XLVI CILAMCE
VITÓRIA - ES - BRAZIL - 2025
XLVI IBERO-LATIN AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGINEERING

Environment Evasion Modeling in Emergency Situations: Computational Simulator with Reinforcement Learning

Victor Geraldo Gomes[1], Gray Farias Moita[2], Jeferson Figueiredo Chaves[3]

[1] *Postgraduate Program in Mathematical and Computational Modeling*
Av. Amazonas, 7675, Nova Gameleira, Belo Horizonte, Brazil
victor@pontobrsistemas.com.br
[2] *Postgraduate Program in Mathematical and Computational Modeling*
Av. Amazonas, 7675, Nova Gameleira, Belo Horizonte, Brazil
gray@dppg.cefetmg.br
[3] *Postgraduate Program in Mathematical and Computational Modeling*
Av. Amazonas, 7675, Nova Gameleira, Belo Horizonte, Brazil
jeferson@cefetmg.br

Abstract. The evacuation of environments during emergency situations is a crucial task in scenarios such as natural disasters, fires, and urban evacuations. This article presents an innovative approach based on advanced artificial intelligence techniques, with an emphasis on reinforcement learning. This methodology enables agents to learn how to make sequential decisions in uncertain environments by receiving feedback in the form of rewards or penalties. In evacuation contexts, the goal is for agents to autonomously and safely navigate dynamic and potentially hazardous environments. Reinforcement learning offers a versatile framework for training agents capable of avoiding obstacles, identifying safe routes, and maximizing the survival of themselves and others. The study applies reinforcement learning concepts - such as rewards and decision-making under uncertainty - to model emergency evacuations, emphasizing the importance of realistic simulated environments and well-defined reward systems. Case studies and experimental results are presented, demonstrating the effectiveness of the proposed approach in simulated scenarios. These experiments show the agents' ability to overcome complex obstacles, adapt to sudden changes, and find safe evacuation routes in a timely manner. The present work underscores the potential of reinforcement learning as a powerful tool to enhance the safety and efficiency of evacuation operations in critical situations.

Keywords: environment evacuation, intelligent agents, machine learning.

1 Introduction

The accelerated urbanization and increasing complexity of building structures have brought significant challenges to the safe evacuation of people during emergencies, including fires, earthquakes, and terrorist attacks, as highlighted by Helbing et al. [1] and Bellomo et al. [2]. Densely occupied environments require careful planning to reduce risk and ensure efficient evacuation procedures. In this context, computational modeling plays a crucial role in simulating real-world scenarios, forecasting individual and group behavior, and evaluating evacuation strategies, as discussed by Crooks and Heppenstall [3] and Takabatake et al. [4].

Among emerging techniques, Artificial Intelligence - especially Reinforcement Learning (RL) - has proven valuable for modeling agents capable of autonomous decision-making under uncertainty. RL allows agents to adapt their actions over time based on environmental feedback, overcoming the rigidity of rule-based systems, as shown in Sutton and Barto [5] and Fitkau and Hartmann [6].

This work presents a hybrid simulator combining RL and Agent-Based Simulation (ABS) to represent human behavior in emergency evacuations. The simulator accounts for spatial configurations, environmental hazards

(such as fire and obstacles), and social influences. The goal is to generate realistic and adaptive evacuation behavior that supports decision-making in urban safety and architectural design.

2 Review of Existing Approaches

Agent-Based Simulation (ABS) enables detailed modeling of heterogeneous agents and their interactions within dynamic environments. It is particularly effective in replicating crowd behavior and emergent phenomena during evacuation scenarios, as explained by Crooks and Heppenstall [3], Pelechano and Badler [7], and Helbing et al. [8].

Reinforcement Learning, on the other hand, is a learning paradigm in which agents learn to maximize cumulative rewards by interacting with the environment. According to Sutton and Barto [5], an RL problem is defined by a Markov Decision Process (MDP), composed of states, actions, rewards, and policies. One of the most widely used RL algorithms is Q-learning, a model-free approach that uses a Q-table to estimate the value of actions in specific states. It enables agents to learn optimal policies by balancing exploration and exploitation, without requiring a model of the environment.

Studies by Takabatake et al. [4] and Fitkau and Hartmann [6] show that hybrid models integrating RL with ABS can improve decision-making in complex built environments. Additionally, Bellomo et al. [2] and Haghani and Sarvi [11] emphasize how small parameter changes and psychological factors—such as stress, visibility, and perceived risk—influence evacuation behavior. Research by Fridolf et al. [12], Lovreglio et al. [13], and Liu et al. [14] explores factors like smoke, familiarity with exits, and emotional states in shaping individual decisions.

Cognitive and perceptual dimensions also play a key role, as shown by Andresen et al. [15], who introduced cognitive maps into simulations, and by Choi and Chi [16], who used hazard prediction to optimize escape routes. A broader review by Şahin et al. [17] highlights how group behavior and corridor characteristics affect evacuation outcomes.

More recently, Wang et al. [18] explored realistic 3D simulations, while Zhong et al. [9] and Xie et al. [10] proposed data-driven reinforcement learning models for crowd movement. In the robotics field, Tu and Juang [19] applied RL to navigation and obstacle avoidance, reinforcing the applicability of these techniques in dynamic and risky environments.

3 Methodology and Simulator

3.1 Overview of Architecture and Tools Used

The proposed simulator is built using a modular, agent-oriented approach. The system was designed with a focus on flexibility and the ability to integrate artificial intelligence techniques, particularly reinforcement learning. The architecture is based on three main layers: the simulated environment, the evacuation agents, and the decision modules, which interact continuously throughout the simulation cycles.

The simulator was developed in Python, using libraries such as NumPy for matrix operations, Matplotlib for visualization, and custom modules for reinforcement learning. These tools were chosen due to their widespread adoption in the scientific community and their ease of integration with artificial intelligence experiments.

3.2 Representation of the Environment

The environment is modeled as a two-dimensional matrix, where each cell represents elements such as floor, wall, obstacle, or exit. This structure enables the use of discrete techniques like cellular automata, supporting the evaluation of agent movement and space occupancy. Environmental elements are encoded with specific values, allowing agents to perceive nearby conditions such as smoke or obstructions. The Figure 1 illustrates a simplified schematic used solely for the initial validation of the model and algorithmic behavior. In later stages, the simulator incorporates real architectural floor plans and actual environments to assess the model's performance in realistic evacuation scenarios, enhancing its practical applicability.

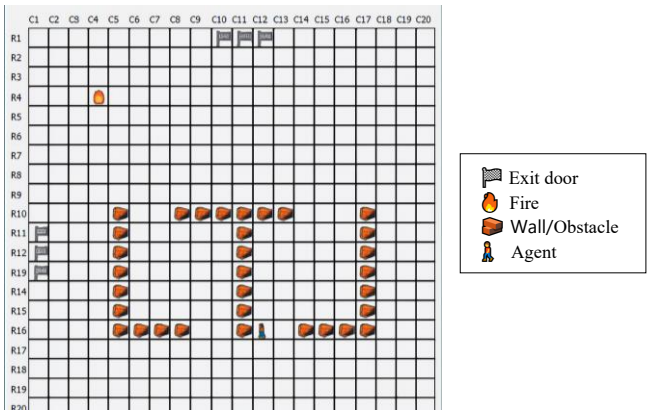


Figure 1 - Simulator Interface

In the proposed simulator, obstacles refer to elements that hinder the agents' movement, represented, in this initial version, only by static walls. These elements define the structure of the environment and directly influence the evacuation strategy learned by the agents through Reinforcement Learning. Algorithm 1 presents this configuration, highlighting how the arrangement of the components impacts navigation in the simulated space.

Algorithm 1- Defining the locations of walls and obstacles

```
Set GRID_WIDTH to 30
Set GRID_HEIGHT to 60
Create empty lists WALLS and EXITS
Define EXIT_ZONES as a list of tuples: (valid_rows, valid_columns, direction)
For each zone in EXIT_ZONES do:
    For each row in zone.valid_rows do:
        For each column in zone.valid_columns do:
            If zone.direction is equal to 'x' then:
                If row ≠ 1 and row ≠ GRID_HEIGHT then:
                    row ← row - 2
                End if
                Add (column, row) to the EXITS list
            Else:  // zone.direction = 'y'
                If column ≠ 1 and column ≠ GRID_WIDTH then:
                    column ← column - 2
                End if
                Add (column, row) to the EXITS list
```

3.3 Agent Modeling

Each agent represents an individual with unique characteristics, such as stress level, movement speed, and spatial perception. These attributes are essential for simulating a variety of behaviors, from rational reactions to chaotic decisions under pressure. Furthermore, agents are equipped with virtual sensors that allow them to perceive the local environment and make decisions based on their learning policy. At the beginning of each simulation, each agent is positioned at a random initial location in the environment (see Algorithm 2). This randomness creates diversity in initial conditions, ensuring that agents face different challenges throughout training, which contributes to more robust and generalized learning.

Algorithm 2- Placing agents randomly in the environment

```
function POSITION_AGENTS_RANDOMLY(cell_matrix, agent_quantity)
    agent_list ← empty list

    for agent_id from 1 to agent_quantity do
        random_position ← SELECT_RANDOM_EMPTY_CELL(cell_matrix)
        new_agent ← CREATE_AGENT(id = agent_id, position = random_position)
        INSERT_AGENT_IN_MATRIX(cell_matrix, random_position)
        add new_agent to agent_list
    end for

    return agent_list
```

In the proposed simulator, agents are autonomous entities that interact with the environment through Reinforcement Learning mechanisms. Each agent makes decisions based on its current state, which encapsulates essential spatial and environmental information. The state of each agent is composed of the following variables:

- Agent's current position: Coordinates in the grid (e.g., x, y);
- Relative position to the goal (exit): Distance and direction;
- Presence of obstacles in adjacent cells (e.g., walls, debris);
- Presence of fire or smoke in adjacent cells;
- Visited cells history: Used to avoid backtracking and promote exploration;
- Reward signal associated with the current cell (e.g., penalized if close to fire or dead ends).

3.4 Reinforcement Learning

The methodology employed integrates reinforcement learning algorithms to define each agent's evacuation strategy. Q-learning served as the foundational algorithm, allowing agents to iteratively learn which action maximizes their accumulated reward toward efficient evacuation. The reward considers multiple factors: proximity to the exit, avoiding crowds, staying away from smoky areas, and minimizing total evacuation time.

Reinforcement Learning is conducted using Equation 1 to calculate rewards R(s, a) based on the actions performed by the agents. This equation takes into account several factors for calculating the reward, including proximity to the exit (Δ_{exit}), distance from the fire (Δ_{fire}), an additional penalty for agents that approach the flames excessively (P_{fire}), and a fixed cost associated with each movement performed (*w4*). This reward structure seeks to incentivize behaviors that increase evacuation efficiency while discouraging trajectories that put the agent at risk.

The reward assigned to an agent depends on the context of the movement performed:

1. Evacuation Reward: When the agent reaches an exit, he receives a fixed reward of 100 points (upper limit, maximum score), directly encouraging the main objective of the simulation: evacuation of the environment.
2. General Reward for Other Situations: When the agent does not reach the exit, the reward is calculated using the following formula:

$$R(s, a) = + w1 \cdot \Delta_{exit} + w2 \cdot \Delta_{fire} - w3 \cdot P_{fire} - w4 \quad (1)$$

The coefficients *w1*, *w2*, *w3*, and *w4* used in Equation 1, which can be understood here as weights, were defined in this initial modeling with the values below. These values can be changed to calibrate the agents' behavior according to the simulation objectives. The coefficients defined in the simulations of this work are:

w1 = 10: Importance of approaching the exit.
w2 = 5: Importance of staying away from fire.
w3 = 20: Penalty for proximity to fire.
w4 = 1: Fixed penalty for actions, including movement or inaction.

The parameters *w1* and *w2* are positive, encouraging desirable actions, while *w3* and *w4* are negative, representing penalties. The weights $w1$, $w2$, $w3$ and *w4*, which compose the reward function, were initially defined arbitrarily and subsequently adjusted through experimentation to improve the learning performance of the simulator.

The attribute Δ_{fire} (Equation 1) represents the variation in the Manhattan distance between the agent and the fire, before and after executing an action. Manhattan distance is a metric that calculates the distance between two points by summing the absolute differences of their coordinates, moving only in horizontal and vertical directions - like navigating a city grid. This attribute is used to encourage behaviors that maximize the agent's safety during the evacuation.

And finally Equation 2 defines the penalty P_{fire}, applied to agents based on their immediate proximity to the fire source. It is an inversely proportional function of the distance between the agent and the fire, expressed as:

$$\mathrm{P_{fire}} = \frac{1}{\text{distance from fire}+1} \tag{2}$$

The closer the agent is to the fire, the greater the penalty applied to its reward. The "+1" in the denominator prevents division by zero and softens the penalty at short distances, discouraging risky movement and reinforcing self-preservation.

Throughout the simulations, agents explore the environment by taking actions (e.g., move up, down, left, or right) and receiving feedback through rewards. Initially acting randomly, they gradually learn to associate certain actions with better outcomes. Repeated interactions lead to an optimal policy, enabling agents to reach exits more quickly while avoiding danger and congestion.

The data structures that support the learning process include:

- Q-table: a dictionary or matrix structure storing Q-values for each state-action pair;
- State representation: a tuple encoding agent position, distance to exits, nearby obstacles, and visited cells;
- Action set: a finite set of movements (e.g., up, down, left, right, stand still);
- Visited cells list: a memory of previously explored positions to avoid backtracking.

This combination of algorithm and data structures enables the agents to learn adaptive and safe evacuation policies tailored to the complexity of the environment.

4 Results and discussions

The tests performed in this phase were primarily aimed at evaluating the basic functionality of the developed simulator, focusing on agent movement, evacuation time, and environmental adaptation. These tests also aimed to validate the reinforcement learning logic incorporated into the agents' behavior, as well as identify potential improvements to the model. Efficiency in optimizing evacuation routes.

For the initial training of the agents and the application of Reinforcement Learning, a custom computer simulator, developed specifically to reproduce emergency scenarios in a controlled environment, was used. The simulator was run 150 times, and in each run, 10 agents participated in the simulation, resulting in a total of 1,500 agent-environment interactions. This approach allowed the generation of a significant volume of data, essential for fine-tuning the learning models and validating the agents' emergent behaviors during the evacuation.

The simulation was configured with 10 exit doors strategically distributed throughout environment's length, representing multiple evacuation options. These doors were positioned to ensure coverage of different regions of the space and to introduce varied decision-making scenarios for agents. The doors, in the Figure 2 below, are represented by checkered flags. The graph in Figure 2 illustrates the evolution of agent learning throughout the simulations for the 10-agent scenario. In it, the metrics "Min Steps " and "Max Steps " highlight, respectively, the minimum number of steps the most efficient agent needed to evacuate the environment and the maximum number of steps required for the slowest agent to exit.

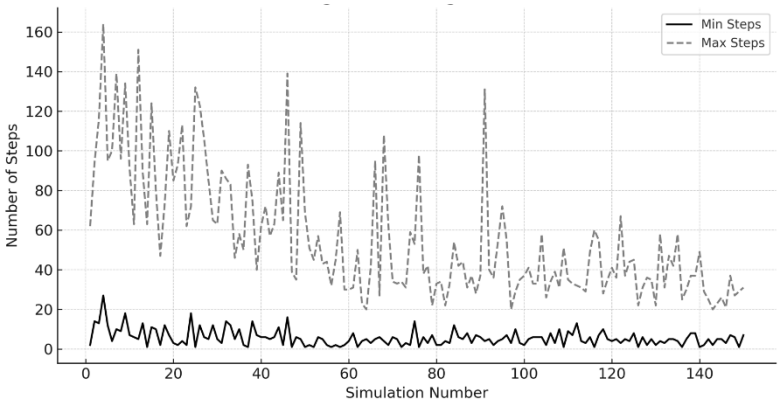


Figure 2 – Learning evolution to the throughout the simulations

Initially, the simulator was evaluated using a baseline scenario with fixed parameters and a limited number of agents. As the model matured, further simulations were conducted to analyze the impact of population growth on total evacuation time. Four main scenarios were defined, varying the number of agents (10, 30, 50, and 100 simulated individuals). Each scenario was run multiple times under similar conditions, and the results indicate that evacuation time increases with the number of officers, but not always linearly. This highlights the influence of factors such as congestion, competition for exits, and fire spread in the environment.

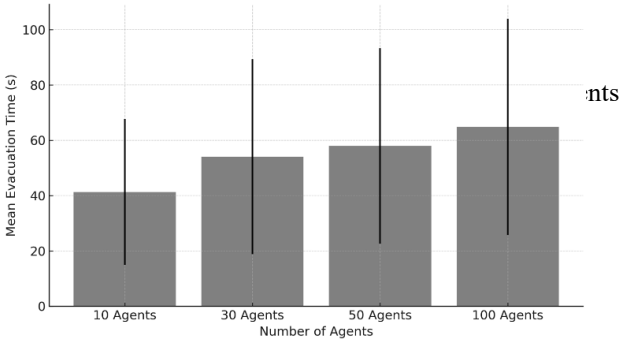
Table 1 and Figure 3 below present the descriptive statistics for the total evacuation time in each of the tested scenarios.

Table 1 - Statistics by number of agents

Number of Agents	Mean Evacuation Time (s)	Standard Deviation (s)	Minimum Time (s)	Maximum Time (s)
10 Agents	41.32	26.35	13	194
30 Agents	54.07	35.24	20	216
50 Agents	57.97	35.32	22	244
100 Agents	64.83	39.07	30	278

5 Conclusions

This work presented the development and validation of a computational simulator for emergency evacuation scenarios, integrating Reinforcement Learning (RL) with ABS. The system enabled agents to autonomously learn and optimize evacuation routes through repeated interaction with the environment, promoting more adaptive and realistic decision-making in critical situations.



Simulations showed that agents improved their performance over time, reducing evacuation duration and adjusting their behavior in response to obstacles, layout, and crowd density. Results also indicated that the number of agents has a nonlinear influence on evacuation time, underscoring the impact of congestion and spatial complexity on overall efficiency.

This study reinforces the potential of combining RL and ABS in complex architectural settings, offering both theoretical insight and practical value for safety planning and design. Future work may explore dynamic fire spread, moving obstacles, and heterogeneous populations. The simulator offers a flexible tool to support safer, data-driven decisions in real-world emergency contexts.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors or has the permission of the owners to be included here.

References

[1] HELBING, D. et al. Simulating dynamical features of escape panic. Nature, v. 407, n. 6803, p. 487–490, 2000.
[2] BELLOMO, Nicola et al. Human behaviours in evacuation crowd dynamics: From modelling to “big data” toward crisis management. Physics of Life Reviews, v. 18, p. 1–21, 2016.
[3] CROOKS, Andrew T.; HEPPENSTALL, Alison J. Introduction to agent-based modelling. In: Agent-based models of geographical systems. Dordrecht: Springer Netherlands, 2011. p. 85–105.
[4] TAKABATAKE, Tomoyuki et al. Optimizing evacuation paths using agent-based evacuation simulations and reinforcement learning. International Journal of Disaster Risk Reduction, v. 117, p. 105173, 2025.
[5] SUTTON, R. S.; BARTO, A. G. Reinforcement learning: An introduction. 2. ed. Cambridge: MIT Press, 2018.
[6] FITKAU, Isabelle; HARTMANN, Timo. Applying multi-agent reinforcement learning for escape path planning in a fire evacuation environment. 2025.
[7] PELECHANO, Nuria; BADLER, Norman I. Modeling crowd and trained leader behavior during building evacuation. IEEE Computer Graphics and Applications, v. 26, n. 6, p. 80–86, 2006.
[8] HELBING, D.; FARKAS, I. J.; VICSEK, T. Simulating dynamical features of escape panic. Nature, v. 407, n. 6803, p. 487–490, 2000.
[9] ZHONG, J. et al. Crowd simulation in the metaverse: a data-driven perspective. Computational Visual Media, v. 8, p. 143–160, 2022.
[10] XIE, D. et al. Data-driven simulation and optimization framework for building evacuation using deep reinforcement learning. Expert Systems with Applications, v. 201, p. 117066, 2022.
[11] HAGHANI, M.; SARVI, M. Stated and revealed crowd safety concerns: A review of the literature. Journal of Advanced Transportation, v. 2017, p. 1–20, 2017.
[12] FRIDOLF, K. et al. Movement speed and exit choice in smoke-filled rail tunnels. Fire Safety Journal, v. 59, p. 8–21, 2013.
[13] LOVREGLIO, R.; FONZONE, A.; DELL’OLIO, L. A mixed logit model for predicting exit choice during building evacuations. Transportation Research Part A: Policy and Practice, v. 92, p. 59–75, 2016.
[14] LIU, Hui et al. A fuzzy cellular automata model for crowd evacuation considering emotion. Physica A: Statistical Mechanics and its Applications, v. 545, p. 123682, 2020.
[15] ANDRESEN, E.; CHRAIBI, M.; SEYFRIED, A. A representation of partial spatial knowledge: a cognitive map approach for evacuation simulations. Transportmetrica A: Transport Science, v. 14, n. 5–6, p. 433–467, 2018.
[16] CHOI, M.; CHI, S. Optimal route selection model for fire evacuations based on hazard prediction data. Simulation Modelling Practice and Theory, v. 94, p. 321–333, 2019.
[17] ŞAHIN, G.; ROKNE, J.; ALHAJJ, R. Modelling pedestrian movement and crowd dynamics: A review. Physica A: Statistical Mechanics and its Applications, v. 535, p. 122370, 2019.
[18] WANG, Xiang et al. Modeling boundedly rational route choice in crowd evacuation processes. Safety Science, v. 147, p. 105590, 2022.
[19] TU, F.; JUANG, J. C. Learning to avoid obstacles using reinforcement learning algorithms. Sensors, v. 23, n. 1, p. 279, 2023.