as main contribution, this paper explains the implementation of the model in which we apply a functional decomposition in order to accelerate the simulation and take advantage of current computer architectures.

Keywords: Evacuation Simulation, Social Behaviours, Cellular Automata, Intelligent Agents. Multithreading.

1 Introduction

In the last years, several modelling approaches have been proposed to deal with the emergency evacuations because the prediction of the people's behaviour is of great public interest. 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. On the other hand, the microscopic approaches allow to investigate how the system state evolves during the model runs. References to different models may be obtained from [2].

We developed a hybrid model where the environmental dynamics are modelled by means of Cellular Automata (CA), because it are suitable for modelling process of diffusion like fire and smoke and for simulating people's behaviour we are using the intelligent agent (IA) concept. We used a behaviour-based agent architecture because it allows us to work with behaviours beyond than those purely reactive[10,11]. This type of system provides solutions in dynamic and uncertain environments, where the agent has only a partial view of the problem.

The proposed simulation system 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. Our proposal could be used by architects, government agencies, foundations, etc. in order to know the security threats of a possible disaster, help with appropriate actions of prevention for a quick and efficient way to evacuate a building through the design of active policies that minimize the evacuation time when circumstances require it.

Our model is a process that consumes a significant amount of time to simulate a complete evacuation when the environment size and / or the number of people is considerable. The emergence of multi core processors introduces a real challenge for parallel applications, that is to exploit such architectures at their maximum potential. This leads us to develop a model to achieve a competitive performance.

Section 2 describes the Hybrid Model for the Evacuation Simulation, by explaining briefly the environmental and the pedestrian sub models. Section 3 explains the implementation of three behaviours commonly observed in emergency evacuation. In Sub-section 3.1 we explain how to perform the association of an agent with a specific behaviour during the execution of the model. In section 4 we present the Multithreading Model for Evacuation simulation. In section 5 we describe our work with different instances of the problem at hand and report the performance analysis of each case and in section 6 discuss the conclusions and future works.

2 Simulation Model

The model consists of two sub-models, called environmental (EsM) and pedestrian (PsM). This model along with the computational methodology allow us building an artificial environment populated with autonomous agents, which are capable of interacting with each other. The Fig. 1 shows the hybrid model.

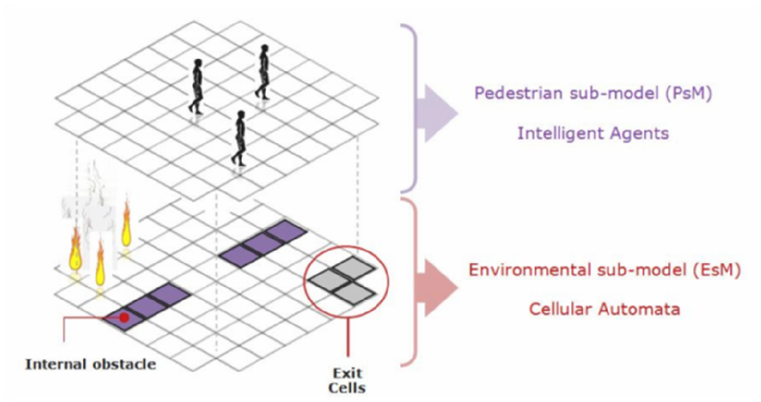


Fig. 1. Hybrid model consisting of environmental and pedestrian sub models

The details of the hybrid model have been reported in [1, 3], and for reasons of space they will not be reproduced in this paper. We briefly mention only those aspects that are necessary to understand the proposal.

The EsM describes the spatial configuration of the environment (geometry of space, exit doors, internal barriers, etc.) and models the processes of diffusion of smoke and fire. The EsM is based on CA, which are discrete dynamic systems that have the capacity to develop complex behaviours from a simple set of rules [9]. Basically, these rules will allow to specify the new state of a cell based on the state of the neighbouring cells.

The PsM uses the concept of intelligent agents to describe the cognitive processes of individual agents and interactions among multiple agents in a specific environment. Through interaction and coordinated evolution of these two sub-models it is possible to obtain a model capable of simulating indoor environments with a finite number of exits that must be evacuated by a group of people due to the threat of fire and the effect of the smoke.

In the proposed model an agent is placed on an environment described by a bi-dimensional grid where they can find different elements such as walls, obstacles, exits, presence of smoke, fire and other agents. The agent architecture is illustrated in Fig. 2 (left).

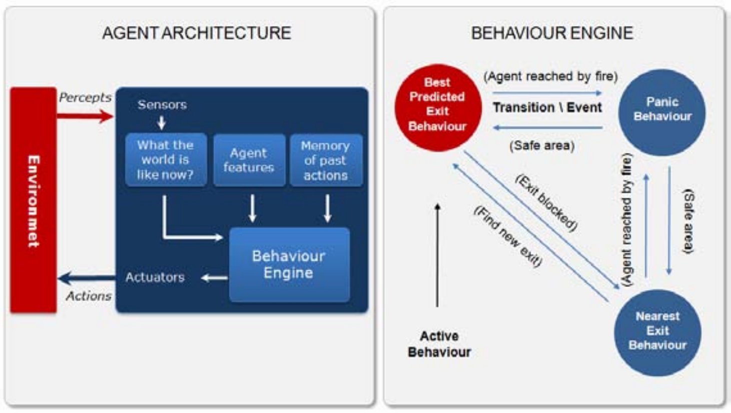


Fig. 2. Agent Architecture and Behaviour Engine

In our model, agents respond to a behaviour-based architecture because one of the major drawbacks of this type of system is that multiple behaviours with different objectives may be attempting to take control of the agent at the same time. To solve this problem, known as the action selection problem [7], it is necessary to develop a mechanism that allows us to select the appropriate behaviour in a given situation. In our model each agent has an associated behaviour engine, shown in Fig. 2 (right) that manages decision-making processes. This engine is a nondeterministic finite automaton, where each node represents the implementation of behaviour while the transitions represent the event for which the agent can change the state.

This arbitration state-based mechanism [8], selects an appropriate behaviour to deal with the current situation from a determinate event detected in the environment

[7]. In this way, an agent can change its behaviour during execution of the model according to a predetermined set of rules that serve as triggers for this change.

3 Primitive behaviours

In the current state of development, the simulator has the capability to implement three behavioural categories.

In the *Nearest Exit behaviour* (NE_B), the agent will try to get out the exit closest to its current position. In this behaviour the decision process will take into account the position of the agent, the direction toward the nearest exit, the state of its environment in relation to the progress of fire and smoke, but it ignores information from other alternative solutions, the behaviour of other agents and it will not take unexpected or altruistic decisions.

In the *Best Predicted Exit behaviour* (BPE_B), the agent will analyse different exits and choose one that it predicts the fastest exit to evacuate. The decision process will take into account the position of the agent, the state of its environment in relation to the progress of fire and smoke, the distance to alternative exits, the density of crowd trying to evacuate for each exit (only if the agent can see the exit) and the stress level in relation to its tolerance to it. As the evacuation progresses, the agent is predicting the cost (in time) to evacuate by each of the exits that are available in the environment. The inferred lower cost will indicate the best exit. For that, the decision-making process evaluates a cost function that indicates which is best exit.

The resulting procedure instructs the agent to which exit to go. In addition, the procedure involves two dynamic factor used to adjust the number of times the agent executes the action to evaluate the exits[1]. This is done to limit the effect of indecision of the agents. This factor depends of a environment size and it is dynamically adjusted according the time elapsed since the start of the evacuation.

Finally, in the *Panic behaviour* (P_B), unlike the previous behaviours, the agent does not realize any type of analysis over the exits. An agent assumes this behaviour in a situation of extreme danger, when the agent's current position has been achieved by the spread of fire. In such situation, the agent only analyzes their position and the state of their environment in relation to the progress of the fire and smoke. The decision-making process of the agent evaluates the condition of the cells in its proximity searching cells that remove it from the fire, without mattering if these cells offer it or not a better position respect of the exit chosen, that is to say, the agent tries to escape and to reach a sure position without presence of the danger.

3.1 How does the behaviour engine work?

So far we have only defined the behaviours implemented in the model, but we have said nothing yet about how these are related through the behaviour engine, giving origin to agents who can change its behaviour along the simulation. Before, it should be noted that the agents at the beginning of the simulation have an assigned behaviour, but as the simulation progresses, could arise different situations in which it is suitable or even imperative change the behaviour of the agent. Next, we will discuss the *Events* that can lead to an agent to change their behaviour:

Agent Reached by Fire and Agent in Safety Zone: Any agent reached by the spread of the fire along the simulation (agent reached by fire event), regardless its behaviour, detects the situation and changes its current behaviour by the P_B with the purpose of going out of the situation of danger of in an immediate way. Once the agent has reached a safe area, the same resumes its original behaviour (safety zone event) with the objective of continuing the evacuation by the selected exit.

Blocked Exit: An agent whose behaviour is BPE_B, uses two parameters to determine the amount of inferences that the agent can perform and how often these inferences can be done [1]. It is possible that along the evolution of the model, an agent has made all possible inferences and taken its last decision, therefore, the exit to which it is addressed cannot change, but it can happen that after taken the final decision, the exit will be blocked due to the spread of the fire, then the agent already cannot evacuate for this exit. When this situation is detected an agent changes its behaviour by NE_B, because it must select a new exit to evacuate but has already exhausted all the instances that allowed it choose an exit.

Select New Exit: An agent whose behaviour is NE_B, along the evolution of the model may be in a situation where it cannot move towards the selected exit because this exit is too congested. When the agent detects this situation and according to its stress level may or may not change their behaviour in a probabilistic way by BPE_B, with the purpose of find a new exit that allow it evacuate more quickly.

Although, currently in our implementation only we have defined a few events of change of behaviour, it is necessary to emphasize that our model allows easily add new events and new behaviours that will allow describe better the reality and therefore improve the model in a progressive way.

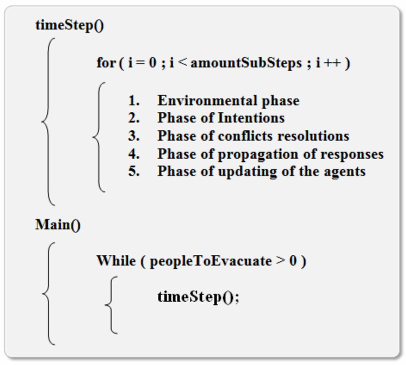
4 The Multithreading Model

The model proposes the execution of so many time steps as necessary until the last alive individual in the environment has been evacuated. The model evolves to discrete steps of time, which leads to discretize the progress of an individual pedestrian, however the movement of a crowd should appear as a continuous phenomenon.

To solve the problem caused by the discretization, our model introduces the execution of sub-time steps between two consecutive time steps [2,3]. As can be seen in Fig. 3, in every sub time step, five phases are executed (*Environmental phase, Phase of Intentions, Phase of conflicts resolutions, Phase of propagation of responses and Phase of updating of the agents)*. In the following, we give a short description of each phase.

Environmental phase: Is responsible to evolve the environmental sub model. In this phase, the evolution rules of the CA for the spread of fire and smoke are applied. After that, this phase should also re calculate the distances from each cell to each exit, due to the spread of fire modifies the environment in which agents must find the way to the exits.

Phase of intentions: this is the first phase in the pedestrian sub model evolution. During this phase, each agent writes a intention of movement in the cell to which one wants to move (target cell). The decision of which is the target cell is determined by current behaviour of the agent. It should be noted that a target cell can be empty or occupied by another agent and can also receive more than one intention since more than one agent may intend to move to it.



```
timeStep()
{
        for ( i = 0 ; i < amountSubSteps ; i ++ )
        {
            1.  Environmental phase
            2.  Phase of Intentions
            3.  Phase of conflicts resolutions
            4.  Phase of propagation of responses
            5.  Phase of updating of the agents
        }
}
Main()
{
        While ( peopleToEvacuate > 0 )
        {
                timeStep();
        }
}
```

Fig. 3. Main structure of the model.

Phase of conflicts resolutions: This stage is responsible for resolving existing conflicts in the cells with more than one intention of movement. During this phase the agents will receive a first response to its request for movement. The response that each agent will obtain can be {*accepted movement, denied movement, uncertain movement*}.

If cell in conflict is empty: the conflict resolution process selects an agent between all candidates to occupy the cell in conflict in the next sub time step[1], therefore this agent will obtain a *accepted movement* response, while the rest of the candidates will obtain a *denied movement* response. The process gives priority to the selection of agents with greater speed and fewer points of damage (agent parameter). If the conflict persists, the selection will be random.

If cell in conflict is currently occupied by another agent: the process will check if the cell could be free in the next sub time step. If it will be free, the same procedure of the empty cell is executed. If there is no possibility of the cell to be unoccupied, all agents candidates will receive the response of *movement denied.*

Finally, in the case that it is no possible determine if the target cell will be free in the next sub time step, due to the fact that the movement of the agent depends on the response of another cell currently occupied by another agent and so on, the agent receives as response to its request *uncertain movement*. As the simulation progresses, the possibility of occurrence of this case increases, since the agents tend to gather in crown in the vicinity of the exit and therefore their movements depends on people who are several positions later. This type of conflict is solved by the following phase.

Phase of propagation of responses: The responsibility of this phase is the propagation of received responses by agents in the previous phase. In this way if an agent has received as response *uncertain movement*, at this stage its movement is accepted or denied.

During subsequent sub steps, all agents with *uncertain movement* will remain in this state. Once one of them receives an *accepted* or *denied movemen*t in some sub step, then it will start a backward propagation of novelty, resolving several conflicts in the process. With the purpose to make it clear the operation of this phase we will exemplify different situations that can occur. Suppose that we will call A to an agent that attempting to move into a cell occupied by an agent B, which in turn wants to move to a cell occupied by an agent C. Clearly the agent A cannot move because it is not possible to determine if the agent B will move. In a similar way the same thing happens with the agent B. But once the agent C receives its response of movement accepted or denied, this will spread its response to agent B which can propagate its response to the agent A. In this way the conflict can be solved. Now, suppose the same previous situation, but with the difference that the agent C wants to move to the position of the agent A. In this situation we are in the presence of a cycle and therefore a deadlock situation since the agents will not move because they are waiting for a response that will never come. To solve this deadlock situation, this phase can detect the cycles and all the agents in a cycle receive a *denied movement* response. At first sight the answer given to the agents may seem arbitrary, why the agents did not get a *accepted movement* response?. This is so, because there can be agents who try to move to a position occupied by other agent which is in a cycle, but these agents do not belong to the cycle. This situation can have a large number of variants when working with thousands of agents, therefore it seems reasonable to give *an movement denied* response, since in the next sub times steps the conflicts will be solved.

Phase of updating of the agents: Finally once all agents have its answer, the position of the agents is updated.

It is important to emphasize, that there is a clear division of tasks between the phases mentioned above. While the environmental phase is responsible to carry out the evolution of the environmental sub model, the four remaining phases are responsible for evolving the pedestrian sub model.

Our proposal is aimed at carrying out a parallel shared memory model where it is possible to perform task-level parallelism, since a set of threads con solve the environmental sub model while another set of threads solve the pedestrian sub model. The multiple threads assigned of the same task assist in the resolution of disjoint areas of the grid in a data parallelism way.

To do this then we will see how to perform the update of the cells in each sub time step.

As mentioned above, our model uses a CA (sub environmental model), where agents are positioned. At the time to evolve a CA, it is necessary to have an auxiliary structure which saved the next states of all cells of the automaton solved by means of the application of the rules of evolution. In this way, will we have then two CA, the first will represent the state at time *T* of the CA, while the second (in built) represents the condition of the CA at time *T+1*. Once completed the process, the new representation becomes the new current state of the automaton and the process repeats. In the case of the sequential implementation of our model, there are no problems at the moment of updating of the cells. This is because when the pedestrian sub model begins with the execution of its phases, the environmental sub model has been fully resolved. It is important to emphasize this point, because rarely it is expected that an individual try to perform a move to a position occupied by the fire.

This is achieved because the environmental model has been solved and therefore it is possible for the agent to be able to determine its next position by looking at the state of the automaton at the next time step to avoid cells with fire. This present a problem in the Multithreading Model since, both spread of fire as the agent evolution are been executed in parallel.

To solve this situation, our proposal is to advance in a time step the resolution of the environmental sub model. That is, while the threads of the pedestrian sub model use the structure of the time *T* to obtain the pedestrian configuration of the time *T+1*, the threads entrusted to solve the environmental sub model will be using the structure of the time *T+1* to obtain the environmental configuration of the time *T+2*. In this way, is possible that the agents can have a forward vision of the environment already resolved as the case of the sequential implementation of the model, as can be seen in Fig. 5.

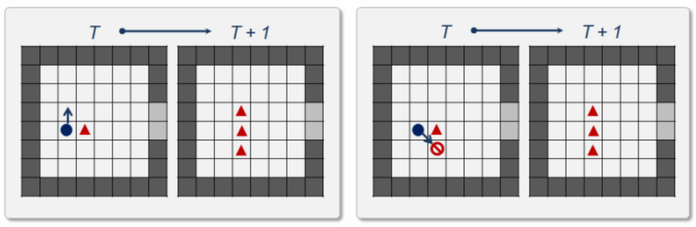


Fig. 4. The agent (circle) in T sees the fire (triangle) in T+1 and then it rules out those cells as next move.

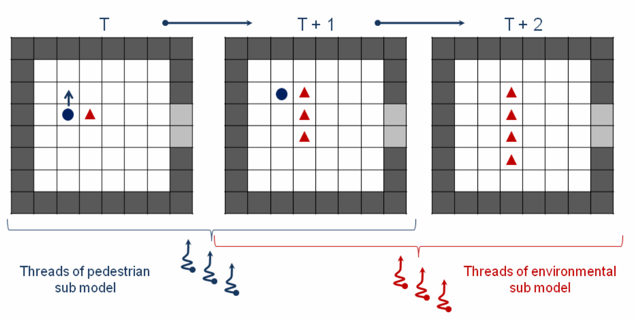

T
T + 1
T + 2
Threads of pedestrian sub model
Threads of environmental sub model

Fig. 5. Threads of pedestrian and environmental sub models.

5 Test-Case Scenario and Results

In this section, we present the simulation results of the explained research. The experiments were carried out with EVACOMP, a hybrid simulation system based on cellular automata and intelligent agent. EVACOMP is a system developed in C and OpenMP and uses the graphical interface (off line) of EVAC Simulator [1,2]. The experiments consider two environment configurations of the buildings to be evacuated (*A, B*):

- *A, 60 × 30 m^2*, three exits of 4 m. each and 2500 pedestrians distributed evenly (50% NE_B and 50% BPE_B).
- *B, 80 × 60 m^2*, three exits of 6 m. each and 4500 pedestrians distributed evenly (50% NE_B and 50% BPE_B).

The experiments are designed to test the performance of the EVACOMP vs. the sequential implementation of the model. With the purpose of obtaining acceptable statistical data, the results shown below correspond to the average of 50 independent replications of each experiment. All execution times values are expressed in seconds and we always set up a thread by core. From the results showed in Table 1, for the Experiment A is possible to visualize, that the best execution times were obtained using 4 threads, and therefore it is where the major speedup is obtained. In this case, while it is possible to see a reduction in execution times, the speedup obtained seems relatively mild. We develop the second experiment, where we increase the size of the environment and the quantity of individuals with the purpose of increasing the quantity of work necessary to solve the model.

Table 1. Execution Times, Speeup and Efficiency values for experiments A and B.

	Sequential	***Threads 2***	***Threads 4***	***Threads 8***	***Threads 16***
Execution Time Experiment A	27,37	19,29	17,16	17,75	22,74
Execution Time Experiment B	439,85	268,12	170,55	134,19	206,41
Speedup Experiment A	X	1,41	1,59	1,54	1,20
Speedup Experiment B	X	1,64	2,57	3,27	2,13
Efficiency Experiment A	X	70,91 %	39,86 %	19,26 %	7,52 %
Efficiency Experiment B	X	82,02 %	64,47 %	40,97 %	13,31 %

As we see in Table 1, the execution times for Experiment B increased significantly compared to the first experiment. Here, the best execution times were obtained using 8 threads, and therefore it is where the major speedup is obtained. It is important to highlight for this case, that the obtained speedup is acceptable and in addition it is better than the speedup obtained for the best case of the first experiment. By making this comparison, it is possible to appreciate, that both the speedup and the efficiency obtained improve for the second experiment, which is a good indication of which on having increased the load of work in the system the performance of the parallel model improves. Because to the orientation of this work and for reasons of space, we do not report the information about times of evacuation and travelled distances by the individuals. A wide series of experiments can be consulted in [1,2,3,4], where the empirical values obtained for the evacuation time are comparable to other implementations, which have validated their results against real evacuation exercises [5] [6]. For the experiments, we used a multicore equipment with 4 processors AMD

Opteron 6128, 2.0GHz (8C), and RAM memory of 64GB Memory (16x4GB), 1333MHz.

6 Conclusions and Future Works

We presented a model capable of simulating indoor environments with a finite number of exits that must be evacuated by a group of people due to the threat of fire and the effect of the smoke. The proposed model consists of two sub-models, the Environmental Model (EsM) and Pedestrian Model (PsM). The EsM, based on CA, manages the spatial configuration of the environment and models the processes of diffusion of smoke and fire. The PsM is the part of the hybrid model focuses on representing the human behaviours.

The proposed model to perform both task-level and data-level parallelism, where a group of threads will be responsible to evolve the pedestrian sub model pedestrian and another group of threads will be responsible to evolve the environmental sub model. While our development is not yet complete, the results of our Multi Threading implementation presented here are encouraging.

As future works, it is important to decide the optimal size of each set of threads to solve each sub model. Our model is going to use some strategy that will enable us to achieve an optimal balance in the allocation of threads to the resolution of each sub model.

References

1. P. C. Tissera, A. Castro, A. M. Printista, E. Luque, Evacuation Simulation Supporting High Level Behaviour-Based Agents, Procedia Computer Science Vol. 18, 2013, pp. 1495-1504, proceedings of the International Conference on Computational Science, ICCS 2013.
2. P. C. Tissera, A. M. Printista, E. Luque, A hybrid simulation model to test behaviour designs in an emergency evacuation, Procedia Computer Science Vol. 9, 2012, 266-275, proceedings of the International Conference on Computational Science, ICCS 2012.
3. P. C. Tissera, A. M. Printista, E. Luque, Implementing sub steps in a parallel automata cellular model, in: Computer Science and Technology Series-XVII Argentine Congress of Computer Science-Selected Paper, 2012, pp. 81–93.
4. P. C. Tissera, A. M. Printista, M. Errecalde, Multi-column partitioning for agent-based ca model, in: HPC Proceeding. JAIIO, SADIO-Argentina, 2011.
5. Aik Lim Eng. Exit-selection behaviors during a classroom evacuation. International Journal of the Physical Sciences, Vol. 6 (13), 2011, pp. 3218–3231.
6. Klupfel Hubert Ludwig. A Cellular Automaton Model for Crowd Movement and Egress Simulation. PhD thesis, Universitat Duisburg-Essen, 2003.
7. P. Pirjanian, Behavior coordination mechanisms - state of the art, Tech. rep., USC Robotics Research Laboratory, University of Southern California, 1999.
8. A. Saffotti, The uses of fuzzy logic in autonomous robot navigation, Soft Computing Vol. 1 (4), 1997, pp. 180–197.
9. Wolfram Stephen, Cellular Automata and Complexity, Addison Wesley, USA, 1994.
10. P. Maes, The dynamics of action selection, in: IJCAI-89, MI, 1989, pp. 991–997.
11. R. A. Brooks, A robust layered control system for a mobile robot, IEEE Journal of Robotics and Automation, 1986, pp. 14–23.