Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 353 (2020) 61–75

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Evaluating the Safety of Crowds in Enclosed Spaces by Markovian Agents

Enrico Barbierato

Dip. di Matematica e Fisica
Università Cattolica del Sacro Cuore, Via Musei 41
25121 Brescia, Italy,
enrico.barbierato@unicatt.it

Marco Gribaudo

Dip. di Elettronica, Informazione e Bioingegneria,
Politecnico di Milano, via Ponzio 34 5,
20133 Milano, Italy,
marco.gribaudo@polimi.it

Mauro Iacono

Dip. di Matematica e Fisica,
Università degli Studi della Campania "L. Vanvitelli", viale Lincoln 5,
81100 Caserta, Italy,
mauro.iacono@unicampania.it

Alexander H. Levis

Department of Electrical and Computer Engineering,
George Mason University,
Fairfax, VA, USA,
alevis@gmu.edu

Abstract

Different paradigms have been used to model the behaviour of pedestrians in a crowd. While some approaches analyze the modelization of crowd motion from a physical perspective, other techniques prefer to focus on similar abstractions following classic paradigms, such as mobile agents or cellular automata, or even from the point of view of social forces interaction. Particular interest has emerged around the topic of crowd behaviour under condition of stress caused by a dramatic event, such as an earthquake, a terrorist attack or a fire. This paper presents a novel approach including a GSPN to describe the behaviour of an individual, which is subsequently translated into a set of Markovian Agents. The model under study reproduces a scenario where a fire, developed in a closed environment, triggers erratic crowd behaviour. It is possible to perceive that a panic situation can be mitigated by enforcing certain measures involving human leaders or small embedded systems (such as Internet-of-things enabled devices), reducing the level of risk related to safety.

Keywords: Crowd motion, fire, closed spaces, Markovian Agents.

https://doi.org/10.1016/j.entcs.2020.09.019

1 Introduction

When emergency situations arise in closed spaces, people's behavior represents a critical factor to ensure physical safety and to avoid casualties and injuries. In similar situations, people's conduct might be illogical, because of panic, and selfish, producing situations that could raise the level of potential danger with respect to the inherent risk due to the emergency itself. This problem can be mitigated by proper training, if potentially involved people are already known in advance and somehow classifiable in a category (i.e. workers in a closed environment, such as an office).

In order to be able to predict people's reaction to unforeseen emergencies and organize suitable training, models must be able to convey the dynamics of spontaneous panic responses into organized behavior. Abstracting these dynamics requires considering social forces driving individuals, and accounting for factors such as the respect of personal space, or the instinct to avoid obstacles. When studying crowd motion, group dynamics (such as a speed gradient between individuals and the crowd speed, the violation of personal and safe space, mass behavior due to propagation of information, instinctive aggregations near exits or obstacles like arching and clogging) emerge.

This article deepens a previous work presented in [4], where *Markovian Agents* (MA, [5]) were used to model and analyze the behavior of people in a closed environment where a panic situation occurred. We present a more complex scenario, in which the crowd motion is studied under a stressful situation and managed by means of visual signals positioned in different places in the office. Individuals have roles also and evacuation follows given guidelines, previously communicated to each member.

The remaining work is structured as follows: in section 2, the authors review the most relevant work on crowd motion and dynamics in different contexts; section 3 shortly presents the theory of MAMs; section 4 discuss a case study. Finally, section 5 draws conclusions and provides insight about future work.

2 Background and related work

Suitable approaches to model crowd's behavior consist typically of i) Cellular Automata, ii) Flow-based Modeling and iii) Multi-Agent Systems, though other interesting experiences are documented in literature.

The modelization of a dramatic event, such as the propagation of a fire, has been studied in different contexts. For example, in [8] the authors consider an open environment where a fire is considered by means of Markovian Agent, studying how the weather conditions, such as wind, and external barriers can significantly affect the flames motion. The propagation of a fire between buildings is analysed in [14] by means of a numerical simulation: the authors take in account numerous

physical factors, such as the temperature increase due to multiple fire plumes, the thermal radiation and so forth. A similar approach can be found in [13], where the fire propagation is examined by considering the effect of the burning rate per unit exposed surface area and a function of a porosity factor.

However, these studies aim at simulating how fire can spread in closed or open environments, without regard of the crowd motion in distressful circumstances.

In [16], Inga deployed a Generalized Stochastic Petri Nets (GSPN) to model (and also predict) the evacuation of a crowd in a building. Specifically, the GSPN reflects the architecture of the offices in a building (office rooms are modeled by places whilst doors can be either immediate transitions (simulating a door opening) or delayed transitions (simulating the act of leaving a room).

In [2], Almeida and others discuss the behavior of a crowd in condition of panic by analyzing conditions of severe stress such as flocking, herding, clogging and arching.

Abu Bakar and others review some modeling techniques in [1], (for instance, Agent-Based Simulation (ABS), Social Force Simulation (SFS) and an hybrid of the two) of fire evacuation in a closed space, though the actual simulation is postponed to future work. An in-depth discussions about the modelization of a pool of human, dynamic crowd simulation, stress and fears models is presented in [3]. Different approaches, such as models based on the leader-follower paradigm, have been presented in [9], where the authors study a scenario, consisting of a collective adaptive system (CAS), by the perspective of a leader, performing a random walk, and an agent acting as follower. The model is further analyzed by the process algebra PALOMA [10].

Cellular automata [19] [15] [22] are based on a finite grid-based representation of the environment, where *cells* describe the local state of the system, which evolves in steps according to its previous state and the state of near cells with proper rules and influence radius, depending on the kind of problem modeled. Applications span over video games related map evolution, human behavior analysis, support to architectural planning, traffic management in cities and artificial intelligence behavioral models. This approach is naturally fit to represent obstacles and exhibits significant computational advantages, because it is generally not computational intensive and easily parallelizable, thus it can scale up with minimal burden. On the other hand, it is not specially fit, due to the cell-oriented representation, to applications that are bound to measure speed and tracing of individuals.

Flow-based models are based on the physics and engineering domains, and are specially suitable to understand the emerging behavior of aggregates of moving particles as a whole, including border effects or pressure-like behaviors and density variations. Applications to crowd are documented: e.g. EVACNET4 [20] provides computing of optimal evacuation plans in buildings by a flow-based simulation of a model representing a network of nodes on which a room, a stair or a hallway may be mapped and a number of people flowing through these spaces by following the network connections, which may be characterized by traversal time.

Multi-Agent Systems (MAS) are fit to modeling crowds, as they are founded on

abstracting the behavior of individuals with a consistent and rich framework and complex interactions. An agent is driven by an intelligent reactive strategy that is articulated in Belief, Desire and Intention [18] to confer it autonomous decisional abilities. Agents interact and react, and can collaborate or compete: their evolution is described and computed individually in an environment of which the emerging global state can be observed as a result.

Noticeable are techniques based on *swarm intelligence*, in which emergent solutions are generated as a kind of collective intelligence of a population of single agents [16] interacting and collaborating individually on a stochastic or chaotic basis. *Markovian Agents* (MA) [5] are agents that are characterized by a stochastic behavior that is described by a discrete-state continuous-time finite Continuous Time Markov Chain (CTMC), in which interactions between agents are a component influencing the CTMC infinitesimal generator. MA interact with the environment in which they operate as well, allowing a flexible description of it in terms of obstacles, distance-like or path-like propagation and similar features. Interactions between MA are shaped as a message-based communication and occur by means of an individual perception function, allowing broadcast communication as well.

3 A Petri Net description for Markovian Agents behavior

Markovian Agents [5] is a formalism to describe spatially distributed systems where agents have a finite number of states, and their dynamics is described by a transition kernel, which consists of an *induced transition matrix* and a *local transition matrix*. The agents are distributed across several locations: as a result, a model that is based on a set of interacting Markovian Agents is said to be a Markovian Agent Model (MAM). The formalism is interesting because it can model distributed systems in which local behavior can be distinct for each agents.

In general, MA behavior is described by graph based state-transition diagrams such the one shown In Fig. 1. In a MA, the states are represented by circles $i, j, \ldots k$, which can be considered the states of a CTMC. However, with respect to a model based on a CTMC, a transition to another state occurs either because of the local behavior of the entity (represented by a solid line that models a failure or a reaction of some sort) or an interaction with another MA (indicated by a dashed line).

Although this representation has several advantages, it is based on a low-level language, that might require a large number of states to describe complex dynamics. Moreover, the interaction dynamics with other agents is not explicitly visible from the models, since it is embedded in functional dependency on the global state of the system, assigned to the transition arcs. This work presents a Petri Net based agent behavior description, which can be used to overcome most of these limitations. In particular, we describe the behavior of an agent, with the Petri Net elements presented in Fig. 2. `Place`, `Token`, `Transition`, `Arc` and `Inhibitor Arc` have

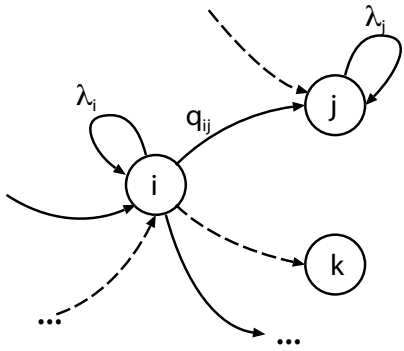

Fig. 1. Schematic structure of a Markovian Agent.

the conventional SPN semantic described for example in [17]. Places holds a non-negative integer number of tokens, which can be transferred to other places thanks to the firing of a transition. Each transition has an exponential firing time distribution associated, and can be enabled if conditions expressed by arcs holds. Arcs are characterized by an integer value called the *weight*: a transition is enabled if all the places to which it is connected by an arc that ends on it have a number of tokens that is greater or equal to the corresponding weight, and if all the places to which it is connected through an inhibitor arc have strictly less tokens than the corresponding weight. Enabled transitions fires after a random time determined by the associated exponential distribution: when a transition fires, it removes from the places connected by incoming arcs an amount of tokens equal to the corresponding weight. Firing also adds to each place to which the transition is connected by and outgoing arc, as many tokens as the corresponding weight. When a place and transition are connected with both an in-going arc and out-coming arc characterized by the same weight, the effect is of enabling the transition when there are enough tokens in the input places, but not changing their marking when the transition fires. `Test Arc` primitives, represented by an arc with an arrow on its two extremities, explicitly show this behavior, by allowing to enable a transition, without changing the marking of their input place at the firing time.

The two new features of the proposed modeling language are the `Remote Place`s and the `Parametric Transition`s. The latter explicitly represents events whose rate depends on the location where the agent is located: in other words, the rate parameter of the corresponding distribution is a spatial property, which depends on the location where the agent is positioned, and which is then used to model actions induced by the environment where the agent is operating. They act however exactly as the other transitions, following the same habilitation and firing rules. `Remote Place`s represents instead places of different agents, positioned either in the same location, or in different locations of the same model. One of the key restrictions however, is that remote places can be connect to transitions only with test or inhibitor arcs. In this way, one agent cannot change the state of other agents, but it can be influenced by their state.

All primitives, except `Remote Place`s, have associated a unique symbolic name $\eta \in \Sigma^*$, where Σ denotes an alphabet, and Σ^* the set of all possible strings on it. η

is used not only to name primitives (as common in modelling languages implemented in most Petri Net tools), but also to define parametrization and interconnection rules of the considered agents. In particular, `Remote Places` have associated a symbolic formula, which works on places belonging to other agents. This paper considers only one kind of formula, that is $\texttt{SUM}(p^{\psi})$, where $p \in \Sigma^*$ is the name of the remote place that is being interconnected, and $\psi \in \mathcal{N}$ is an index that might be used in case the same remote place is used more than once in an agent. In this case, $\texttt{SUM}(p^{\psi})$ corresponds to the sum of the token contained in place p for all the agents in the interconnected location, which will be specified in the topology definition of the model that will be detailed below. Future work will consider more advanced formulas, taking advantage of syntaxes similar to the one used in SQL aggregation functions when the `GROUP BY` clause is used, which considers for example functions such as $\texttt{MAX}(p^{\psi})$, $\texttt{MIN}(p^{\psi})$ and $\texttt{AVG}(p^{\psi})$ that respectively return the maximum, minimum or average umber of tokens in place p of the interconnected agents.

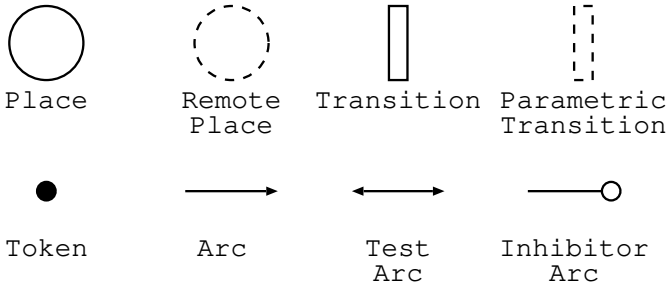

Fig. 2. Schematic structure of a Markovian Agent.

A *Petri Net Markovian Agent Model*, can then be described by a set of agent classes $\mathcal{C} = \{C_c, \ldots\}$ specified by a different model designed using the primitives of Fig. 2, plus a description of the environment $\mathcal{V}$, which is considered to be discrete and composed of a set of locations $\mathcal{V} = \{\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_N}\}$. Each location $\mathbf{v_i} = \{c_{j@i}, \ldots\}$[1] (with $1 \leq i \leq N$) is defined by the set of agent class instances that can be found in that location. Each instance $c_{j@i} \in \mathbf{v_i}$, is a tuple $c_{j@i} = (C_{c_{j@i}}, N_{c_{j@i}}, Par_{c_{j@i}})$ containing the following information: the *class definition* $C_{c_{j@i}} \in \mathcal{C}$, the *class population* $N_{c_{j@i}}$, and the parameter assignments $Par_{c_{j@i}}$. In the following, to simplify the notation, we will write $c \equiv c_{j@i}$ The class definition C_c corresponds to one of the classes that composes the model, and N_c the number of agents of the particular class present in that location. Parameters $Par_c = (PTR_c, CON_c, RPC_c)$ can be of three types: *parametric transition rates* (PTR_c), *constants* (CON_c) and *remote places connections* (RPC_c). Parametric rates $(p, r) \in PTR_c, r \in \mathcal{R} > 0$, associate firing rates to parametric transitions. Constants $(p, n) \in CON_c, n \in \mathcal{N}$ defines integer constants that can be used to define both initial marking for places and weights for arcs. Remote place connections consists instead in tuples $(p, v), v \in \mathcal{V}$, which connect the agents being instanced $c_{j@i}$ with the ones in another location v. In all the three types of tuple, component $p \in \Sigma^*$ represents a symbolic name, which uniquely identifies a primitive of an agent.

[1] The notation $j@i$ is read as *j at position i*.

A visual representation of the model topology is given with the box representation shown in Fig.3.

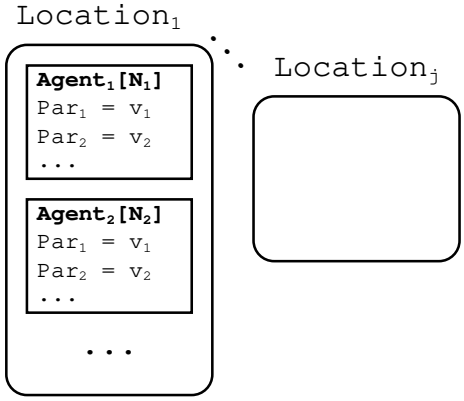

Fig. 3. Graphical formalism to define topologies.

The solution process first computes the state space and the transition matrix for each class instance $c_{j@i}$. Specifically, it follows the conventional techniques of state space generation defined for Petri Nets with the following extensions. Firstly, remote places are not considered in state space generation: since they are connected only to transitions with test and inhibitor arcs, they have no impact on the set of reachable states of an agent. Before generating the state space of the model, parameters are replaced with their actual value assigned by the topology definition. Remote places are then used to create guard functions that multiply the rate of the transitions to which they are connected, implementing the enabling rules defined by the test and inhibitor arcs. In this way, for each agent instance $c_{j@i}$ in each location $\mathbf{v_i}$, an infinitesimal generator $Q_{c_{j@i}}(\mathbf{X})$ is computed. Here $\mathbf{X} = (x_k^{c_{j@i}}, \ldots), \forall c_{j@i} \in \mathbf{v_i}, \mathbf{v_i} \in \mathcal{V}$ represents the complete state of the model, and $x_k^{c_{j@i}}$ accounts for average number of agents in state k of the instances $c_{j@i}$ in position $\mathbf{v_i}$. Let us call $\mathbf{X}_{c_{j@i}} = (x_k^{c_{j@i}}, \ldots)$ the sub-vector of $\mathbf{X}$ accounting for instances $c_{j@i}$. Using the conventional mean-field approximation of the counting process, as it is regularly done for Markovian agent based models, we can compute the transient evolution of the state of the model by solving the following set of ordinary differential equations:

$$\frac{\mathbf{X}_{c_{j@i}}(t)}{dt} = \mathbf{X}_{c_{j@i}}(t) \cdot Q_{c_{j@i}}(\mathbf{X}(t)), \forall c_{j@i} \in \mathbf{v_i}, \mathbf{v_i} \in \mathcal{V} \tag{1}$$

The initial condition is derived from the one of the Petri Net models. In particular, let $k_0^{c_{j@i}}$ be the initial state of Petri Net model obtained from instance $c_{j@i}$. Then, the corresponding initial condition is:

$$x_k^{c_{j@i}}(0) = \begin{cases} N_{c_{j@i}} \textbf{ if } k = k_0^{c_{j@i}} \\ 0 \textbf{ otherwise} \end{cases} \tag{2}$$

Note that the interconnection of agents in different location and of different classes

is obtained by the dependencies of each infinitesimal generator $Q_{c_{j@i}}(\mathbf{N})$ on the complete state of the model $\mathbf{N}$. More details can be found in [5]. Another interesting formulation is presented in [12]. Interested readers may also check additional readings such as [21], [6], [7] and [11] that present relevant research on different views of the proposed methodology.

4 A case study: Fire emergency and floor evacuation

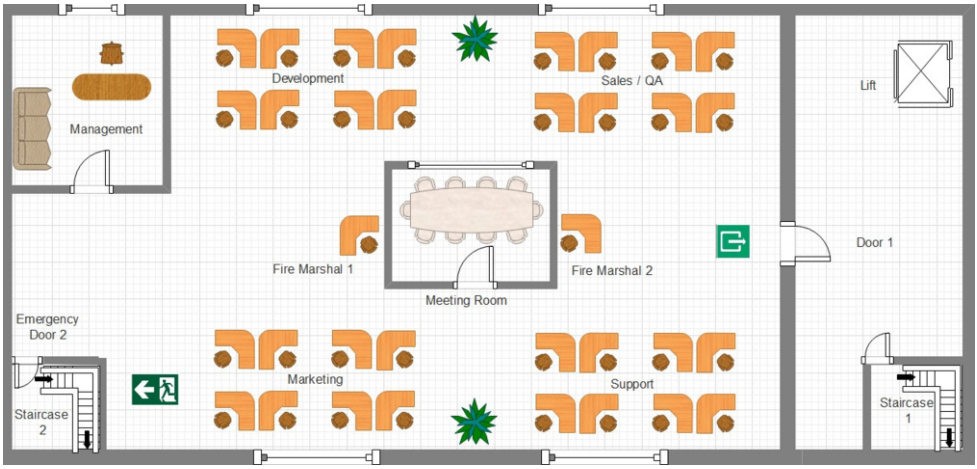

Fig. 4. The office layout

A new, small IT company has recently moved to the first floor of an empty building, and the ground floor hosts a server room. People can access the office (depicted in Fig.4) from the main entrance by using the main $Staircase_1$ and $Door_1$ to the first floor. The office consists of an open space, including four teams of eight people each (*Development*, *Support*, *Marketing* and *Sales & QA*), two additional employees who have been trained to act as *Fire Marshals* in case of fire and one *Manager*. If an emergency occurs, employees can evacuate the floor either by using the main $Door_1$ or the emergency exit $Door_2$, leading to $Staircase_2$ by following the *Fire Marshals* instructions.
They can leave the floor only if all the employees have left. There is an autonomous system that calls the *Fire Brigade* as soon the fire is detected. One or both the doors can be damaged by the fire, preventing the access to the staircases: in this case, the employees who haven't escaped yet and the Fire Marshals are trapped in the building, waiting for the *Fire Brigade*. One day, due to a failure of the service monitoring the servers' heat and because of faulty hardware, a fire develops in the server room and propagates to the first floor.

4.1 The modeled scenario

The office model (OM) has been abstracted as a MAM consisting of the following five classes of Markovian Agents: MA_f modeling the fire developing in the building; MA_d modelling an emergency exit door; MA_e modeling an employee working in

the open space; MA_{fm}, acting as Fire Marshals; and MA_{fb} representing the Fire Brigade. There is one instance of agent MA_f and two instances of agent MA_d to

Fig. 5. Two Markovian Agents classes modeling respectively a fire (MA_f, left) and the doors (MA_d, right)

respectively model the fire evolution and the behavior of the two doors. A fire is ignited and as it expands, it can make unavailable first $Door_1$ at time T_1 (firing of transition T for the first MA_d agent), then $Door_2$ at time T_2 (firing of transition T for the second MA_d agent). At that point, it is not possible anymore to leave the office, as the fire is now spreading everywhere. Employees might have or have not left the open space yet; the only way to extinguish the fire is by the intervention of the fire brigade. The fire enters the room at T_3, with the firing of transition T of agent MA_f. Fig. 5 shows the Petri Net definition of both agents.

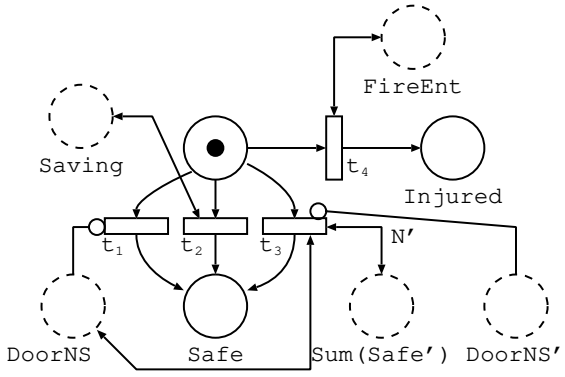

Fig. 6. Markovian Agent modeling an employee (MA_e)

Fig. 6 shows the behavior of an employee, modelled by agent MA_e. When the presence of a fire is detected, MA_e moves to the closest exit as long it is safe (transition t_1). When the place corresponding to the considered door becomes marked, the exits is no longer available (inhibitor arc connecting remote place $DoorNS$ to t_1). In this case, the agent MA_e proceeds to the other exit (transition t_3). Note that the remote exit is used only if the closest one is no longer available (test arc from $DoorNS$ to t_3), the new one is still operational (inhibitor arc from $DoorNS'$ to t_3), and all the other employee who run for that way out have already escaped (test arc from $Sum(Safe')$ to t_3). If both exits are damaged, MA_e is trapped in the office and eventually he/she may be injured (transition t_4 which fires when remote place $FireEnt$ becomes marked). Trapped agents can still be rescued by the *Fire Brigade* (transition t_2 which fires when remote place *Saving* becomes marked).

The role of MA_{fm} is to facilitate the evacuation of the office. Once the fire has been detected, MA_{fm} invites the employees to take the closest exit. If the fire has made the exit unavailable, MA_{fm} will urge the employees to abandon the office

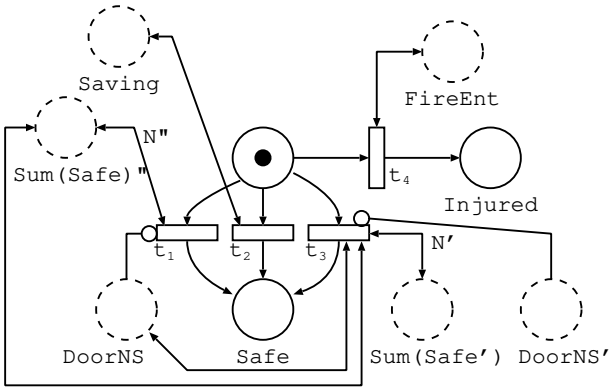

Fig. 7. Markovian Agent modeling a Fire Marshal (MA_{fm})

from the emergency exit. If even the latter is blocked, then MA_{fm} has no way out and is trapped in the office; eventually he/she could be injured. On the other hand, when all the employees (who are not a *Fire Marshal*) have left the area, then MA_e actually abandon the office from the closest exit or the emergency one to go to a safe area. The Petri Net describing the behavior of a fire marshal is shown in Fig. 7. It is very similar to the one shown in Fig. 6 for regular employees, with one addition: remote place $Sum(Safe")$ accounts for the employees in the same location that successfully escaped. This place controls with a test arc both transitions t_1 and t_3 to model the fact that the fire marshal might escape only when all the other employees have successfully left.

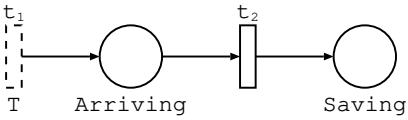

Fig. 8. Markovian Agent modeling a fire brigade (MA_{fb})

The fire brigade model MA_{fb} is shown in Fig. 8. MA_{fb} is available 24x7: when a call from the automatic system has been received (firing of parametric transition t_1), the Fire Brigade team arrives after a time ΔT, and deploys full effort to extinguish the fire and save the life on the workers in the building (transition t_2).
Fig. 9 shows the topology of the considered model. In particular, it connects employee (MA_e) and fire marshals (MA_{fm}), to the doors on the corresponding sides, and to both the fire (MA_f) and fire-brigade (MA_{fb}) agents.

We consider that agents will react to events at a speed λ_r, and that they will be saved by the Fire Brigade at a rate λ_s. Each agent can leave the room at a rate λ_e, but doors allow a maximum exit rate λ_m. If one door is crowded by agents on both sides of the office, due to the other being unavailable, the employees closer to that exit have priority over others, since they would have reached that way out first. Besides the two Fire Marshals, there are respectively N_L and N_R agents in both sides of the office. The scenario is studied against a *Fire Spread Factor* α, that increases and slows down the speed at which fire related events occurs. Parameters used in the model are summarised in Table 1 below.

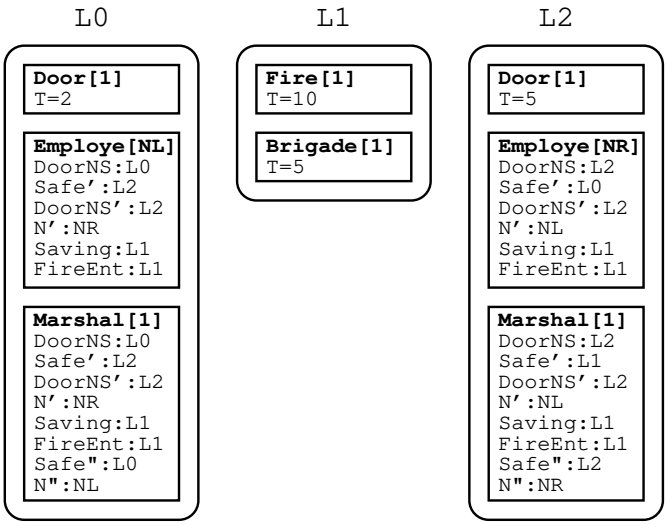

Fig. 9. Topology of the considered scenario

$T_1 = 2\alpha$ min.	$T_2 = 5\alpha$ min.
$T_3 = 10\alpha$ min.	$\Delta T = 3$ min.
$\gamma = 0.4$	
$\lambda_r = 20$ min^{-1}	$\lambda_s = 2$ agent/min.
$\lambda_e = 1$ min^{-1}	$\lambda_m = 3$ agent/min.
$N_L = 17$ agents.	$N_R = 16$ agents.

Table 1
Office scenario parameters

Figures 10 to 15 show the evolution of the various agents. In particular it is interesting to see the way in which the asymmetric behavior penalizes agents in the part of the office where the door becomes unavailable first. It is also interesting to see how the employees (Fig. 12) and the Fire Marshals (Fig. 14) on the left-side, switch to the other exit when the corresponding door becomes unavailable.

Beside the described scenario (addressed in the following as `Scen.1`), two improvements are studied. In the first, the Fire Marshals immediately calls the Fire Brigade, even if that task should have been done automatically (`Scen.2`), allowing a faster intervention. In the second improvement (`Scen.3`), an IoT device can immediately suggest not to use the door on the left, and direct all the agents to the door on the right. Moreover, the combined intervention of the Fire Marshals, can increase the speed at which employees abandon the room of 66%, and delay the fire spreading time of 25%. Fig. 16 to 19 shows the average number of injures for each fire spread factor α, broken down for employees and Fire Marshals starting from the left and right sides of the building, while Fig, 20 accounts for the total injures. Fig. 21 evaluates the fairness of the scenario with an index that is 0 if the process is fair,

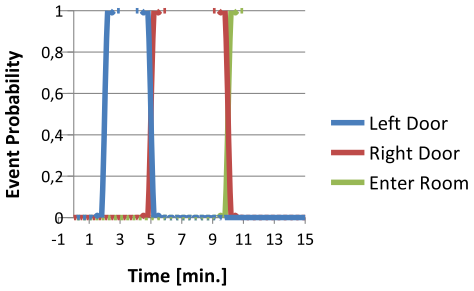

Fig. 10. Fire

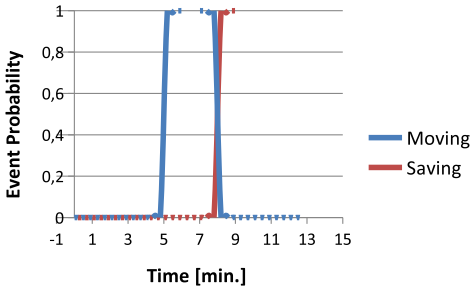

Fig. 11. Fire Brigade

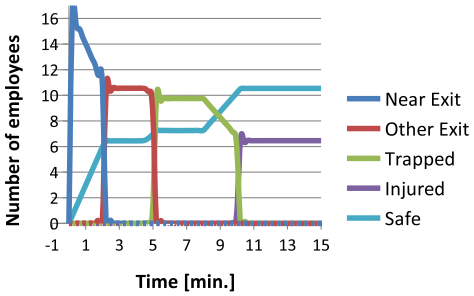

Fig. 12. Employees (L)

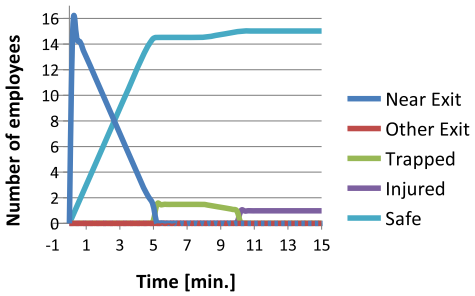

Fig. 13. Employees (R)

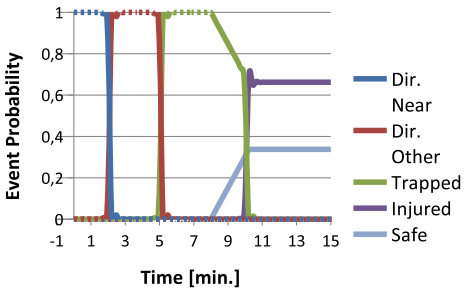

Fig. 14. Fire Marshal (L)

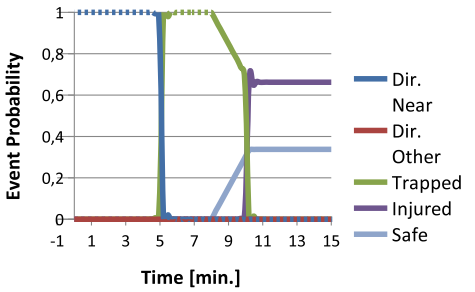

Fig. 15. Fire Marshal (R)

positive if it penalises the employees on the left-side, and negative if it penalises the ones on the right-side. It is defined as:

$$\frac{\text{Injures on the left side}}{\text{Total injures}} - 1$$

As can be seen, calling the fire agent earlier (`Scen.2`) can be very effective when the fire spreads faster, while the cooperation of the Fire Marshal and a better advice that can identify earlier the safest door can be extremely effective when the speed at

that fire spreads is more limited. From a fairness perspective, the first two scenarios are equivalent, while the last one tends to be a little more unfair, since all employees exits from the same door, which is inevitably closer to the people on the right side.

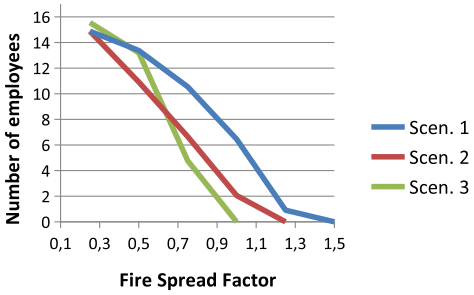

Fig. 16. Employee injuried (L)

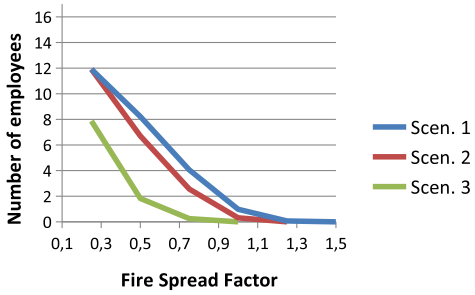

Fig. 17. Employee injuried (R)

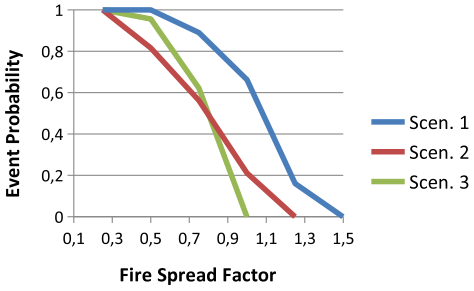

Fig. 18. Fire marshal injuried (L)

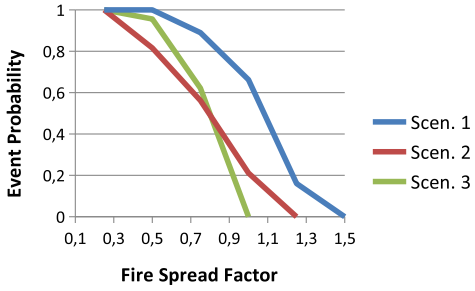

Fig. 19. Fire marshal injuried (R)

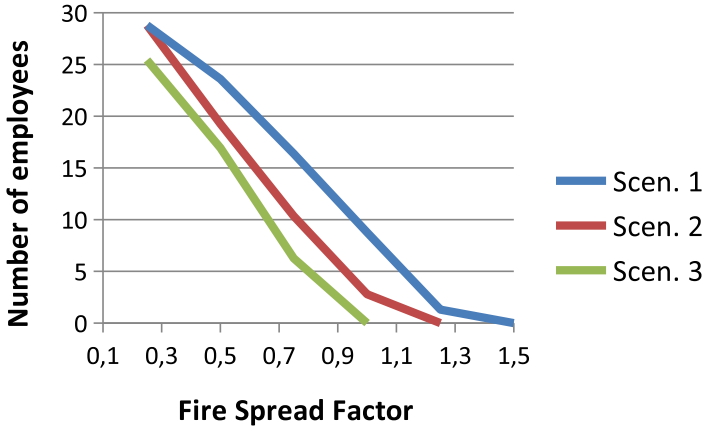

Fig. 20. Total injuries

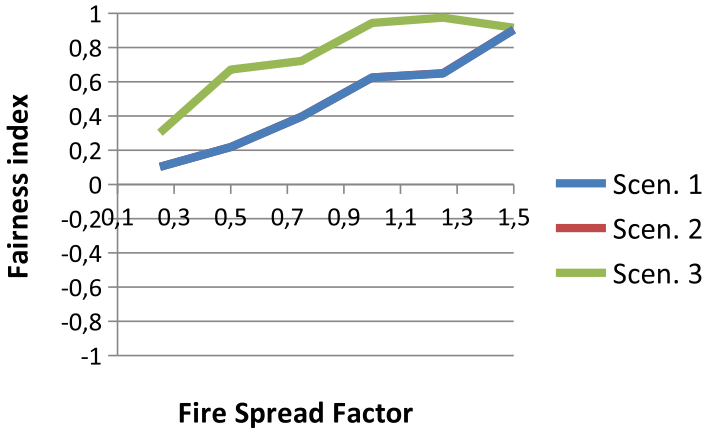

Fig. 21. Left/right fairness

5 Conclusions and Future Work

This paper has presented a Petri Net formulation to describe the behavior of Markovian Agents, and proposed an MA-based approach to analyze the behavior of crowds in presence of panic in closed environments with obstacles. MAs provide a convenient, yet natural tool for this purpose as they have been here applied to two very common cases from the domain and provided coherent results. Future work will be focused both on the formalism definition to further clarify the graphical representation and on the modelling capabilities to codify the analysis procedure. Furthermore, more complex study cases will be taken in account, such as a terrorist threatening a closed environment (e.g., a theater or a discotheque) or a panic situation in an open environment (e.g., a pedestrian area), also including additional parameters that consider proper theories from the domain of psychology.

References

[1] Abu Bakar, N. A., K. Adam, M. Majid and M. Allegra, *A simulation model for crowd evacuation of fire emergency scenario*, 2017.

[2] Almeida, J. E., R. J. F. Rossetti and A. L. Coelho, *Crowd simulation modeling applied to emergency and evacuation simulations using multi-agent systems*, CoRR **abs/1303.4692** (2013).

[3] Bakar, J. A. A., R. C. Mat, A. A. Aziz, N. A. N. Jasri and M. F. Yusoff, *Designing agent-based modeling in dynamic crowd simulation for stressful environment*, Journal of Telecommunication, Electronic and Computer Engineering (JTEC) **8** (2016), pp. 151–156.

[4] Barbierato, E., M. Gribaudo, M. Iacono and A. H. Levis, "Modeling Crowd Behavior in a Theater," Springer International Publishing, Cham, 2018 pp. 49–54.

[5] Bobbio, A., D. Cerotti, M. Gribaudo, M. Iacono and D. Manini, "Markovian Agent Models: A Dynamic Population of Interdependent Markovian Agents," Springer International Publishing, Cham, 2016 pp. 185–203.
URL https://doi.org/10.1007/978-3-319-33786-9_13

[6] Bortolussi, L., R. De Nicola, V. Galpin, S. Gilmore, J. Hillston, D. Latella, M. Loreti and M. Massink, *CARMA: collective adaptive resource-sharing markovian agents*, in: *Proceedings Thirteenth Workshop on Quantitative Aspects of Programming Languages and Systems, QAPL 2015, London, UK, 11th-12th April 2015.*, 2015, pp. 16–31.
URL https://doi.org/10.4204/EPTCS.194.2

[7] Castiglione, A., M. Gribaudo, M. Iacono and F. Palmieri, *Modeling performances of concurrent big data applications*, Software: Practice and Experience **45** (2015), pp. 1127–1144.

[8] Cerotti, D., M. Gribaudo, A. Bobbio, C. T. Calafate and P. Manzoni, *A markovian agent model for fire propagation in outdoor environments*, in: A. Aldini, M. Bernardo, L. Bononi and V. Cortellessa, editors, *Computer Performance Engineering* (2010), pp. 131–146.

[9] Feng, C., M. Gribaudo and J. Hillston, *Performance analysis of collective adaptive behaviour in time and space*, Electronic Notes in Theoretical Computer Science **318** (2015), pp. 53 – 68, twenty-ninth and thirtieth Annual UK Performance Engineering Workshops (UKPEW).

[10] Feng, C. and J. Hillston, "PALOMA: A process algebra for located Markovian agents," Lecture Notes in Computer Science, Springer International Publishing, 2014 pp. 265–280.

[11] Gribaudo, M., M. Iacono and D. Manini, *Three layers network influence on cloud data center performances*, in: *Proceedings - 30th European Conference on Modelling and Simulation, ECMS 2016*, 2016, pp. 621–627.

[12] Guenther, M. and J. Bradley, *Higher moment analysis of a spatial stochastic process algebra*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **6977 LNCS** (2011), pp. 87–101.

[13] Heskestad, C., *Modeling of enclosure fires*, Symposium (International) on Combustion **14** (1973), pp. 1021 – 1030, fourteenth Symposium (International) on Combustion.
URL http://www.sciencedirect.com/science/article/pii/S008207847380092X

[14] Himoto, K. and T. Tanaka, *A physically-based model for urban fire spread*, Fire Safety Science **7** (2003), pp. 129–140.

[15] Ilachinski, A., "Cellular Automata: A Discrete Universe," World Scientific, Singapore, 2001.

[16] Inga, T., *Quantitative analysis of the evacuation system by means of generalized stochastic petri nets.*, Computer Science Journal of Moldova **24** (2016), pp. 184 – 191.

[17] Marsan, M. A., G. Balbo, G. Conte, S. Donatelli and G. Franceschinis, "Modelling with Generalized Stochastic Petri Nets," John Wiley & Sons, Inc., USA, 1994, 1st edition.

[18] Nakajima, Y. and M. Hotta, *A developmental study of cognitive processes in decision making: Information searching as a function of task complexity*, Psychological Reports **64** (1989), pp. 67–79.

[19] Page, S. E. and J. H. Miller, "Complex Adaptive Systems: An Introduction to Computational Models of Social Life (Princeton Studies in Complexity)," Princeton University Press, 2007, kindle edition edition, 284 pp.

[20] Santos, G., B. E. Aguirre, G. Santos, B. E. Aguirre, G. Santos and B. E. Aguirre, *A critical review of emergency evacuation simulation models*, in: *In Proceedings of 30 M. Massink*, pp. 10–11.

[21] Tschaikowski, M. and M. Tribastone, *A partial-differential approximation for spatial stochastic process algebra*, 2014, pp. 74–81.

[22] Wolfram, S., "A New Kind of Science," Wolfram Media Inc., Champaign, Ilinois, US, United States, 2002.