Predictive Agent-Based Crowd Model Design Using Decentralized Control Systems

Cristian Berceanu, Ionuţ Banu, Bettina S. Husebo, and Monica Patrascu, *Senior Member, IEEE*

***Abstract*—As a complex system, crowd dynamics emerge bottom-up from the local interactions between pedestrians as component subsystems. This article proposes a predictive agent-based crowd simulation model to analyze the outcomes of emergency evacuation scenarios taking into account collisions between pedestrians, smoke, fire sprinklers, and exit indicators. The crowd model is based on a decentralized control system structure, where each pedestrian agent is governed through a deliberative-reactive control architecture. The simulation model for evacuation includes a routing-based control system for dynamic-guided evacuation. A design case illustrates the modeling process. Results show that the crowd simulation model based on agent autonomy and local interactions is able to generate higher level crowd dynamics through emergence.**

***Index Terms*—Agent-based modeling, complex systems, decentralized control, emergence, human behavior modeling, intelligent agents, simulation model.**

I. Introduction

RECENT developments in large city infrastructure have led to urban crowding. Emergency evacuation of buildings in case of fire became a challenging problem as it demands a short evacuation duration to ensure the safety of people's lives. Emergency evacuation of buildings requires fast and effective decision-making to be successful [1], while an adequate number of exits can help mitigate safety risks because smoke and crowded exit routes reduce pedestrian moving speed and may create panic and confusion.

The aim of this article is to design a predictive agent-based crowd simulation model to analyze the outcomes of emergency evacuation scenarios taking into account collisions between pedestrians, smoke, fire sprinklers, and exit indicators.

Understanding specific behavioral response under extreme situations can significantly affect an evacuation [2]. Agent-based models are a good candidate for crowd behavior forecasts because pedestrians can be simulated by individual agents [3]. Crowd behavior visualization is a complex multiagent modeling problem [4] and various techniques for crowd simulation have been proposed in the past 20 years [5].

There are three main categories of pedestrian simulation models based on the levels of detail [6]: Microscopic, mesoscopic, and macroscopic. Macroscopic models might be computationally efficient, but they rely on global knowledge, which is not available for the pedestrian evacuation problem [7]. Microscopic models consist of a large number of interacting agents with specific behaviors [8], [9]. Mesoscopic models are a reasonable compromise between detailed microscopic models and computationally efficient macroscopic models [6], [10]. In this article, we aim to track the movement of people, hence we propose a microscopic modeling approach.

The dynamics of a crowd [11] cannot simply be limited to the principles of mechanics and deterministic causality [12]. In fact, the heterogeneous behaviors of individuals and their social dynamics can have an important influence on the crowd dynamics and, in particular, on the strategy they use to reach a certain goal. As a *complex system*, the dynamics of a crowd emerge from local interactions between component systems (i.e., the human participants). Evacuation dynamics show special stress conditions, which propagate along the crowd through mechanical (with the environment, e.g., running into walls) and social interactions (e.g., observing the agitation of another person). Current approaches usually [7], [13], [14] reduce the individuals partaking in the crowd to particles without dynamics or decision-making capabilities. However, in a real-life situation, each participant makes individual decisions that are influenced by the environment, the other crowd participants, their own psychological and physiological responses, etc. This behavior requires modeling each participant with a certain level of autonomy and agency. To obtain this, we base our approach on agents and decentralized control principles [15]. In decentralized configurations [16], each subsystem (i.e., control loop) is sensitive and reacts only to local factors, either from the environment, or from other subsystems. The global behavior is obtained through local interactions or information in a given vicinity, reducing necessary resources in terms of communication and computation needs [17]. Individualizing the decision-making process for each agent, i.e., assigning personal attributes [18] observable in their behavior, brings the crowd model closer to the real-world structure: composed of autonomous systems with agency.

We employed this approach before in a preliminary study focused on building management [19], agent-based modeling (ABM) [20] applied to traffic control, and microscopic modeling of traffic participants [21], with good results.

The opportunity of modeling decentralized complex interactions using multiagent systems aligns with the concept of an agent as a *system with agency* [22]. Thus, we propose a crowd

Manuscript received 18 November 2021; revised 19 March 2022 and 14 May 2022; accepted 27 June 2022. Date of publication 22 July 2022; date of current version 24 February 2023. *(Corresponding author: Monica Patrascu.)*

Cristian Berceanu and Ionuţ Banu are with the Complex Systems Laboratory, University Politehnica of Bucharest, 060042 Bucureşti, Romania (e-mail: cristian.berceanu@stud.acs.upb.ro; ionutbanu094@gmail.com).

Bettina S. Husebo is with the Centre for Elderly and Nursing Home Medicine, University of Bergen, 5007 Bergen, Norway (e-mail: bettina.husebo@uib.no).

Monica Patrascu is with the Complex Systems Laboratory, University Politehnica of Bucharest, 060042 Bucureşti, Romania, and also with the Centre for Elderly and Nursing Home Medicine, University of Bergen, 5007 Bergen, Norway (e-mail: monica.patrascu@uib.no).

This article has supplementary downloadable material available at https://doi.org/10.1109/JSYST.2022.3188339, provided by the authors.

Digital Object Identifier 10.1109/JSYST.2022.3188339

modeling method as the result of three combined concepts: Microscopic crowd models, decentralized control, and agent-based simulation. The main contributions of the article are threefold: 1) crowd modeling based on agent autonomy and local interactions to create higher level crowd dynamics through emergence; 2) a model of crowd participants using a deliberative-reactive agent architecture; and 3) introducing a routing-based control system for dynamic guided evacuation.

Article organization. Section II outlines the concept of the decentralized agent-based model for crowds during evacuation. Sections III and IV detail a design scenario for a particular floor plan and fire evacuation, for both the environment and the human agents. Section V shows simulation model results and discussion. Finally, Section VI concludes this article.

II. Decentralized Agent-Based Model

A. Concept

The aim of the model is to ascertain the behavior of the crowd during evacuation as it emerges from the individual autonomous behaviors of the entities comprising the crowd. The approach we take is a bottom-up design using agents to model each person and using the principles of decentralized control systems (DCSs) to generate high-level behaviors of the crowd based on local interactions. The modeling concept we present here generates an agent-based simulation model (ABSM).

In this article, we design a predictive ABSM: Known agents are placed in a known environment and allowed to interact with the purpose of analysing their emergent behavior. As opposed to the illustrative case (the demonstrative experiment), in a predictive model, the ABSM overall outcome patterns are not known *a priori* (the generative experiment) [23].

The ABSM elementary agent is a persistent entity characterized by at least one definable state [24]. Through interaction, this entity can cause changes in the environment or in the states of other agents. The model itself is the set of agents, their interaction rules, and their environment.

Consider a single pedestrian agent with state-space dynamics described by $x_{k+1} = f(k, x_k, w_k)$ and $y_k = g(k, x_k, w_k)$, where x is the vector describing the complete internal state of the agent, w is the vector of environment inputs, y is the vector of output variables, while f and g are update laws, all dependent on discrete time step k. In a group, the agent interacts with other agents. Let r be the set of local interaction rules in a radius Δ. Agent i in a network of n agents is

$$\begin{cases} x^i_{k+1} = f^i\left(k, x^i_k, w^i_k, r^i\left(k, x^i_k, w^i_k, \{y^{i-\Delta}_k, \ldots, y^{i+\Delta}_k\}\right)\right) \\ y^i_k = g^i\left(k, x^i_k, w^i_k\right) \end{cases} \tag{1}$$

where the set $\left\{y^{i-\Delta}, \ldots, y^{i+\Delta}\right\}$ contains the actions (outputs) of other agents in radius Δ affecting agent i as inputs. The dimensions of Δ are dependent on the agent sensing elements. Direct interactions happen inside the areas delimited by Δ, while indirect interactions propagate via overlapping vicinities.

The problem of constructing an agent-based model resumes, besides modeling the individual dynamics f^i and g^i, to designing the set of local interactions r^i so that patterns emerge, observable through the outputs of the agents. Rules are often at least partly known *a priori* in illustrative models, whereas this article deals with a type of predictive ABM without immediately obvious interaction outcomes. In the case of systems with simple interactions, e.g., in flocking or herding, the rules r^i are the same for all agents and easy to design, but not suitable for humans [5], who possess reasoning capabilities. Panicked people make decisions based on uncontrolled personal interests, social, and cultural constraints, leading to social disfigurement (e.g., following the decisions of others). In an agent network, collective panic is an emergent behavior resulting from the individual internal decision-making process and the interaction rules in a certain locally observable vicinity.

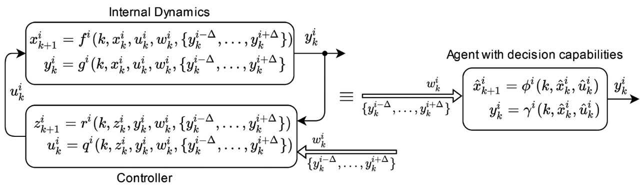
Internal Dynamics
Controller
Agent with decision capabilities
$x^i_{k+1} = f^i(k, x^i_k, u^i_k, w^i_k, \{y^{i-\Delta}_k, \ldots, y^{i+\Delta}_k\})$
$y^i_k = g^i(k, x^i_k, u^i_k, w^i_k, \{y^{i-\Delta}_k, \ldots, y^{i+\Delta}_k\}$
$z^i_{k+1} = r^i(k, z^i_k, y^i_k, w^i_k, \{y^{i-\Delta}_k, \ldots, y^{i+\Delta}_k\})$
$u^i_k = q^i(k, z^i_k, y^i_k, w^i_k, \{y^{i-\Delta}_k, \ldots, y^{i+\Delta}_k\})$
$\hat{x}^i_{k+1} = \phi^i(k, \hat{x}^i_k, \hat{u}^i_k)$
$y^i_k = \gamma^i(k, \hat{x}^i_k, \hat{u}^i_k)$
y^i_k
u^i_k
w^i_k
$\{y^{i-\Delta}_k, \ldots, y^{i+\Delta}_k\}$
$\equiv$

Fig. 1. Agent overview.

Thus, we propose equipping each agent (Fig. 1) with its own controller. Instead of attempting to model the interaction rules from the perspective of the observer, we model the internal decision-making of the agent via the controller. This approach sustains the agency of subsystems as crowd participants and allows their interactions to emerge organically from the local autonomous decisions of each individual agent. The feedback structure allows the agent to generate continuous decisions based on changes in its vicinity of radius Δ, either environmental w^i, or from other agents $\{y^{i-\Delta}, \ldots, y^{i+\Delta}\}$.

For an agent described by the internal dynamics

$$\begin{cases} x^i_{k+1} = f^i\left(k, x^i_k, u^i_k, w^i_k, \{y^{i-\Delta}_k, \ldots, y^{i+\Delta}_k\}\right) \\ y^i_k = g^i\left(k, x^i_k, u^i_k, w^i_k, \{y^{i-\Delta}_k, \ldots, y^{i+\Delta}_k\}\right) \end{cases} \tag{2}$$

let the controller for each agent with state z^i, inputs y^i, w^i, $\left\{y^{i-\Delta}, \ldots, y^{i+\Delta}\right\}$, and output u^i, be described by

$$\begin{cases} z^i_{k+1} = r^i\left(k, z^i_k, y^i_k, w^i_k, \{y^{i-\Delta}_k, \ldots, y^{i+\Delta}_k\}\right) \\ u^i_k = q^i\left(k, z^i_k, y^i_k, w^i_k, \{y^{i-\Delta}_k, \ldots, y^{i+\Delta}_k\}\right) \end{cases} \tag{3}$$

then, the closed-loop agent with controller becomes

$$\begin{cases} \hat{x}^i_{k+1} = \phi^i\left(k, \hat{x}^i_k, \hat{u}^i_k\right) \\ y^i_k = \gamma^i\left(k, \hat{x}^i_k, \hat{u}^i_k\right) \end{cases} \tag{4}$$

where $\hat{x}^i = \left[x^i \; z^i\right]^T$ and $\hat{u}^i = \left[w^i \; y^{i-\Delta} \; \ldots \; y^{i+\Delta} \; u^i\right]^T$. The agent behavior is described by nonlinear functions ϕ^i and γ^i.

In the case of emergency evacuation, the environmental inputs w^i model the presence of obstacles, for instance, walls or smoke inhalation, and provide signals from the evacuation systems, such as alarms and indicators. The output y^i reflects the trajectory of the agent through the environment. Issues of stability are not formally relevant here, as the simulation model must reveal these situations as the crowd develops and moves during evacuation. In terms of observable actions, the agent trajectories are graphically displayed for analysis.

The network of agents with controllers is what is known as a DCS. In this particular case, the DCS is noncooperative, meaning that each agent has its objective locally and independently

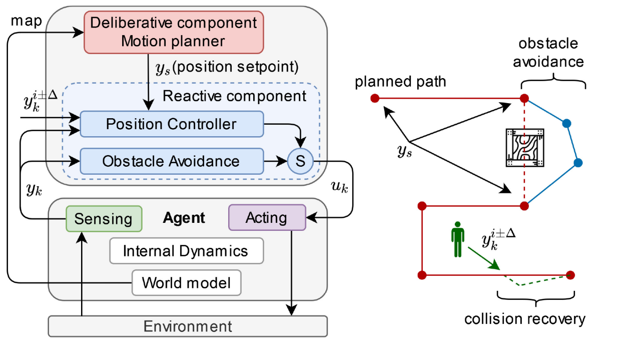

Fig. 2. Deliberative-reactive motion control architecture for the pedestrian agent.

defined. For an individual in a crowd, this objective is to reach an exit as fast as possible.

B. Pedestrian Agent Controller Structure

For humans moving through a physical space, a deliberative-reactive structure [25] for the controller is suitable. The reactive component deals with path following through the building space, including collision avoidance [26], while the deliberative component performs motion planning (calculates point-based trajectories) and responds to environmental stimuli such as smoke and alarms. Fig. 2 shows the structure of the motion control architecture. Here, the movement between two points on the computed trajectory is adjusted through the subsumption module S to locally avoid unexpected obstacles. Other behaviors are modeled similarly; for instance, pushing through a crowd adds the generated collisions with other agents, which require recovery to the trajectory.

C. Scenario and Technologies

In what follows, we perform the design of an ABSM for crowd evacuation inside a building in case of fire using the described concept. For the fire suppression and alarm system, and for evacuation guidance, we design the so-called static agents [19] in closed loop, which, from the point-of-view of the crowd, are part of the environment.

All agents are implemented in JADE (JAVA Agent Development Framework) [27], an agent-oriented programming software platform. Interagent communication is message-based, using FIPA specifications [27]. For ABSM visualization, we use jMonkeyEngine [28], an open-source 3-D graphic engine.

III. Modeling the Environment

A. Environment Structure and Floor Plan

The environment in this scenario is three-dimensional. The positions of static (environmental artifacts) and mobile agents (humans) are tracked in a three-coordinate system XYZ, relative to the origin point $O = (0, 0, 0)$.

The floor plan includes different spaces: 14 work rooms with areas ranging from 14 to 33 m^2, a conference room of 55 m^2, two restrooms, two storage (one houses electrical equipment), and a reception of 33 m^2. These are connected through four halls with lengths between 13 and 18 m. The plan has two exits: One

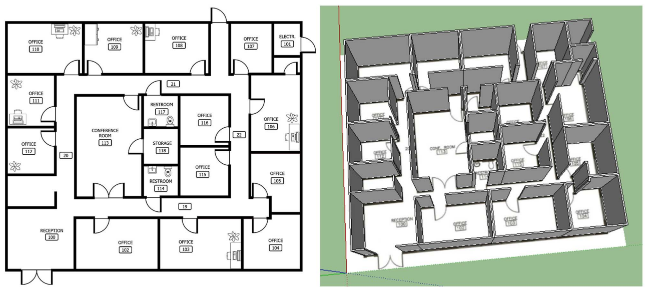

Fig. 3. Environment structure: The floor plan [29].

through the reception hall, and one situated in the diagonally opposite corner of the plan [29]. Fig. 3 shows the Sketchup [30] 3-D model of the floor plan.

In the simulated environment, each room is an object defined by a set of variables: An identification number, four points representing the spatial coordinates of the four corners, the presence or absence of fire, the smoke amount and increase rate, the states of the alarms and fire extinguishers. When generating the room, alarms are placed in the center of the room, while fire origins are randomly situated.

The protection control system is part of the environment. Its purpose is to respond to fire events with two controllers: One for fire suppression and one for evacuation guidance. In buildings, this system works alongside HVAC and other internal structural control systems. Compliance with safety standards dictates the design criteria [31], taking into account the time until the level of smoke affects visibility for people; critical times for the active protection system, such as fire detection time, time required to activate the fire suppression system, time required to extinguish the fire; duration of occupant evacuation; and heat release rate.

We model the protection system components as static agents (with fixed positions within the environment) in closed loops composed of sensor, actuator, and controller agents, forming cooperative structures with control objectives [32].

B. Smoke Model

Several factors such as smoke spread and temperature influence the evacuation process. The speed of people walking [33] or the choice of exit are affected by the presence of smoke. The smoke mass conservation equation based on the McCaffrey correction [34] and adjusted for fire suppression is

$$\dot{M} = \dot{m}_s - \rho_s M \tag{5}$$

with initial condition $M(0) = 0$, where M is the smoke in the room, and $\dot{m}_s$ is the smoke generation rate. The smoke density is $\rho_s = \rho_a(T_a/T_s)$, where T_a is the ambient temperature, T_s is the smoke temperature, and ρ_a is the ambient air density.

For a continuous region fire, experimental data [34] shows that $\dot{m}_s/Q = 0.011(Z/Q^{\frac{2}{5}})^{0.566}$ for $0 \leq Z/Q^{\frac{2}{5}} < 0.08$, where Z is the height of the smoke layer, and Q is the heat release rate [35]: $Q = \alpha\tau^2$ (τ time). Coefficient α depends on the nature of the material feeding the fire, with hysteretic behavior for increasing and decreasing fire intensity [36].

Noting $k_1 = 0.011(Z/\dot{Q}^{\frac{2}{5}})^{0.566}$ and $\rho_s = k_2$, (5) becomes $\dot{M} = k_1 Q - k_2 M$. For discrete-time simulation, a zero-order hold element is necessary, and the discrete form of the smoke

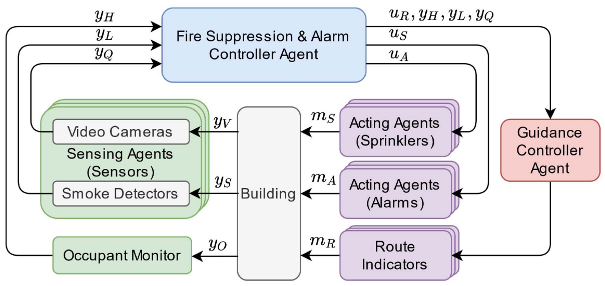

Fig. 4. Protection control system structure.

Algorithm 1: Sensor Operation.

```
1 if fire exists then
2     send location to controller;
3     while smoke amount not zero do
4         send smoke amount to controller;
5     end
6     report successful fire suppression;
7 end
```

spread for simulation is

$$M_k = q_0 Q_k + q_1 Q_{k-1} + m_1 M_{k-1}. \tag{6}$$

Coefficients q_0, q_1, and m_1 are updated based on k_1, k_2, and the sampling time.

C. Protection Control System

The aim of the fire suppression and alarm control system is to reduce damage caused by fire, while the guidance controller aids the evacuation of occupants. A well-designed fire suppression system has sprinklers with high efficiency, especially in the early stages [37]. The control system components are sensing agents, acting agents, and two controller agents. Fig. 4 shows the control system structure, where y_S contains air parameters, y_V the image of the room, and y_O human occupants; y_Q is the measured smoke quantity, y_L is the fire location, and y_H is the number of occupants; u_A is the alarm command, u_S is the sprinkler command, and u_R is the routing computation trigger command; m_A is the result of alarm execution (sound or light); m_S is the result of sprinkler activation; and m_R is the result of route indicator activation.

Sensing agents emulate the functioning of sensors. Their role is to ascertain the presence of fire and the amount of smoke, and to convert this environmental data into information for the controller. Conventional smoke detectors, such as ionization and optical detectors, might not respond efficiently for safe evacuation [38] due to the time it takes for particles to reach the sensitive elements. Video detectors are a fairly recent technique that exploits the visual features of fire, such as color, texture, geometry, pulsation, and movement [38]. The sensing agents in this article are composed of video detectors for fire location and smoke detectors for smoke amount.

Sensing agent operation is illustrated in Algorithm 1. Each sensor is identified by an identification number (IDN) corresponding to their room. Sensor operation is twofold: A triggered-based behavior that transmits the fire location (in the XYZ coordinate system) only in case of fire detection (to avoid overloading the controller communication queue), and a continuous behavior that transmits smoke amount information for as long as smoke is detected in the room.

Algorithm 2: Fire Suppression and Alarm Controller Operation (Auto Mode).

```
1  if fire exists then
2      retrieve fire location and room IDN;
3  end
4  if smoke quantity $>$ threshold value then
5      activate alarms;
6      activate sprinkler in room IDN;
7      activate guidance controller;
8      send fire location to guidance controller;
9  end
10 if building is evacuated and smoke amount is zero then
11     deactivate alarms, sprinklers, guidance controller;
12 end
```

Algorithm 3: Actuator Operation.

```
1 if activation is requested and actuator is off then
2     activate actuator;
3 end
4 if deactivation is requested and actuator is on then
5     deactivate actuator;
6 end
```

The occupant monitor is an environment artifact and acts as a sensing unit counting the occupants of each room.

The fire suppression and alarm controller agent monitors changes in sensor information and computes commands for the acting agents. The controller objective is to maintain the state of the environment as *without fire/smoke*. The controller has two operation modes: Manual (open loop) and auto (closed loop). In manual mode, commands are set by a human operator through the input interface, for diagnostic procedures. In auto mode, the commands are generated such that fire suppression and alarm activation happen in the shortest time possible.

Auto-mode operation is illustrated in Algorithm 2. Fire presence triggers the fire location retrieval from the sensor IDN. Information on smoke amount is continuously received from the fire-reporting sensors, based on which alarm and sprinkler commands are triggered. Both types are ON/OFF, with activation thresholds set by the human operator. Activation commands are sent to all alarm agents and only to the sprinkler agent in the affected room. Deactivation happens when the building has been evacuated and no smoke or fire remains.

Acting agents emulate the actuators of the fire suppression and alarm control system. The two types, sprinklers and alarms, have similar operation (Algorithm 3). Acting agents have their own IDNs, used for the conversation with the controller. In the graphical interface of the simulation engine, the alarms emit a light signal and become visible as red elements when active, positioned in the center of the room they serve. Sprinkler activation is visible in the graphical interface as a burst of blue particles.

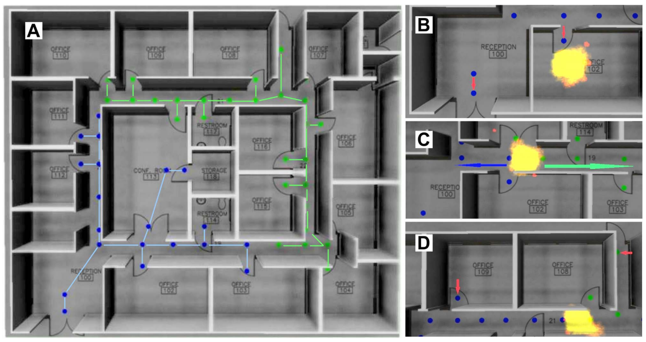

Fig. 5. Guidance. (a) Indicators visibility graph for fire drill case. (b) Office routes for fire in office case. (c) Hallway routes for fire in hallway. (d) Office routes for a fire in hallway.

Algorithm 4: Case-Based Route Computation.

```
if fire exists then
    retrieve fire location;
    create safety graph (SG);
end
foreach node on the visibility graph (VG) do
    determine closest exit (CE);
    case Fire Drill do
        calculate shortest route to CE on VG;
    case Fire with Two Accessible Exits do
        case End node in room with fire do
            calculate shortest route to CE on VG;
        case Intersection node in room with fire do
            find closest node on SG;
            create temporary edge;
            if all possible temp. edges pass through fire
              then
                select edge with least smoke;
            end
            calculate shortest route to CE on SG;
        case Any node in room without fire do
            calculate shortest route to both exits on SG;
            foreach pair of distinct route sections do
                compare distances to fire location;
            end
            choose route farthest from fire location;
        end
    case Fire with One Accessible Exit do
        case End node in room with fire do
            calculate shortest route on VG;
        case Intersection node in room with fire do
            find closest node on SG;
            create temporary edge;
            calculate shortest route on SG;
        case Any node in room without fire do
            calculate shortest route on SG;
        end
    end
    assign associated indicator to route;
end
```

D. Guidance Control

Safe evacuation of humans is a complex and nonlinear process involving several factors such as the architectural features of the building, the evolution of the fire, and the behavior of the people in the building, leading to major difficulties in understanding and streamlining the evacuation process. The prevalence of larger structures, which have increased room sizes, functional diversity, burning materials, high electrical loads, etc., requires adequate solutions [39].

In this article, we design a guidance control system that computes the shortest and *safest* routes in case of emergency based on fire location and displays them using a series of visual indicators. Each route begins in each room on the floor plan and ends in one of the exits.

Route indicators are light emitters (e.g., LED) embedded in the floor. In real-world implementations, these indicators would be directional strips and in the simulated world of the model, the human agents perceive them as such, thus knowing the routing direction. For computational reasons, however, the graphical interface only displays the *visibility graph* (i.e., the minimal amount of indicator-nodes for which there are no obstacles between neighboring nodes), creating the effect of dotted lines. Each route is a succession of indicator-nodes on a directed graph. Each node can display one of two colors, blue or green, associated with each of the two exits. The command received from the guidance controller assigns each node to a route by changing their color to either blue or green. When the indicators are deactivated, they are not visible on the graphical interface. Fig. 5(a) shows the visibility graph for a fire drill.

The guidance controller agent computes the shortest and safest evacuation routes based on fire and exit locations. Route safety is determined using a case-based algorithm, within which shortest paths are obtained using Dijkstra's algorithm, weighted to favor routes with less smoke. Route computation is triggered by an incoming request from the fire suppression and alarm controller, which can be either for a drill or when a fire event occurs. For each room on the floor plan (associated in the visibility graph with an end node that is not an exit), the guidance controller computes a route to the nearest exit, which is then transmitted to the route indicators. The indicator deactivation command is triggered when the guidance controller itself receives a deactivation request.

Algorithm 4 illustrates the case-based route computation. The safety graph is generated in case of fire by eliminating the nodes closest to the fire, and subsequently the edges passing through the fire location. The main cases are as follows: Fire drill with no constraints [Fig. 5(a)], both exits accessible, and one exit blocked. In this implementation, we assume only one fire location is active, and thus end nodes near fire compute their routes on the visibility graph [Fig. 5(b)]. Intersection nodes near fire must use the safety graph with themselves as origin points [Fig. 5(c) shows two opposing routes, each guiding away from the fire]. All other nodes compute the closest routes to all accessible exits on the safety graph, but in the case of two exits, the route farthest from the fire is chosen [Fig. 5(d)]. The case in which no valid routes can be found is of special importance, since it showcases a fault in floor plan design from which, in certain contexts, safe evacuation is not possible.

IV. Modeling the Crowd

The crowd model aligns with the approach presented in Section II; in what follows, we present the design of the human agent as crowd participant. The user interface allows setting

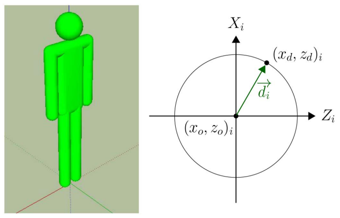

Fig. 6. Graphical model of human agent and $X_iO_iZ_i$ plane representation.

the number of human agents, while their initial placement on the floor plan is randomized. Agent diversity is ensured by probability-based decisions and internal state updates, thus obtaining behaviors of people who walk faster or slower, have different responses to smoke inhalation, knowledge of exit positions, or levels of spatial orientation within the floor plan. Some, under the stress of evacuation, make decisions to rush to the exit farthest from them. For the rigid body physics behavior in the simulated jMonkeyEngine environment [40], we use Bullet Physics [41], a library providing physics simulation (forward and inverse dynamics and kinematics, collision detection) for games, robotics, etc.

A. Internal Dynamics and State Updates

The human agent is represented by an entity (Fig. 6) that, in the graphic engine, can display physical properties akin to real life. Human agents move at different speeds, can collide with other agents or with the walls, and can inhale smoke. The behavior associated with these processes is the one described by (2), *internal dynamics*. For agent a_i, the inputs of this behavior are as follows: a) w_i: M_j smoke quantity in room j, state (ON/OFF) of alarms, state (ON/OFF) and color (green/blue) of guidance indicators, wall collision-generated forces, distance to fire δ_i; and b) $\{y_{i-\Delta}, \ldots, y_{i+\Delta}\}$: collision-generated forces from other human agents. The internal states x_i are as follows: inhaled smoke m_i, agent position $(x_o, z_o)_i$, walking direction $\vec{d_i}$, speed s_i, and life status λ_i. The output y_i is composed of the walking direction and speed to be processed for the representation of movement on the graphical interface. The ABSM is discrete and all dynamics are computed in discrete time of step k [sampling time 500 (ms)].

The *world model* or knowledge base of the agent contains the location of exits and the visibility graph (world map).

The internal state of the agent is updated based on previous states and inputs. Smoke inhalation affects the life status and movement speed, alarms trigger the evacuation behavior, while the controller updates the movement direction.

State update: Inhaled smoke. Smoke causes asphyxiation, serious injuries to the upper airways, and chemical irritation due to ash and toxic gases [42]. The inhaled smoke state variable m_i is based on the smoke quantity M_j in room j, smoke density ρ_s, distance to fire δ_i, and the approximation of the smoke plume of volume M_j/ρ_s as it lowers from the ceiling

$$m_k^i = m_{k-1}^i + \nu_i^k, \nu_i^k = \begin{cases} \rho_i \left(k_3 \delta_i^k / R_{pl}^k + k_4\right), \nu_i^k > 0 \\ 0, \nu^k \leqslant 0. \end{cases} \tag{7}$$

where ρ_i is a diversity parameter, while $k_3 < 0$ and $k_4 > 0$ are dependent on breathing rate. For $R_{pl}^k = \left(3M_j^k / (2\pi\rho_s)\right)^{1/3}$, the

Algorithm 5: Pedestrian Speed Update.

```
Input: Agent $a_i$, inhaled smoke $m_i$
Data: Agent speed $s_i$, $fire\_location$, $ws_i \in [0.8, 1]$,
      $rs_i \in [1.1, 1.4]$, $ss_i \in [0.3, 0.7]$
1  if smoke exists then
2      $inhale\_smoke(m_i, dist(a_i, fire\_location))$
3      $rs_i \leftarrow s_i - \omega \cdot m_i$
4  end
5  if $\neg fire \wedge \neg\, alarm$ then
6      $s_i \leftarrow ws_i$
7  else if $\neg fire \wedge alarm$ then
8      $s_i \leftarrow rs_i$
9  else if $fire \wedge alarm$ then
10     $s_i \leftarrow ss_i$
11 else
12     $s_i \leftarrow ss_i$
13 end
14 if $rs_i < ss_i$ then
15     $ss_i \leftarrow rs_i$
16 end
```

ceiling radius of the smoke spread, and the term $\delta_i^k / R_{pl}^k < 1$ models the relative position of the agent inside the smoke plume at time k. Quantity ν_i decreases farther from the fire, while its rate of decrease lowers as the plume becomes larger.

State update: Direction and current position. In the XYZ space, XOZ is the floor plane on which an agent moves, while its position on the OY axis is fixed. The current position of the agent is updated as it moves. Within the virtual space, each agent has its own coordinate system $X_iO_iZ_i$ (Fig. 6), with the current position in the XOZ plane $(x_o, z_o)_i$ as origin. The walking direction $\vec{d_i}$ is a vector originating in $(x_o, z_o)_i$ and ending in $(x_d, z_d)_i$, the target position for movement. The agent direction $\vec{d_i}$ is updated by the controller.

State update: Speed. The agent moves at different speeds in an emergency situation [43]. Without disturbances (smoke, active alarm, obstacles, etc.) on the floor plane XOZ, the agent moves between two points in a straight line with a base walking speed $ws_i \in [0.8, 1]$ (m/s), but switches to a running speed $rs_i \in [1.1, 1.4]$ (m/s) when alarms are active, and to $ss_i \in [0.3, 0.7]$ (m/s) when walking through areas with smoke. The triggering of the evacuation behavior considers the human reaction time to the alarms (pre-evacuation procedures). Real-world reaction times are difficult to measure during real fires [44], [45] and experimental studies found that people might ignore fire alarms if they are subjected to too many drills [46]. In this article, we consider the pedestrian agents to be rational, i.e., to not act against their own goals, thus we choose a randomized interval of 0–20 (s) as delay in reacting to the fire alarm [44], [47]. During evacuation, the agent speed s_i is adjusted based on environmental context, such as quantity of inhaled smoke. Algorithm 5 illustrates the speed calculation, where ω is an adjustable parameter modeling how strongly the inhalation of smoke affects the agent, while $inhale\_smoke(m_i, dist(a_i, fire\_location))$ implements the smoke inhalation state update described in (7). Smoke inhalation continues to affect the agent speed even after leaving the area with smoke present.

State update: Life status. The status of the agent as alive, safely evacuated, or dead is stored in this life status variable, updated based on whether or not the agent has reached an exit, has spent too much time in the area affected by fire, or has inhaled too much smoke (m_k^i larger than a preset tolerance). Stampede crushing and trampling happens during active fire evacuation bottlenecks or when parts of the crowd are moving in different directions [48]. We model this effect using crowd density and the accumulated effect of collisions. We choose a stampede density of 10 (agents/m^2) tuned from [48], [49] to account for the slim homogeneous agent shapes and lack of leg movement while walking in the graphic engine. The number of high-speed collisions (relative to maximum agent speed [50], [51]) at which we consider that the agent was crushed or trampled (life status changes to *dead*) is 10, based on maximum stampede density [48] and the mechanisms of traumatic asphyxia [52]. We consider the status of critically injured agents that would die in the hospital as *dead* and count them as such.

Output $y_i = [\vec{d_i} \;\; s_i]^T$ is sent to the graphic engine as a parameter independent of the number of visualization *fps* (frames/second), which determines the agent movement in the direction given by $\vec{d_i}$ and with speed s_i in m/s.

B. Motion Planner: Deliberative Controller Component

During regular operation, the agent moves randomly, with self-generated directions and speeds. Changes in alarm states (sensory inputs) trigger the evacuation behavior. During drills or emergencies, the agent moves between points of the visibility graph, on paths generated by the deliberative component of its control systems and adjusted by the reactive component.

The route of an agent through the XOZ plane is formed of nodes over the visibility graph. At each time step, the agent selects a target node t_i from the graph toward which it moves. When the guidance system is not active, the agent makes a choice for an exit and generates its evacuation route accordingly. The routes marked by active guidance indicators overwrite the self-generated paths. Algorithm 6 shows the deliberative case-based motion planner with route generation based on Dijkstra's algorithm. However, pedestrians might panic and they might not be familiar with the layout of the building; therefore, agents do not always choose the optimal path to the exit. A probability-based operation $get\_accuracy(a_i)$ models this choice, where *accuracy* is a user-defined parameter. In a real-world scenario, the decision to choose one exit over another is based on the distance to the exit, familiarity, and the emergency exit being perceived as open or closed [53]. The probability of misjudgment in a semipanic situation is up to 40% for residents without disabilities and up to 50% for persons with low stamina [54]. All these factors affect *accuracy*, which can be very low for some pedestrians or high for others. The agent is evacuated when a specified distance between it and an end node in the visibility graph is met, i.e., when the agent reaches one of the exits, it leaves the building.

The movement between nodes is governed by the position controller of the reactive component through functions $gen\_direction(a_i, t_i)$ that generates a new walking direction, and $reached(a_i, t_i)$ that checks if the current target node has been reached. The $next\_node(p_i)$ function identifies the next node on the graph closest to the exit, which then becomes the new target node t_i; this way, even if the local interactions with other pedestrians cause significant positions deviations, the agent avoids returning to a previous position on the graph. The other abbreviations in Algorithm 6 are as follows: gen_dest generates the next target node, $path\_avail$ checks if there is a path available for agent a_i, ind indicator, get_path returns the path to the closest exit based on agent a_i position and exit indicator color, $is\_empty(p_i)$ checks if there are still nodes in the path, $target\_reached(p_i)$ is a flag set when the agent reaches a node t_i in the path, and $distance(a_i, t_i)$ computes the distance to the target node. The $get\_shortest(p_i, p_j)$ function returns the shortest of paths p_i and p_j, while the $get\_longest(p_i, p_j)$ returns the longest path. The $random\_path(a_i)$ function returns the shortest path between current agent a_i and a random exit, but not necessarily the closest one.

Algorithm 6: Deliberative Component.

```
Input: Agent $a_i$
Data: Room $r_j$ where $a_i$ is located, current target
      node $t_i$ of $a_i$, path $p_i$
1  while $a_i$ alive and not evacuated do
2      if Alarm disabled then
3          if $\neg(path\_avail(a_i))$ then
4              $t_i \leftarrow gen\_dest(a_i, r_j)$
5              $target\_reached \leftarrow false$
6          if $path\_avail(a_i) \wedge \neg(reached(a_i, t_i))$ then
7              $d_i \leftarrow gen\_direction(a_i, t_i)$
8          if $path\_avail(a_i) \wedge reached(a_i, t_i)$ then
9              $target\_reached \leftarrow true$
10         end
11     else if (Alarm enabled $\vee$ $is\_fire(r_j)) \wedge$
           $\neg(path\_avail(a_i))$ then
12         $exit\_ind \leftarrow get\_closest\_ind(a_i)$
13         $ind\_color \leftarrow get\_color(exit\_ind)$
14         if $exit\_ind$ is available then
15             $p_i \leftarrow get\_path(a_i, ind\_color)$
16             $t_i \leftarrow next\_node(p_i)$
17         else
18             $p_{shortest} \leftarrow dijkstra(a_i, exit_0)$
19             for $exit \in available\_exits\_set$ do
20                 $p_i \leftarrow dijkstra(a_i, exit)$
21                 $p_{shortest} \leftarrow get\_shortest(p_i, p_{shortest})$
22             end
23             $p_i \leftarrow random\_path(a_i)$
24             if $get\_accuracy(a_i) < accuracy$ then
25                 $p_i \leftarrow p_{shortest}$
26             end
27         end
28     end
29     if $reached(a_i, t_i) \wedge \neg(is\_empty(p_i))$ then
30         $t_i \leftarrow next\_node(p_i)$
31     end
32     if $distance(a_i, t_i) < 0.45$ then
33         $target\_reached \leftarrow true$
34     end
35     if $is\_empty(p_i) \wedge distance(a_i, t_i) < 0.25$ then
36         $a_i$ evacuated
37     else
38         $d_i \leftarrow gen\_direction(a_i, t_i)$
39     end
40 end
```

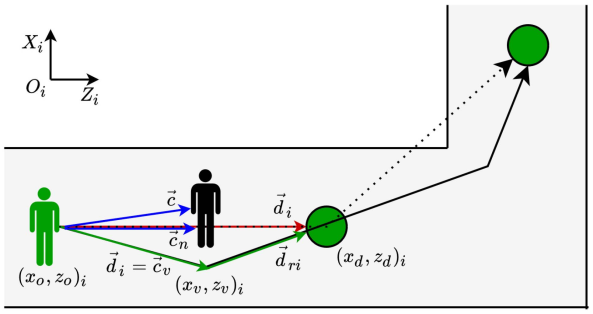

Fig. 7. Direction update during collision avoidance.

C. Reactive Component

The reactive component of the motion control architecture for the pedestrian agent is formed of a *position controller* subsumed with an obstacle avoidance behavior. Since the floor plan in this particular scenario does not include furniture, we focus on the other humans as obstacles. In this case, the behavior we implement is one of *collision avoidance*.

The position controller governs the movement in a specified direction d_i with a specified speed s_i. For a trip without obstacles or collisions, the controller computes the direction d_i based on the agent current position $(x_o, z_o)_i$ and the position of the next node it needs to reach. In areas with high occupant density, some agents push through the crowd in their desire to reach the exit quickly, colliding with agents that move at lower speeds. When an unavoidable collision sends the agent off its straight path, the position controller updates the direction d_i based on the new agent position (Fig. 2) toward the next node in the visibility graph (the agent does not turn around if it has passed a node in the specified exit direction, i.e., the position setpoint is always positive, as generated by $next\_node(p_i)$ in Algorithm 6). The position controller ensures that the agent travels the distance to target via a proportional law, integrated with the graphic engine and sufficient for position control [55].

Collision avoidance. During evacuation, agent interactions are represented by collisions. These are disturbances that the agent attempts to avoid. Algorithm 7 shows the collision avoidance behavior. When another agent is detected on the direct path of agent a_i (or close enough that a_i cannot continue forward without collision), within radius β, a new direction is computed relative to the current agent direction d_i toward the target (Fig. 7). Once no other agent is interposed between a_i and its target, the position controller takes over. The general direction of the agent is maintained: The new direction is $\vec{c_v}$ until the obstacle is cleared, followed by the recovered direction either toward the initial target $\vec{d_{ri}}$ (Fig. 7), or toward the next node in the graph if the initial target has been surpassed. Collision avoidance with walls and fixed door frames is similarly implemented. This behavior accounts for cases in which, while avoiding one agent, a different collision happens, which is expected as part of crowd dynamics under stress conditions. Perception is limited because attention is a finite resource, and therefore human agents rushing toward an exit will not be able to deliberately plan their path among other pedestrians in the crowd. Thus, the reactive nature of this behavior allows us to emulate the inattentional blindness of weaving through a crowd while influenced by emergency anxiety and panic modes, i.e., making decisions on direction under pressure [56]–[59].

Algorithm 7: Collision Avoidance Behavior.

```
Input: Agent $a_i$ positioned in $(x_o, z_o)_i$, collision point
       vector $\vec{c}$ with target coordinates $(x_c, z_c)_i$,
       agent direction $\vec{d_i}$: $(x_d, z_d)_i$;
Data: Normalized collision vector $\vec{c_n}$: $(x_n, z_n)_i$;
Result: Collision avoidance direction $\vec{c_v}$: $(x_v, z_v)_i$;
1  $\vec{c_n} = \vec{c}/|\vec{c}|$
2  if $|x_d - x_n| < \beta$ then
3      if $x_d > x_n$ then
4          $x_v \leftarrow x_n + \beta$
5          if $x_v > 0$ then
6              $x_v \leftarrow 1 - x_v$
7              $z_v \leftarrow -sgn(z_n)\sqrt{1 - x_v^2}$
8          else
9              $z_v \leftarrow sgn(z_n)\sqrt{1 - x_v^2}$
10         end
11     else
12         $x_v \leftarrow x_n - \beta$
13         if $x_v < 0$ then
14             $x_v \leftarrow -1 - x_v$
15             $z_v \leftarrow -sgn(z_n)\sqrt{1 - x_v^2}$
16         else
17             $z_v \leftarrow sgn(z_u)\sqrt{1 - x_v^2}$
18         end
19     end
20     $\vec{d_i} \leftarrow \vec{c_v} : (x_v, z_v)_i$
21 end
```

V. Results and Discussion

To analyze the resulting emergent crowd behavior and to validate the ABSM, we simulated various scenarios. In what follows, *evacuated ped.* (pedestrians) is the number of agents that safely reach the exit; *total collisions* is the sum of all collisions between human agents and with the walls. First, we perform the overall crowd assessment during evacuation procedures, in scenarios with 142 total human agents. Second, we present a suite of patterns emerging from agent interactions.

A. Overall Assessment

Fire drill. The impact of the closest exit estimation accuracy for pedestrians is presented in Table I and Fig. 8(a): Drills with enabled indicators take, on average, up to 38.8% less time, with 77.2% more collisions (including soft bumps with pedestrians or walls). The case with 100% closest exit estimation accuracy is ideal, in which every human agent does not panic and knows exactly which exit is closest to their current position when the alarm starts. Table I and Fig. 8(b) show that this case is comparable with the one when the exit indicators are enabled, illustrating the usefulness of the guidance controller.

Active fire. Three scenarios are analyzed with fire present in the building: Guidance indicators enabled with 6 s delay, and disabled with various accuracy. Table I and Fig. 8(c) show that enabling the indicators results in much shorter total evacuation times (up to 40%) and significantly less collisions (up to 77.4%). Fig. 8(d) shows a single case with fire in the upper corridor, in which the density of pedestrians is 200% higher without indicators, and while the smoke production is mitigated by the

TABLE I
SIMULATION RESULTS ACROSS MULTIPLE EXPERIMENTS (10 FOR EACH CASE)

Experiment	Exit indicators	Closest exit estimation accuracy	Mean evacuation time [sec]	Standard deviation evacuation time	Mean total collisions	Standard deviation total collisions	Mean number of victims Stampede	Mean number of victims No stampede
Drill	Disabled	0%	64.7	11.42	5121.3	366.53	N/A (calm behavior)	0
		25%	55.9	3.45	3204.9	293.25		
		50%	52.1	3.60	2136.2	404.09		
		75%	50.4	2.63	1406.4	92.71		
		100%	39.61	1.44	1101.92	130.46		
	Enabled	N/A	39.6	1.43	1168.7	137.98		
Active fire	Disabled	0%	65.4	21.70	5206.4	336.03	9.9	1.2
		25%	58.9	11.81	3276.2	443.69	2.1	0.5
		50%	53.8	2.78	2075.8	191.38	1.6	0
		75%	50.9	2.64	1390.7	89.63	1.5	0
		100%	39.6	1.35	1220.4	143.91	1.3	0
	Enabled (no delay)	N/A	39.4	1.90	1173.2	157.44	0.5	0
	Enabled with 6 [sec] delay	50% during the delay N/A otherwise	41.1	4.99	1149.6	135.53	0.5	0

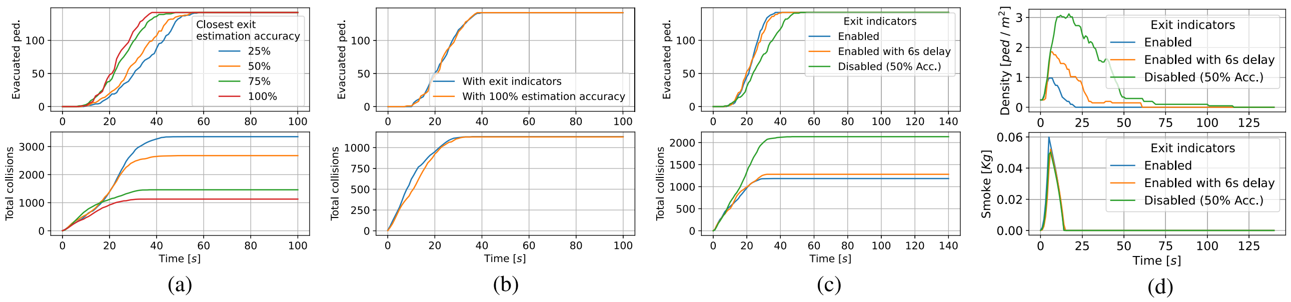

Fig. 8. Fire drill and emergency evacuation results (single case example). (a) Fire drill without exit indicators. (b) Fire drill results comparison. (c) Fire emergency evacuation. (d) Human density and smoke.

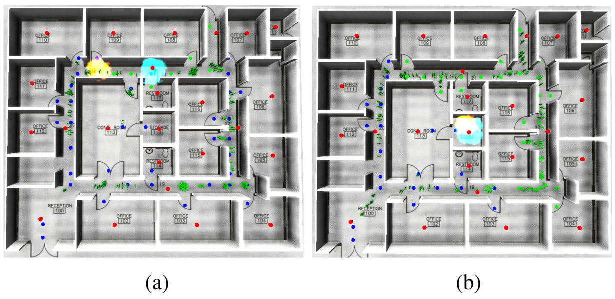

Fig. 9. Fire location effect on guidance controller. (a) Exit blocking. (b) Non-exit blocking.

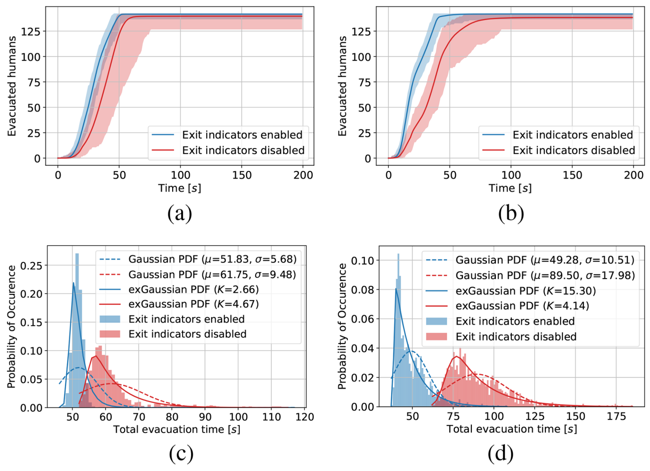

Fig. 10. Cumulative results (top) and histograms (bottom). (a) With agent delayed reaction. (b) Without delayed reaction. (c) With agent delayed reaction. (d) Without delayed reaction.

fire suppression system, this scenario shows that the likelihood of victims is higher without guidance.

Fig. 9 shows the effect of the fire location on the route computation by the guidance controller. If the fire location is blocking one of the exits [Fig. 9(a)], the safe routes predominantly point toward the other exit, and most evacuation routes are longer. When the fire is nonexit blocking [Fig. 9(b)], the two sets of routes are balanced in length.

Crowd reaction time. In psychology, *reaction time* (RT) is the system equivalent of response time: The duration between the appearance of an exogen stimulus and the occurrence of a specific response to that stimulus [60]. RT is usually measured and modeled as an ex-Gaussian probability density function (PDF) [61], in which the probability of occurrence of the expected response is plotted against the response times. We introduce *crowd reaction time* (CRT): The duration between the activation of alarms and the finalized evacuation.

We ran 1000 simulations for each of four scenarios (Fig. 10): Indicators enabled or disabled (25% accuracy for inefficient decisions in panic mode), with or without human agent reaction delay. The total evacuation times are comparable, even though the overall crowd response has ~4 s larger delay in the delayed reaction case [Fig. 10(a) and (b)].

Fig. 10(c) and (d) shows the probability of occurrence for the specified total evacuation times, fitted against Gaussian and ex-Gaussian PDFs (μ normal distribution mean, σ standard deviation, K exponential shape parameter). The total evacuation

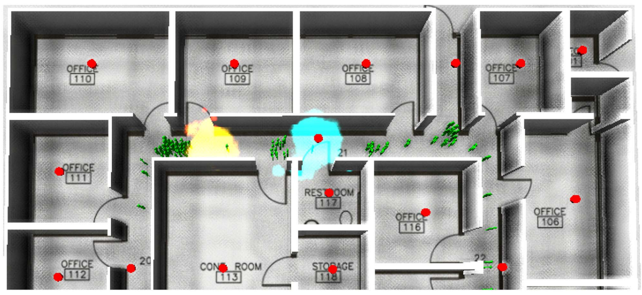

Fig. 11. Bottleneck formation during emergency evacuation.

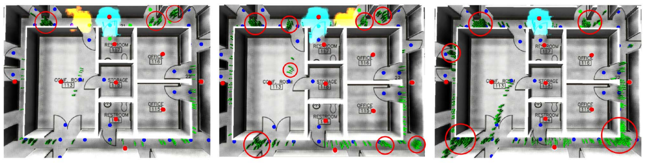

Fig. 12. Cluster formation for different crowd sizes. (a) With 80 agents. (b) With 142 agents. (c) With 284 agents.

times are more consistent in the case with randomized human agent reaction delay. We found that when pedestrians react to alarms at different times, there is less crowding around the exit areas and in the hallways.

In all cases, the CRT distribution is an ex-Gaussian PDF, mirroring for the crowd the RT observations for individuals. This shows that the crowd behaves as an organic whole: *An emergent, complex system-of-systems*.

This interesting result aligns with crowd psychology concepts, furthering ABSM-driven studies of crowd behavior.

B. Emergent Crowd Behaviors

Bottleneck formation. A likely scenario is when building occupants do not have information on the location of the fire (Fig. 11). At alarm activation, pedestrians use the fire drill routes, rushing toward the closest exit. If the fire blocks their path, a bottleneck forms, leading to agents pushing against each other. Identifying areas with bottleneck risks is a useful feature for testing and validating the evacuation safety of floor plans during the design stage.

Cluster formations can be observed for various crowd sizes, showing the scaling property of the ABSM as a complex system. Fig. 12 shows frames with 80, 142, and 284 pedestrian agents in the model. These patterns emerge naturally due to collisions, different speeds, and common goals.

Effect of the guidance controller on the crowd dynamics. The guidance controller enables pedestrians to easily find the nearest exit [Fig. 13(a)], thus offering external knowledge to incorporate into their own reasoning process. Otherwise, pedestrians estimate the nearest exit, resulting in the formation of collision points. Here, we consider $accuracy = 25\%$ to analyze the case of 3 in 4 people choosing wrong, e.g., no drills in the building or no evacuation maps provided. With guidance [Fig. 13(a)], pedestrians move in the same direction, whereas without it, two groups form, each aiming toward a far exit [Fig. 13(b)]. These two crowd subsets intersect along the same hallway, creating bottlenecks for each other.

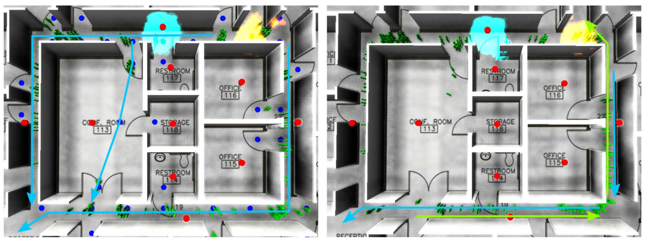

Fig. 13. Collision point formation. (a) With routing. (b) Without routing.

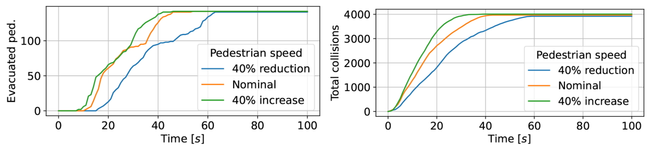

Fig. 14. Effect of pedestrian speed.

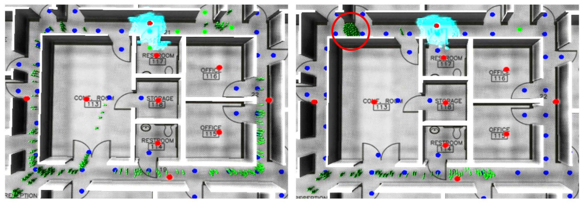

Fig. 15. Bottleneck formation depending on pedestrian speed. (a) 40% speed decrease. (b) 40% speed increase.

Effect of the pedestrian agent movement speed. In this set of scenarios, we consider three cases, in which the crowd is formed of agents with a) nominal base speed as in algorithm 5; b) 40% base speed increase; and c) 40% decrease. Simulations show that increasing or decreasing the base speed of the agents (Fig. 14) does not influence the number of collisions between pedestrians, but the higher the agents speed, the faster the building is completely evacuated.

There is, however, an observable change in crowd cohesion. In the case with slower movement [Fig. 15(a)], emulating a calmer advance toward the exits, there are less clusters of smaller size. With the increase in speed [Fig. 15(b)], emulating running, we observe bottleneck formation and larger clusters. The bottleneck highlighted in Fig. 15(b) occurred at $T = 15$ s and the evacuation rate slowed down after this point in time. Fig. 14 shows the bottleneck impact on the evacuation rate.

Analysis against the crowd models is discussed in [5] (pp. 11–13). Our approach simulates crowd dynamics: Lane formation, crossing flows, faster-is-slower, blocking, stop-and-go waves, vortices and turbulence, and obtains psycho-social behaviors through emergence. Our deliberative-reactive pedestrian architecture allows for psychological and health characteristics to be modeled. Unlike macroscopic models aimed at simulating the crowd from a top-down perspective, our design obtains behaviors at crowd level in a bottom-up manner. Regarding group dynamics, our approach is able to illustrate group formation, group cohesion, and group–individual interaction.

The computational efficiency of our ABSM implementation is good: Although the jMonkey engine requires graphical processing resources, it is adaptable to parallel computation. This feature leverages the JADE scaling ability by distributing the workload across multiple networked hosts, e.g., for computationally intensive simulations of larger, multistory buildings. We compared our results with [62] (p. 23, Table I, Columns 1 and 2), using the same type of GPU, Nvidia GeForce GTX 1060. We obtained on average 120.7 fps compared to 125 fps in [62], for a) 254 agents in our model (including pedestrians with deliberative-reactive architecture, static agents, and the two controller agents); and b) 250 particle-based pedestrian agents in [62]. In our case, as opposed to [62], the model also includes the building itself (walls and rooms), rendering for fire and sprinklers, control and collision logic, smoke computation with discrete differential equations, and the interaction between pedestrians and smoke.

The overall result of our predictive agent-based design using DCSs is a crowd model with realistic behaviors during fire evacuation. The design is scalable to crowds of different sizes and allows for more human-specific behaviors to be integrated in the controllers of the pedestrian agents, in either the deliberative or reactive component.

Supplementary video material demonstrating the ABSM and crowd dynamics during guided evacuation is provided.

VI. Conclusion

Crowd simulation could greatly benefit from the inclusion of social dynamics and the duality of the deliberative-reactive human reasoning model. In this article, we answer this issue by proposing an agent-based crowd modeling technique using DCSs. As a complex system, the crowd dynamics are obtained through bottom-up autonomous interactions, while the simulation model adheres to the principles of the generative experiment.

To illustrate this approach, we presented a design scenario for fire drills and evacuation on an office building floor, for which we also designed a fire suppression control system and a guidance controller. Results show that the crowd model based on agent autonomy and local interactions is able to generate higher level crowd dynamics through emergence.

Further work is focused on enriching the pedestrian deliberative component with a reasoning engine modeling rules of social interaction. Placing the crowd in other scenarios, for instance, travel across urban spaces, venue ingress, etc., is also of great interest, in a collaborative interdisciplinary setting.

References

[1] F. Mirahadi and B. Y. McCabe, “EvacuSafe: A real-time model for building evacuation based on Dijkstra’s algorithm,” *J. Building Eng.*, vol. 34, pp. 1–14, 2021.

[2] A. Shipman and A. Majumdar, “Fear in humans: A glimpse into the crowd-modeling perspective,” *Transp. Res. Rec.*, vol. 2672, no. 1, pp. 183–197, 2018.

[3] P. M. Kielar and A. Borrmann, “Spice: A cognitive agent framework for computational crowd simulations in complex environments,” *Auton. Agents Multi-Agent Syst.*, vol. 32, no. 3, pp. 387–416, 2018.

[4] A. Simonov, A. Lebin, B. Shcherbak, A. Zagarskikh, and A. Karsakov, “Multi-agent crowd simulation on large areas with utility-based behavior models: Sochi olympic park station use case,” *Procedia Comput. Sci.*, vol. 136, pp. 453–462, 2018.

[5] S. Yang, T. Li, X. Gong, B. Peng, and J. Hu, “A review on crowd simulation and modeling,” *Graphical Models*, vol. 111, pp. 1–19, 2020.

[6] M. Čadeckỳ, M. Varga, and N. Adamko, “Mesoscopic movement model of deliberative pedestrian agents,” in *Proc. Int. Conf. Inf. Digit. Technol.*, 2015, pp. 61–67.

[7] N. Gaud, S. Galland, F. Gechter, V. Hilaire, and A. Koukam, “Holonic multilevel simulation of complex systems: Application to real-time pedestrians simulation in virtual urban environment,” *Simul. Model. Pract. Theory*, vol. 16, no. 10, pp. 1659–1676, 2008.

[8] M. Lachowicz, “Microscopic, mesoscopic and macroscopic descriptions of complex systems,” *Probabilistic Eng. Mechanics*, vol. 26, no. 1, pp. 54–60, 2011.

[9] P. Mathieu, G. Morvan, and S. Picault, “Multi-level agent-based simulations: Four design patterns,” *Simul. Model. Pract. Theory*, vol. 83, pp. 51–64, 2018.

[10] F. de Souza, O. Verbas, and J. Auld, “Mesoscopic traffic flow model for agent-based simulation,” *Procedia Comput. Sci.*, vol. 151, pp. 858–863, 2019.

[11] R. Alqurashi and T. Altman, “Multi-class agent-based model of crowd dynamics,” in *Proc. Int. Conf. Comput. Sci. Comput. Intell.*, 2017, pp. 1801–1802.

[12] A. Sieben, J. Schumann, and A. Seyfried, “Collective phenomena in crowds-Where pedestrian dynamics need social psychology,” *PLoS One*, vol. 12, no. 6, pp. 1–19, 2017.

[13] X. Yang, H. Dong, Q. Wang, Y. Chen, and X. Hu, “Guided crowd dynamics via modified social force model,” *Physica A: Stat. Mechanics Appl.*, vol. 411, pp. 63–73, 2014.

[14] M. Lhommet, D. Lourdeaux, and J.-P. Barthés, “Never alone in the crowd: A microscopic crowd model based on emotional contagion,” in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol.*, 2011, pp. 89–92.

[15] B. Doerr and R. Linares, “Decentralized control of large collaborative swarms using random finite set theory,” *IEEE Trans. Control Netw. Syst.*, vol. 8, no. 2, pp. 587–597, Jun. 2021.

[16] L. Bakule and M. Papik, “Decentralized control and communication,” *Annu. Rev. Control*, vol. 36, no. 1, pp. 1–10, 2012.

[17] Z. Li, C. H. Sim, and M. Y. H. Low, “A survey of emergent behavior and its impacts in agent-based systems,” in *Proc. 4th IEEE Int. Conf. Ind. Inform.*, 2006, pp. 1295–1300.

[18] A. Poulos, F. Tocornal, J. C. de la Llera, and J. Mitrani-Reiser, “Validation of an agent-based building evacuation model with a school drill,” *Transp. Res. C: Emerg. Technol.*, vol. 97, pp. 82–95, 2018.

[19] M. Patrascu and M. Drăgoicea, “Integrating agents and services for control and monitoring: Managing emergencies in smart buildings,” in *Service Orientation in Holonic and Multi-Agent Manufacturing and Robotics*. Berlin, Germany: Springer, 2014, pp. 209–224.

[20] R. de Amorim Silva and R. T. V. Braga, “Simulating systems-of-systems with agent-based modeling: A systematic literature review,” *IEEE Syst. J.*, vol. 14, no. 3, pp. 3609–3617, Sep. 2020.

[21] A. Ion, C. Berceanu, and M. Patrascu, “Applying agent based simulation to the design of traffic control systems with respect to real-world urban complexity,” in *Proc. Eur. Conf. Multi-Agent Syst. Agreement Technol.*, 2015, pp. 395–409.

[22] A. Moreno, “On minimal autonomous agency: Natural and artificial,” *Complex Syst.*, vol. 27, no. 3, pp. 289–313, 2018.

[23] T. Bertolotti, “Generative and demonstrative experiments,” in *Proc. Model-Based Reasoning Sci. Technol.*, 2014, pp. 479–498.

[24] C. M. Macal, “Everything you need to know about agent-based modelling and simulation,” *J. Simul.*, vol. 10, no. 2, pp. 144–156, 2016.

[25] R. Girardi and A. Leite, “A survey on software agent architectures,” *IEEE Intell. Inform. Bull.*, vol. 14, no. 1, pp. 8–20, 2013.

[26] J. Wu, H. Wang, N. Li, and Z. Su, “Formation obstacle avoidance: A fluid-based solution,” *IEEE Syst. J.*, vol. 14, no. 1, pp. 1479–1490, Mar. 2020.

[27] F. Bellifemine, A. Poggi, and G. Rimassa, “JADE–A FIPA-compliant agent framework,” in *Proc. Int. Conf. Pract. Appl. Intell. Agents Multi-Agent Technol.*, 1999, pp. 1–12.

[28] *jMonkeyEngine*. 2017. Accessed: Apr. 11, 2021. [Online]. Available: www.jmonkeyengine.org

[29] D. Grassl, “Total system efficiency: Condensing boiler system myth busting,” *Cleaver Brooks White Papers*, Accessed: Aug. 11, 2021.

[30] *SketchUp*. Accessed: Mar. 11, 2021. [Online]. Available: www.sketchup.com

[31] *FSE, Fire safety engineering*. ISO/TS 16733-2:2021(en), British Standards Inst., London, 1999.

[32] A. Ion and M. Patrascu, “A scalable algorithm for self-organization in event-triggered networked control systems,” in *Proc. 18th Eur. Control Conf.*, 2019, pp. 2725–2730.

[33] M. J. Hurley et al., *SFPE Handbook of Fire Protection Engineering*. Berlin, Germany: Springer, 2015.

[34] B. McCaffrey, “Momentum implications for buoyant diffusion flames,” *Combustion Flame*, vol. 52, pp. 149–167, 1983.

[35] G.-Y. Wu and R.-C. Chen, “The analysis of the natural smoke filling times in an atrium,” *J. Combustion*, vol. 2010, pp. 1–9, 2010.

[36] H.-J. Kim and D. G. Lilley, “Heat release rates of burning items in fires,” *J. Propulsion Power*, vol. 18, no. 4, pp. 866–870, 2002.

[37] I. Bennetts and I. Thomas, “Designing buildings for fire safety: A risk perspective,” *Prog. Struct. Eng. Mater.*, vol. 4, no. 2, pp. 224–240, 2002.

[38] A. E. Çetin et al., “Video fire detection–review,” *Digit. Signal Process.*, vol. 23, no. 6, pp. 1827–1843, 2013.

[39] H. Ran, L. Sun, and X. Gao, “Influences of intelligent evacuation guidance system on crowd evacuation in building fire,” *Automat. Construction*, vol. 41, pp. 78–82, 2014.

[40] *jMonkeyEngine Documentation on Gravity, Collisions, Forces*. Accessed: Dec. 03, 2022. [Online]. Available: https://wiki.jmonkeyengine.org/docs/3.4/physics/physics.html

[41] Bullet real-time physics simulation. Accessed: Dec. 03, 2022. [Online]. Available: https://pybullet.org

[42] D. Rorison and S. McPherson, “Acute toxic inhalations,” *Emerg. Med. Clin. North America*, vol. 10, no. 2, pp. 409–435, 1992.

[43] K. Fridolf, K. Andrée, D. Nilsson, and H. Frantzich, “The impact of smoke on walking speed,” *Fire Mater.*, vol. 38, no. 7, pp. 744–759, 2014.

[44] S. Björnsson, “Automatic fire alarms-response procedures,” *Rep. 5391*, Lund University, Sweden, 2012. [Online]. Available: https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=3917860&fileOId=3917861

[45] G. Proulx, “Response to fire alarms,” *Fire Protection Eng.*, vol. 33, pp. 95–113, 2007.

[46] G. Proulx, “Why building occupants ignore fire alarms,” *Construction Technol. Update*, vol. 42, pp. 1–4, 2000.

[47] D. Bruck, “The who, what, where and why of waking to fire alarms: A review,” *Fire Saf. J.*, vol. 36, no. 7, pp. 623–639, 2001.

[48] M. M. de Almeida and J. von Schreeb, “Human stampedes: An updated review of current literature,” *Prehospital Disaster Med.*, vol. 34, no. 1, pp. 82–88, 2019.

[49] M. Pretorius, S. Gwynne, and E. R. Galea, “Large crowd modelling: An analysis of the Duisburg love parade disaster,” *Fire Mater.*, vol. 39, no. 4, pp. 301–322, 2015.

[50] Y. Zhang, Z. Yang, and Z. Sun, “A dynamic estimation method for aircraft emergency evacuation based on cellular automata,” *Adv. Mech. Eng.*, vol. 11, no. 2, pp. 1–12, 2019.

[51] V. V. Kholshevnikov and D. A. Samoshin, “Laws of motion of pedestrian flow-Basics for evacuation modeling and management,” in *Proc. Resilience Cities Terrorist Other Threats*, 2008, pp. 417–442.

[52] J. R. Gill and K. Landi, “Traumatic asphyxial deaths due to an uncontrolled crowd,” *Amer. J. Forensic Med. Pathol.*, vol. 25, no. 4, pp. 358–361, 2004.

[53] L. Benthorn and H. Frantzich, “Fire alarm in a public building: How do people evaluate information and choose an evacuation exit,” *Fire Mater.*, vol. 23, no. 6, pp. 311–315, 1999.

[54] J. Koo, B.-I. Kim, and Y. S. Kim, “Estimating the effects of mental disorientation and physical fatigue in a semi-panic evacuation,” *Expert Syst. Appl.*, vol. 41, no. 5, pp. 2379–2390, 2014.

[55] S. G. Tzafestas, “Mobile robot control and navigation: A global overview,” *J. Intell. Robot. Syst.*, vol. 91, no. 1, pp. 35–58, 2018.

[56] A. D. Castel, M. Vendetti, and K. J. Holyoak, “Fire drill: Inattentional blindness and amnesia for the location of fire extinguishers,” *Attention, Perception, Psychophys.*, vol. 74, no. 7, pp. 1391–1396, 2012.

[57] J. Lin, R. Zhu, N. Li, and B. Becerik-Gerber, “Do people follow the crowd in building emergency evacuation? A cross-cultural immersive virtual reality-based study,” *Adv. Eng. Inform.*, vol. 43, pp. 1–13, 2020.

[58] L. Fu, S. Cao, W. Song, and J. Fang, “The influence of emergency signage on building evacuation behavior: An experimental study,” *Fire Mater.*, vol. 43, no. 1, pp. 22–33, 2019.

[59] N. Shiwakoti, M. Sarvi, G. Rose, and M. Burd, “Consequence of turning movements in pedestrian crowds during emergency egress,” *Transp. Res. Rec.*, vol. 2234, no. 1, pp. 97–104, 2011.

[60] *Amer. Psychol. Assoc. (APA) Dictionary*. Accessed: Sep. 11, 2021. [Online]. Available: http://dictionary.apa.org/reaction-time

[61] R. Denga, “Need for speed: Applying ex-Gaussian modeling techniques to examine intra-individual reaction time variability in expert tetris players,” in *Proc. Annu. Meeting Cogn. Sci. Soc.*, 2021, pp. 2733–2739.

[62] O. Hesham and G. Wainer, “Explicit modeling of personal space for improved local dynamics in simulated crowds,” *ACM Trans. Model. Comput. Simul.*, vol. 31, no. 4, pp. 1–29, 2021.

Cristian Berceanu received the M.Sc. degree in advanced computer architectures from the University Politehnica of Bucharest, Bucharest, Romania, in 2017. He is currently working toward the doctoral degree in development of simulation models for particle systems using agent-based technologies at University Politehnica of Bucharest, Bucharest, Romania.

His research interests include multiagent systems, amorphous computing, and high-performance computing.

Ionuţ Banu received the B.Sc. degree in systems engineering and the M.Sc. degree in complex systems from the Faculty of Automatic Control and Computers, University Politehnica of Bucharest, Bucharest, Romania, in 2017 and 2019, respectively.

His research interests include multiagent systems, object-oriented programming, control systems, evolutionary computing, and multiobjective optimization.

Bettina S. Husebo received the M.D. degree from the University of Bonn, Germany, in 1988 and the Ph.D. degree in medicine from the University of Bergen, Norway, in 2008.

She is a Specialist in Anesthesia and Intensive Care, and is currently a Professor; Head of Innovation, Department of Global Public Health and Primary Care; and Head of the Center for Elderly and Nursing Home Care, University of Bergen (UiB), Bergen, Norway, where she is responsible for a national paradigm change in elderly care addressing digitalization, innovation, and modeling human behavior in elderly people and persons with dementia.

Monica Patrascu (Senior Member, IEEE) received the B.Sc. degree in control engineering, the M.Sc. degree in intelligent control systems, and the Ph.D. degree in systems engineering, in 2007, 2009, and 2012, respectively, all from the University Politehnica of Bucharest, Romania.

They are currently the Coordinator of the Complex Systems Laboratory, University Politehnica of Bucharest, Bucharest, Romania, working with complex and intelligent systems, and with the Centre for Elderly and Nursing Home Medicine, University of Bergen, Bergen, Norway, working with modelling complex biosystems.