



Modeling pedestrian behavior with Cell-DEVS: theory and applications

Ala'a Al-Habashna and Gabriel Wainer*

Abstract

In this work, we provide an approach for Modeling and Simulation (M&S) of crowds using Cellular Discrete Event System Specification (Cell-DEVS). We present many examples of Cellular Discrete Event System Specification entity-based crowd models, and we show how to use Cellular Discrete Event System Specification for entity-based modeling and simulation of crowds. We provide an approach for using Cellular Discrete Event System Specification theory in modeling and simulation of crowds, and we propose Cellular Discrete Event System Specification entity-based models for modeling and simulation of one-, two-, and three-dimensional movement of crowds. We extend the models above, and propose a more advanced model for crowd movement in multi-level building. Furthermore, we use this model for simulation of building evacuations. We propose another advanced model for crowd modeling, and deploy the model in occupancy analysis of buildings. Simulation results verify the usability of the proposed models.

Keywords

Cell-DEVS, crowd modeling, DEVS, modeling and simulation, pedestrian behavior

1. Introduction

The world's population is increasing rapidly and is moving towards urban areas. This has resulted in overcrowded areas, and a frequent occurrence of crowd phenomena.¹ Consequently, modeling and analysis of crowd behavior is an area of increasing interest, which has received a lot of attention by different researchers.

Crowd analysis can be used for developing crowd management strategies to increase safety in highly crowded situations, such as concerts. Crowd analysis can also be used in building design, in order to provide a more efficient use of spaces. Crowd analysis is also important in virtual environments as it leads to better simulation of crowd behavior in such artificial settings. As such, crowd behavior is now being investigated with applications in many areas such as safety, architectural design, industrial design, computer games, transportation, etc.²

In many situations, it could be very difficult or costly to study the behavior of crowd by real-life experiments. On the other hand, crowd may exhibit highly complex dynamics, which makes it difficult to characterize its behavior using pure mathematical and analytical models. As such, Modeling and Simulation (M&S) of crowds can be very useful in these situations. When modeling a small crowd (i.e., one with tens of individuals), it is easy to investigate the behavior of each individual. However, in a

large crowd (i.e., one with thousands of individuals), the interest is more in the overall emergent behavior.

Many models have been proposed recently for M&S of crowds. These models can be categorized based on the modeling approach into as fluid dynamics based models,³ social force-based models,⁴ agent-based models,⁵ and Cellular Automata (CA)-based models.⁶ Fluid-dynamics-based models (also referred to as flow-based models) study the crowd as a whole, and focus on the physical aspects of the crowd, such as the movement pattern.³ Because of its low granularity level, these models have light computational demand, which makes them suitable for modeling large-sized crowds. Social-force-based models consider the human motion as a complicated behavior that is subject to “social forces”. These models will be discussed in more detail in next section. CA-based models

Department of Systems and Computer Engineering, Centre for Visualization and Simulation (V-Sim), Carleton University, Ottawa, Ontario, Canada

*SCS member

Corresponding author:

Ala'a Al-Habashna, Department of Systems and Computer Engineering,
1125 Colonel By Drive, Carleton University, Ottawa, ON, K1S 5B6
Canada.

Email: alaaalhabashna@sce.carleton.ca

simulate the crowd using the CA theory, which is one of the oldest and most popular methods of natural computing. Such methods are similar to our approach, and hence will be explained in detail later. Both social force and CA models provide higher level of granularity than fluid-dynamics-based models, as they do not treat the crowd as a whole. They model the crowd as a set of homogeneous entities, and a set of global/local rules are used to control the behavior of these entities. Hence, such methods are called entity-based approaches.² Agent-based approaches⁵ have higher computational demands than the previous methods, which makes them more suitable for short-term simulations of small-sized crowds. Each individual is modeled as separate autonomous agent that takes decisions independently. The decision rules for each agent are usually customized based on information relevant to the agent.

In CA, the space under study is represented as a lattice of cells, with each cell being a state machine. As time advances, changes to the states of the cells are triggered based on the value of the cells and their neighbor cells. CA theory has been widely used for crowd modeling. CA is one of the oldest models of natural computing, dating back over half a century.⁷ It is a very popular approach owing to its simplicity and ability to model complex systems. CA is massively parallel, homogeneous, and all interactions are local. These properties exist in many real-world systems. As such, CA has been used successfully to simulate many physical and biological systems.

The Cell-DEVS formalism⁸ provides an asynchronous cellular modeling technique including a better definition of timing properties in the model and allowing combining models easily. Cell-DEVS is a combination of CA and Discrete-Event System Specifications (DEVS)⁹ with explicit timing delays.

We propose an approach for M&S of crowd using Cell-DEVS to overcome the shortcomings of the previous models. As Cell-DEVS is an entity-based approach, it provides a higher level of detail of the behavior of the individuals in the crowd than fluid dynamic models. Likewise, Cell-DEVS models require less computational demands than agent-based models.

Furthermore, Cell-DEVS have many advantages over CA: it provides easy and fast implementation of entity-based crowd modeling (owing to the rule-driven nature of cell behavior definition). The Cell-DEVS formalism solves the problem of unnecessary processing burden in quiescent cells in CA, and it allows for a more efficient asynchronous execution. This results in more efficient and intelligent computation of cell-state variations when compared to CA. The modularity and formal I/O port definitions of Cell-DEVS that are inherited from DEVS formalism allow interfacing with variety of environment models, tools, data sets and visualization mechanisms, both locally and remotely. This makes it easier and more efficient to build complex models, including models with multiple cellular

submodels interfacing others as standard DEVS models (and doing this using CA is cumbersome). Another major difference is that Cell-DEVS defines the cell's timing behavior explicitly, making it easy to build models with varied timing functions according to the needs. This is complex in CA. The advantages of Cell-DEVS over CA are well established and have been discussed extensively in the literature. For further details on the advantages of Cell-DEVS over CA, the reader is referred to the relevant literature.^{8–16}

2. Related work

As mentioned in the previous section, the work on crowd modeling can be categorized into fluid dynamics models, social force models, agent-based models, and CA models. In the following sections, we discuss different recent research on each of these areas.

2.1. Fluid dynamics models

Fluid dynamics models study the crowd as a continuum using coupled non-linear, partial differential equations that can be easily solved in simple geometries. It has been proposed that crowds move in a similar manner as fluid flows. Based on that, early work in this area¹⁷ suggested that the Navier–Stokes equations could be applied to pedestrian flows. However, this does not take into account many factors that affect the crowd behavior such as various physiological, psychological, and social factors. As such, recent models do not use the Navier–Stokes equations in their entirety; rather the concepts of fluid dynamics are combined with consultation from behavioral scientists.

In Hughes,³ the author extended the work in this area, where the crowd is modeled using classical fluid dynamics, but with the additional assumption that human flows “think”. In his work, Bradley depends on the recent sociological view of crowds that non-orchestrated crowds are rational and can therefore be expected to abide by scientific rules of behavior¹⁸. Hence, the non-linear, time-dependent, simultaneous equations representing a crowd are conformably mappable. This property makes many interesting applications analytically tractable. The theory has been used to study the annual Muslim Hajj, in an attempt to improve the flow of pilgrims over the Jamarat Bridge near Mecca. For further and more up-to-date examples and applications in the literature on fluid dynamics models, the reader is referred to the relevant literature.^{19–23} The problem with the fluid dynamics models is that they do not provide an accurate simulation of the crowd, as they do not provide high-resolution details of the behavior of the individuals in the crowd.

2.2. Social-force models

Another method, proposed by Helbing and Molnár,⁴ is called social force. In this method, the human motion is

viewed as a complicated behavior that is subject to “social forces”. A self-driving force and repulsive forces from the environment (other pedestrians and obstacles) affect the motion. The notion of this model can be summarized as the superposition of attractive and repulsive effects determining the behavior of individuals.

Each pedestrian is assumed to be affected by four major factors: the destination to be reached, the distance to be kept from other people, the distance to be kept from borders and other obstacles, and other people or objects that might attract the pedestrian. The completed social force model can be found from these factors, and the paths taken by pedestrians can be predicted.

In Helbing et al.,²⁴ the social force model was extended to create a model that could exhibit numerous phenomena to study the build-up of pressure observed during escape panics. The model considers a mixture of socio-psychological and physical forces influencing the behavior in a crowd. Furthermore, real data and additional video material was used to test the model quantitatively. For further and more up-to-date examples in the literature on social-force models, the reader is referred the relevant literature.^{25–27} Although social force models can be used to reproduce pedestrian behavior in some cases, the underlying assumptions in these models oversimplify how pedestrians behave.

2.3. Agent-based models

In agent-based models, each individual in the crowd is modeled as separate agent that takes its decisions independently. The local phenomena can affect the decision-making process of each individual, whereas the entire crowd can produce an emerging pattern that is deduced by the social and physical aspects of each individual.

Klügl and Rindsfuser,⁵ presented an agent-based simulation of pedestrian traffic of the complete railway station of Bern during rush hour. In their simulations, more than 40,000 agents pass through the station during 1.5 hours. Furthermore, in their simulations, pedestrians are not only capable of avoiding collisions, but are also able to flexibly plan and re-plan their way through the railway station.

Ronald et al.²⁸ investigated the behaviors that pedestrians may exhibit, the different techniques used for pedestrian modeling, and the appropriateness of each technique for particular domains. They studied the belief–desire–intention (BDI) architecture, presenting the development of a sample model using Prometheus, an agent-oriented design methodology, and JACK, an agent-oriented programming language. Further and more up-to-date examples in the literature on agent-based models can be found, for instance, in Pluchino et al.²⁹ and Liu et al.³⁰ Agent-based models usually try to simulate crowd at fine scale, which makes them computationally demanding and more

suitable for short-term simulations with small-sized crowded.

2.4. CA-based models

CA is one of the oldest models of natural computing; it was introduced by John von Neumann in the late 1940s.^{31,32} As mentioned above, in CA, the studied space is represented as a lattice of cells, with each cell being a state machine. The states of a cell come from a finite set of states.³³ Cells change their states synchronously at discrete time steps. The state of a cell at the next time step depends on its current state, and the current states of the neighboring cells (neighborhood) according to an update rule. The neighborhood usually contains some or all the adjacent cells, but more general neighborhoods can be specified.

CAs have been used recently for M&S of pedestrian movement. Burstedde et al.⁶ proposed a two-dimensional CA model to simulate pedestrian traffic, and to simulate the evacuation of a large room with reduced visibility. The model introduced the concept of “floor field”, which can be thought of as a second grid of cells underlying the grid of cells occupied by the pedestrians. Floor field holds the probabilities of moving from a cell to other cells. Dynamic floor cells can evolve with time so that probabilities change with time depending for example on the presence of pedestrians. Hence, floor field is used to model a “long-ranged” attractive interaction between the pedestrians.

The Situated Cellular Automata (SCA) model³⁴ is a particular class of Multilayered Multi-Agent Situated Systems (MMASS). SCA provided an explicit spatial representation, and defined adjacency geometries. It also exploited the concept of autonomous agents by defining an individual state and behavior. Bandini et al.³⁴ used SCA to build a small-sized model to simulate an environment with the crowd as a Multi-Agent Systems (MAS). In Tao and Jun,³⁵ an entity-based model was used to represent bidirectional pedestrian flow using CA. Different behavioral factors were considered such as position exchange and step back. In Ji et al.,³⁶ the authors presented a CA pedestrian model, focusing on acceleration and overtaking. They divided pedestrians into two categories: aggressive and conservative. The model was used to simulate the movement of pedestrians in a corridor.

Masuda et al.³⁷ provide a simple CA model that is able to reproduce oscillation phenomena owing to formation and destabilization of arches in two-dimensional flows. This is used to study the jamming of pedestrian crowds that occurs owing to the formation of arches at bottlenecks. The model predicts the existence of a critical bottleneck size for particle flows without congestion, and it determines the dependency of the jamming probability on the system size. Vihas et al.³⁸ provide a CA model of crowd in which pedestrians follow leaders as this phenomena is a

fundamental driving mechanism. The model provides a microscopic simulation of the crowd as all configurations of the CA model are triggered by simple rules applied locally to each of the group members. The work also study the emergence of qualitative attributes of crowd behavior, such as collective effects, random to coherent motion owing to a common purpose and transition to incoordination (arching) owing to clogging.

Was et al.³⁹ studied the proxemic approach, which is the process of acquisition of space, in evacuation modeling. A new discrete model is proposed in this work. The new model is based on a more detailed representation of space and the idea of floor fields. The presented model allows for efficient, real-time simulation of evacuation from large facilities using more detailed representation of spatial relations.

Some work has been done to reduce the computation cost of pedestrian models. In response to the Hermes project which aims to reduce simulation time of evacuation models by parallelization of the code using parallel computers, Steffen and Chraïbi⁴⁰ attempted to reduce the simulation time by reducing the simulation cost instead of using expansive hardware. The work presents a multicast approach that performs fast simulation of probable evacuation scenarios. The work deals with the problem of passing agents from a CA model to a force-based model. The work also provides a modified CA method that can address the problem at less computational cost, with some possible loss of accuracy.

Cell-DEVS is based on the DEVS^{8,9} a well-known formal discrete-event M&S methodology. A Cell-DEVS model is also an n -dimensional grid of cells. Each cell is defined as a DEVS atomic model, and a procedure to couple cells is defined. Figure 1(a) shows the contents of an atomic cell. A cell is only active when an external event occurs, or when an internal event is scheduled. When there are no further scheduled events, the cell will passivate. When an external event occurs, the external transition function is executed, and the local computing function (τ) is activated. When the cell's state changes, the external function will schedule an internal transition, and the state change is transmitted after a delay of d .

The local computing function in a Cell-DEVS model computes the next state of a cell depending on its current state, and the states of a finite set of nearby cells (called its *neighborhood*). The internal computing function is defined using a set of rules indicating the output VALUE for the cell's state after some time DELAY, when a PRECONDITION is satisfied. The rule format is denoted as $\langle \text{VALUE} \rangle \langle \text{DELAY} \rangle \langle \text{PRECONDITION} \rangle$, which means that when the PRECONDITION is satisfied, the state of the cell will change to the assigned VALUE, and this new value will be transmitted to its neighborhood after a period time of DELAY

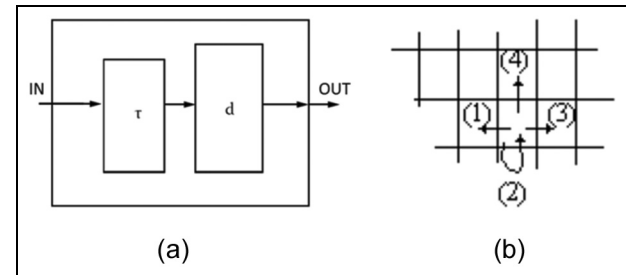


Figure 1. Cell-DEVS model: (a) informal description of an atomic cell; (b) two-dimensional lattice of cells.

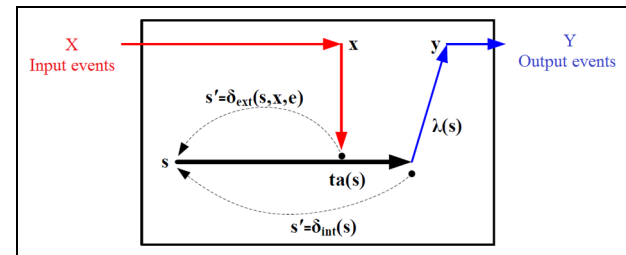


Figure 2. DEVS atomic model semantics.⁸

After a cell is defined, it can be later integrated to a coupled model representing the cell space. We used CD++, which is an M&S tool that provides a development environment for implementing DEVS and Cell-DEVS models. DEVS atomic models can be developed and incorporated into a class hierarchy programmed in C++. Coupled models can be defined using a built-in specification language.⁸

DEVS provides a formal framework for modeling generic dynamic systems and includes hierarchical, modular, and component-oriented structure, and formal specifications for defining structure and behavior of a discrete event model. Created models can be used to build a simulator, which is a device that is able to execute the models instructions and generate its behavior. According to DEVS, a real system can be described as a composition of atomic and coupled components.⁹ The coupled component maintains the hierarchical structure of the system, whereas each atomic component is the basic building block of the system, which represents its behavior.

Figure 2 shows the behavior of an atomic model. An atomic component is in state s for a specified time, $ta(s)$. An external transition, δ_{ext} , is triggered by input to the input port of the atomic model. The external transition takes as its input the current state, s , the time elapsed since the last transition, e , and the external event that has been received, x . This might generate a new state. In contrast to the external transition, the state transition, δ_{int} , (referred to as “internal transition”) is triggered at the end of the

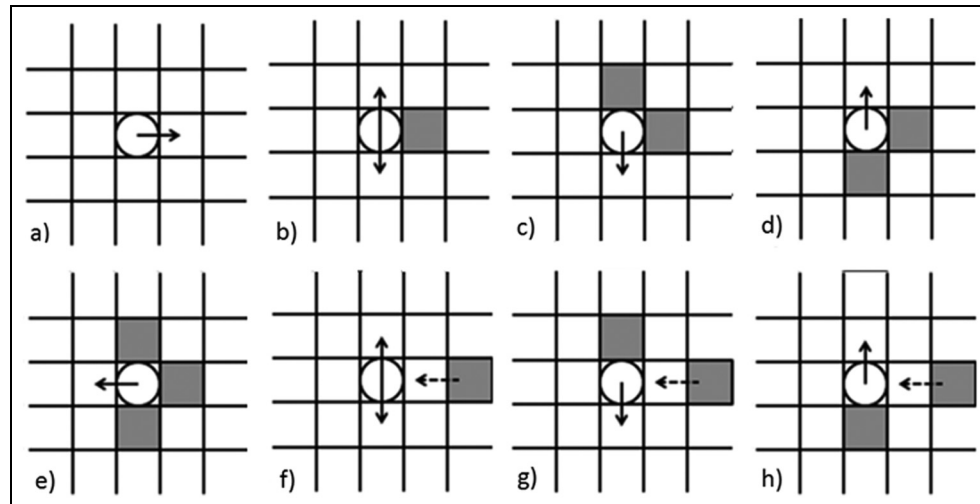


Figure 3. Pedestrian movement: (a) straightforward, (b)–(e) barriers avoidance, (f)–(h) collision avoidance.

time-delay of each state. Figure 2 shows the state transition of an atomic component. If the atomic component passes time of the state without interruption, it will produce an output y at the end of this time and change state based on its δ_{int} function, and continues the same behavior. However, if it receives an input x during its $ta(s)$ time, it changes its state as determined by its δ_{ext} function and does not produce an output (external transition).

A coupled model connects the basic models together in order to form a new model. This model can itself be employed as a component in a larger coupled model. This allows hierarchical construction of complex models.

As previously mentioned, Cell-DEVS is a combination of CA and DEVS with explicit timing delays.⁸ Once we define the behavior of a single cell, we need to form a cell space. The cell space is a coupled model defined as an n -dimensional lattice of atomic Cell-DEVS models. Each cell has a set of neighboring cells, defined by the neighborhood set. A neighborhood can be a subset of the adjacent cells, all the adjacent cells, or might even contain remote cells. When triggered, the local computing function in a Cell-DEVS atomic model computes the next state of a cell depending on its current state and the states of the cells in the neighborhood.

Since Cell-DEVS implement entity-based modeling, it provides a higher level of detail than fluid dynamics models and needs less computation demands than agent-based models.² Furthermore, as discussed in Section 1, Cell-DEVS has multiple advantages over CA, which makes it easier to develop larger models, and allows faster execution of models.

In the following section, we introduce multiple crowd models that were developed using Cell-DEVS. These models simulate the movement of pedestrians in one, two, and three dimensions.

3. Cell-DEVS models of crowd behavior

When modeling pedestrian movement using Cell-DEVS, we divide the space under study into a grid of cells, with each cell representing a limited part of the space under investigation. A set of states is defined for the cells (occupied, vacant, etc.), and the next state of a cell will be generated using a set of rules that takes as input the cell's current state and the states of its neighboring cells.

As such, to build a Cell-DEVS pedestrian model, we need to define the dimensions of the model, the set of possible states for the cells, the neighborhood, and a set of rules that define the behavior of the cells. A model may also contain multiple Cell-DEVS models interconnected in a hierarchical fashion.

There are three main issues to consider when modeling pedestrian movement: direction, collision avoidance, and speed.

- *Moving direction:* The movement of pedestrians in the crowd can be one-directional or multi-directional. Figure 3 shows most of the possible directions a pedestrian (represented by the circle in the middle) might take to get to a destination to the east. The arrow in the core cell indicates the direction of movement in that cell. From Figure 3, we can see that pedestrians might move towards their destinations directly (Figure 3(a)), or indirectly (Figure 3(b)–(h)) to avoid obstacles or other pedestrians.
- *Collision avoidance:* As we assume that each cell accommodates only one pedestrian at a time, collision will occur when multiple pedestrians are vying for a cell. Figure 3(f)–(h) shows this case. The dash

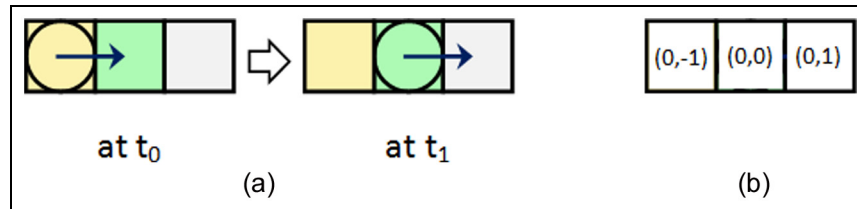


Figure 4. (a) Pedestrian moving forward in one direction. (b) Model's neighborhood.

line indicates the intent of other pedestrians, so the core cell should change direction to avoid potential collision.

- *Adjusting speed:* Pedestrians may have different speeds owing to different time constraints. Pedestrians might also walk at a constant speed or have to change their speed.

In this section, we discuss models for pedestrian movement in one, two, and three dimensions. The models presented in this section are basic models that provide examples and explanation of how Cell-DEVS are used for M&S of crowd. More complex and realistic models will be provided in Section 4. For example, the pedestrians in the models in Section 4 follow pathways based on the shortest distance to the point of interest, and pedestrians in the model in Section 4.1 try to sidestep to a cell that is closer to the point of interest (exit/stairway).

3.1. One-dimensional movement model

Moving straightforwardly is the most fundamental and important aspect of crowds. Pedestrians need to change lanes only when necessary. As mentioned above, we consider that each cell can accommodate a single person. Each cell can be either empty (state 0) or occupied (state 1).

Figure 4(a) shows a person intending to move in one direction (east). Each pedestrian can move only one cell forward at a time. The time delay can be adjusted to simulate certain speed. Figure 4(b) shows the neighborhood for this simple model, which consists of three cells. Cell (0,0) is the core cell.

The rules for this model are simple. If the cell ahead is vacant, the pedestrian moves forward. This can be represented in the following rules.

Rule 1: 1 400 { (0,0)=0 and (0,-1)=1 }
Rule 2: 0 400 { (0,0)=1 and (0,1)=0 }

The first rule states that if the core cell is empty and there is a pedestrian to the west (in cell (0,-1)), the next state of the core cell will be 1. The second rule states that if there is a pedestrian in the core cell, and the cell to the east (cell (0,1)) is vacant, the next state of the core cell will be 0.

3.2. Two-dimensional movement model

In this model, we use more rules to model the movement of pedestrians in multiple directions, and we handle cases of collision avoidance. For example, when two flows of pedestrians walking in opposite directions, the rules governing their movement should consider this and avoid collision.

The space is divided into two-dimensional grid of cells. Each cell can be empty (state 0), occupied by a pedestrian moving east (state 1), occupied by a pedestrian moving west (state 2), or the cell can be representing an obstacle in which case it cannot be occupied by a pedestrian (state 3). We consider pedestrians moving with constant speed.

A pedestrian can move in four possible directions: to the adjacent cell to the north, south, east, or west. Moving east and west are the main directions (states 1 and 2, respectively), whereas moving north and south is used for sidestepping.

In the following, we define the rules that govern the movement of a pedestrian walking to the east (the same logic applies to a pedestrian moving west). We present four possible cases. In each case, we define two rules. As in the model above, the first rule considers the case where the pedestrian leaves the current cell, whereas the second rule considers the cell where the pedestrian moves. The rules are checked sequentially starting from the first rule and continuing until one of the preconditions is satisfied.

1) *No pedestrian/obstacles ahead, move forward:* This is the case where there is no other pedestrian or an obstacle in the cell ahead (0,1), and there is no pedestrian in cell (0, 2) walking in the opposite direction. This case is illustrated in Figure 3(a).

Rule 1: 0 400 { (0,0)=1 and (0,1)=0 and (0,2) !=2 }
Rule 2: 1 400 { (0,0)=0 and (0,1) !=2 and (0,-1)=1 }

The first rule checks if the current cell has a pedestrian walking to the east. In this case, the pedestrian will move forward, i.e., the current cell will be vacant. The second rule checks if the core cell is empty and the cell

to the west has a pedestrian walking to the east, in which case the pedestrian will move to the core cell.

2) *Pedestrian/obstacle ahead; move to your right (south) if possible*: This rule states that if the next cell forward contains a pedestrian or an obstacle, the pedestrian will move to their right (south), given that the cell is vacant and that there is no other pedestrian is going to move into it.

Rule 3: 0 400 { (0,0)=1 and (0,1)!=0 and (1,0)=0 and (1,1)!=2 and (1,-1)!=1 }

Rule 4: 1 400 { (0,0)=0 and (-1,0)=1 and (-1,1)!=0 and (0,1)!=2 and (0,-1)!=1 }

3) *Pedestrian/obstacle ahead and to the right (south), move to your left (north) if possible*: In this case, the cell ahead is occupied by an obstacle or a pedestrian moving in the opposite direction, and the cell to their right (south) is occupied by a pedestrian or an obstacle. In such case, the pedestrian will have to move to the cell to their left (north), given that, it is empty, and that no pedestrian is walking into it. This is the case of Figure 3(c). The rules for this case can be stated as follows.

Rule 5: 0 400 { (0,0)=1 and ((0,1)=2 or (0,1)=3) and (1,0)!=0 and (-1,0)=0 and (-1,1)!=2 and (-1,-1)!=1 }

Rule 6: 1 400 { (0,0)=0 and (1,0)=1 and ((1,1)=2 or (1,1)=3) and (2,0)!=0 and (0,1)!=2 and (0,-1)!=1 }

4) *Two pedestrians are vying for the same cell; move to your right (south) if possible*: When there is pedestrian moving east, and another moving west, and both are moving into the same cell, the one walking to the east will move to the cell to their right (south), given that it is vacant and no other pedestrian is moving into it. Rules 7 and 8 below correspond to this case.

Rule 7: 0 400 { (0,0)=1 and (0,1)=0 and (0,2)=2 and (1,0)=0 and (1,1)!=2 and (1,-1)!=1 }

Rule 8: 1 400 { (0,0)=0 and (-1,0)=1 and (-1,2)=2 and (-1,1)=0 and (0,1)!=2 and (0,-1)!=1 }

5) *Two pedestrians are vying for the same cell and there is a pedestrian or an obstacle to your right (south); move to your left if possible (north)*: When there is pedestrian moving east, and another moving west, and both are moving into the same cell, and it is not possible for the east walker to sidestep to their right (south) owing to Pedestrian/obstacle, the pedestrian will sidestep to the cell to their left (north), given that it is

vacant and no other pedestrian is moving into it. Rules 9 and 10 below correspond to this case.

Rule 9: 0 400 { (0,0)=1 and (0,1)=0 and (0,2)=2 and (1,0)!=0 and (-1,0)=0 and (-1,1)!=2 and (-1,-1)!=1 }

Rule 10: 1 400 { (0,0)=0 and (1,0)=1 and (1,2)=2 and (1,1)=0 and (2,0)!=0 and (0,-1)!=1 and (0,1)!=2 }

When none of the cases above is satisfied, the pedestrian simply does not move. In addition to the main rules above, other rules to control the entrance and exiting of the pedestrians have been implemented.

3.2.1. Pedestrian movement in a walkway. Based on the generic model presented above, here we present a more sophisticated model for two-dimensional movement. We use a Cell-DEVS model that represents a pedestrian walkway, and avoid more cases of collisions.

In the previous two-dimensional model, we have seen that the rules are checked sequentially, and the order of the rules define their priority. In this model, we add more rules to avoid other collision cases that were not considered before: The collisions that might occur when two pedestrians walking in the opposite directions try to sidestep into the same empty cell, i.e., when one of the cells has to step north, whereas the other one has to step south.

An approach that can be used to handle the sidestepping collisions is to assign different priorities for different directions (in the same rule), in addition to having different priorities for different rules. When a sidestepping collision is about to occur, the rule will check the direction-based priorities, authorizing the pedestrian with higher priority to go, while keeping the other one waiting. When two sidestepping pedestrians are about to collide, the following priorities are applied.

- 1) West pedestrian sidestepping to their right (north): This has the highest priority
- 2) East pedestrian sidestepping to their left (north)
- 3) West pedestrian sidestepping to their left (south)
- 4) East pedestrian sidestepping to their right (south)

Furthermore, a rule was added to the model to allow west walkers who are at even rows to step back if the cells ahead, to their left (south), and to their right (north) are all occupied. Figures 5 and 6 show some results obtained from simulations based on this model.

As can be seen in Figure 5, x is a pedestrian walking to the east, whereas y and z are pedestrians walking to the west. The pedestrian x is blocked, as there is someone in the cell ahead, someone else to the south, and there is a wall to the north. Therefore, x does not move. Pedestrian y

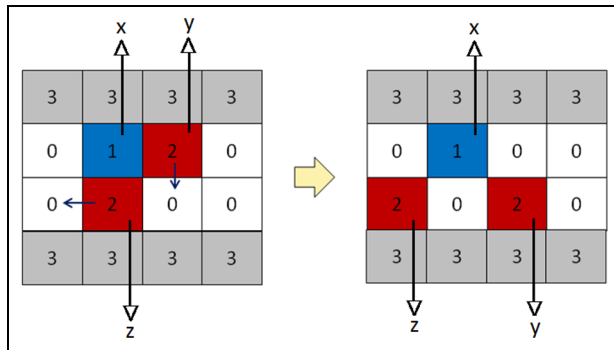


Figure 5. Moving left if blocked.

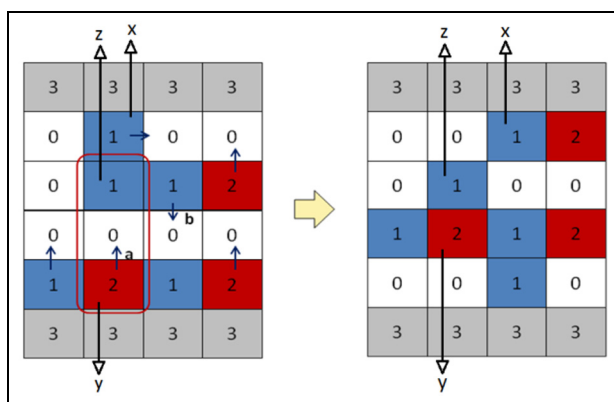


Figure 6. Handling sidestepping collision with movement priorities.

has x in front, and a wall to the north, but the cell to their left (south) is vacant, so y will sidestep to the left (south). Pedestrian z , on the other hand, is walking to the west, the

cell ahead is vacant, and no one is moving into it. So z will move forward.

In Figure 6, x and z are walking to the right, whereas y is walking to the left. Pedestrian x moves forward as the cell ahead is vacant. Both y and z have other pedestrians in front of them, and the cells to their left are not available, so both want to move to the same cell to their right. As the west pedestrian sidestepping to their right (north) has higher priority than the east pedestrian sidestepping to their right (south), y will move to their right (north), and z will stay in their cell.

Figure 7 shows another example in which there is no space for pedestrians to sidestep. At the beginning, the figure shows two pedestrians walking east that meet two pedestrians walking west. As there is no space for pedestrians to sidestep, they will stay in their cells, except the west walker at the bottom, which steps back (as s/he is in an even row). The west walker who stepped back is now vying for the same cell as the corresponding east walker, and hence, will sidestep to the right (north), as can be seen in screenshot (3). As the cell ahead of the west walker who sidestepped is a pedestrian who is walking in the same direction, it will not sidestep to its left (pedestrian sidestep to their left only if the cell ahead is an obstacle or pedestrian walking in the opposite direction, and the cell to their right is occupied). The east walker at the bottom now can move straight to the cell ahead, as can be seen in screenshot (4). In (5), we can see that the east walker that was at the top sidestepped to his right (south). In screenshot (6), pedestrians start moving forward.

3.3. Three-dimensional movement model

Three-dimensional models can be used in crowd modeling to simulate the movement of pedestrians in multi-level

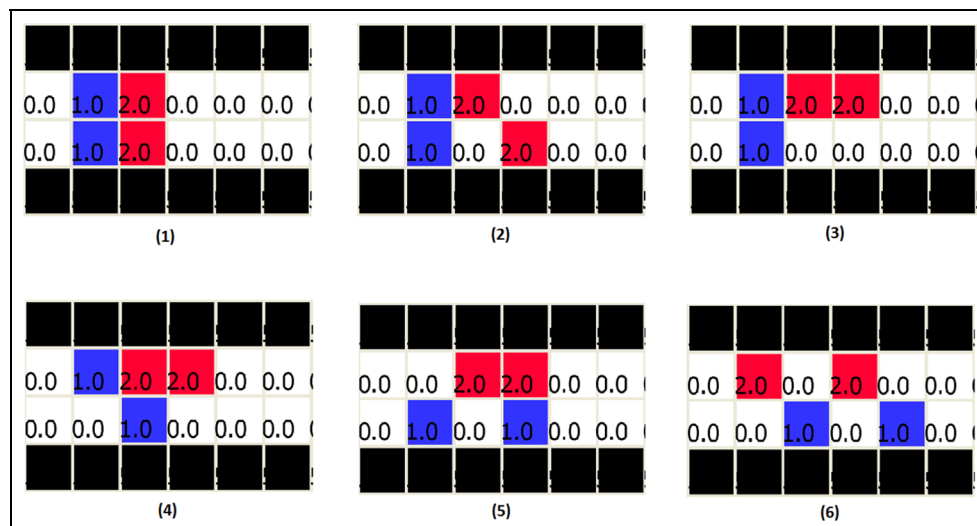


Figure 7. Stepping back.

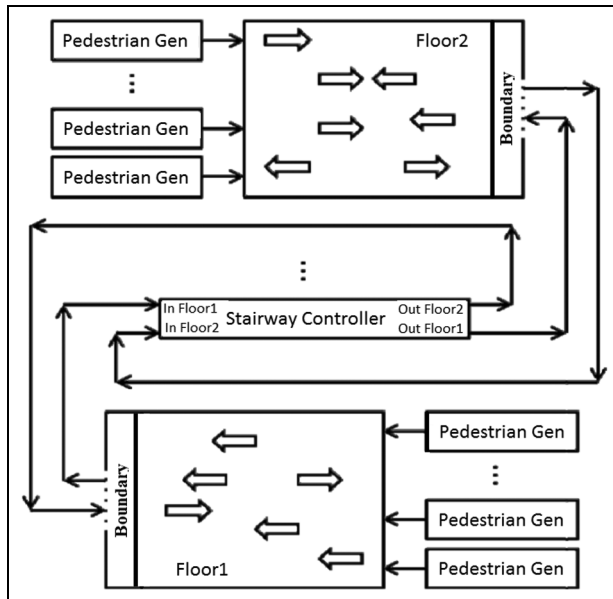


Figure 8. Cell-DEVS coupled model for two-floor building.

buildings. Such models can be very useful in applications such as building design and emergency evacuation.

Three-dimensional movement can be implemented using a three-dimensional cell lattice and a three-dimensional neighborhood, where a cell in the neighborhood can be identified with a 3-tuple, i.e., (x, y, z) . However, owing to the fact that pedestrians usually move from one level to another through certain cells (such as an elevator or a stairwell in a building), the model design can be simplified by building a model, that is composed of multiple two-dimensional Cell-DEVS models. Each one represents a two-dimensional movement in one level. We can then interconnect these models using a coupled model, and design the model and the rules in such a way that allows pedestrians to move between levels. Although this does not simulate full three-dimensional movement that is exhibited for example by birds or fish flocks, it provides a convenient way to model the movement of pedestrians in multilevel buildings.

In this section, we will present a simple model that employs the latter approach. The model represents the movement of pedestrians in a two-floor building. The movement of pedestrians in each level is based on the pedestrian rules from the previous models. In the next section, we will present a more complicated model for three-dimensional movement that employs three-dimensional Cell-DEVS model with a three-dimensional neighborhood, to simulate evacuation of multi-floor building.

Figure 8 shows a coupled model for the movement in a two-floor building. This model is implemented using two different Cell-DEVS components to represent each floor. These two components both use the pedestrian behavior rules in the previous model. In order to simulate the

movement of pedestrians from one floor to the other, a DEVS model (Stairway Controller) is created and interfaced to the Cell-DEVS models. This connects the two Cell-DEVS components together. Furthermore, a new zone is implemented in each floor to get a proper behavior of the pedestrians when they reach the boundary (stairway-edge) of each floor.

The Pedestrian Gen blocks are DEVS atomic random sequence generator components used to generate pedestrians at each floor. Floor1 and Floor2 are Cell-DEVS models, and the stairway controller block represents the stairway controller DEVS atomic components. Two stairway controller DEVS atomic components are used here. Each atomic component has two input ports, and two output ports (one input port and one output port for each floor). At the edge of each floor, there are two door cells (represented by the dashed lines in Figure 8). As pedestrians can enter and exit to both floors, each door cell has both input and output ports. The output port from each door cell is connected to the corresponding input port of one of the stairway controllers. Furthermore, an output port from each stairway controller is connected to an input port of one of the door cells in each floor. Pedestrians leave and enter each floor through the door cells. When the stairway controller receives a message on an input port, it sends the value of the message to the associated output port after a specific time. For example, when port “in Floor1” receives a value of 2, port “out Floor1” will generate a value of 2 after 50 ms. Port “out Floor1” is connected to the input port of one of the door cells in Floor2. This simulates the movement of a pedestrian through the stairway from Floor1 to Floor2. To sum up, the pedestrian generator components in each floor generate pedestrians. Pedestrians move with the same speed to the other side of the floor, and then leave each floor at the door cells, through the stairway to the other floor.

The stairway controller is a DEVS atomic model that we can control its behavior to simulate different scenarios. For example, we could make it move a pedestrian from the first to the second floor and a pedestrian from the second floor to the first floor at the same time. This simulates the case where the stairway can accommodate two pedestrians. To implement this, we can make the stairway controller generate a value at both outputs if it receives a value from both inputs at the same time. Furthermore, by having multiple stairway controllers (each one is associated with a door cell on each floor), we can simulate a stairway that accommodates multiple pedestrians. We also can, to some extent, simulate different speeds, by having different delays for each one of the stairway controllers. We can also implement the stairway controller in such a way that it sends values to the second floor at higher delay than that for values to the first floor, which simulates that pedestrians move up the stairs slower than going down the stairs. Furthermore, the delay of the stairway controller

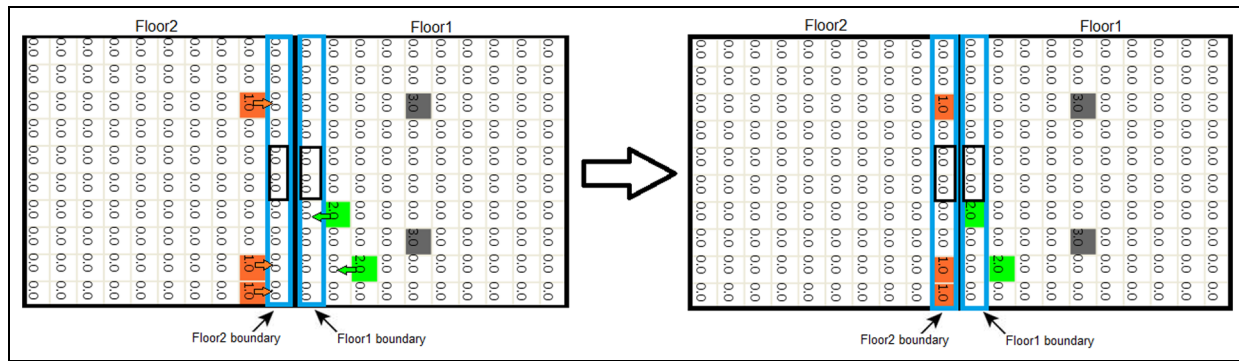


Figure 9. Moving to enter boundary.

can be adjusted in the model depending on the available door cells between the two floors and the initial density of pedestrians in the building.

The rules in the previous models were used to control the movement of pedestrians in each floor. Furthermore, a new zone was implemented at the stairway-edge of each floor (referred to as the floor boundary), and additional rules are defined and used to regulate the behavior of the pedestrians at the boundary of the floor, so that pedestrians will access the stairway.

The rules applied in the boundary zone are listed below. These rules are used for Floor1 (The same logic applies to Floor2). The rules are listed from the highest to the lowest priority. No certain criterion such as social or physiological factors, are considered to set the priority of the rules below.

Rule 1: A west-pedestrian at the door, exit Floor1. This is the case when a pedestrian walking west is at one of the door cells of the first floor. This pedestrian should exit the first floor to go to the second floor through the stairway. To implement the movement of the pedestrian from the first floor to the second floor, the next state of the cell will be 0, and a value of 2 is sent to the door cell output port, which is connected to the input port of the Stairway Controller DEVS model.

Rule 2: Enter boundary: This is when a west-pedestrian is at the cell just before the boundary of Floor1. In this case, the pedestrian will move forward (to the boundary), if the intended cell in the boundary is empty. This rule has the second highest priority.

Rule 3: A west-pedestrian reaches the end of the floor, to the north of the door. This is the case when a pedestrian is at the boundary (the end) of the first floor, but it is in one of the cells to the north of the door cells. In this case, the pedestrian should walk south towards the door cells. To implement this, the pedestrian will move south one cell at a time, if the cell to the south is vacant, and no one else is entering into it.

Rule 4: A west-pedestrian reaches the end of the floor, below the door. This case is similar to rule 3 but the

pedestrian will move north one cell at a time, if the cell to the north is vacant, and no one else is walking into it.

Rule 5: An east-pedestrian entered Floor1. This is the case when an east-pedestrian that is coming from the second floor just entered one of the door cells in the first floor. In this case, the pedestrian will leave the boundary, by moving forward if the cell ahead is empty.

Rule 6: Default: If none of the other rules is valid, the pedestrian waits.

After implementing the model, we show some of the results generated that are related to the door entrance and exit. Figure 9 shows the two levels. The 10×10 cells to the right represent the first level, where the 10×10 cells to the left represent the second level. As shown in Figure 9, the rectangle in each floor represents the boundary of that floor. The small rectangle in the middle of each floor boundary represents the two door cells of that floor. Figure 9 demonstrates that pedestrians walking to the west in the first floor are eventually able to enter the boundary of the floor, and that pedestrians walking to the east in the second floor are able to enter the boundary of the second floor.

Figure 10 demonstrates how a pedestrian in the first floor, who is trying to walk to the second floor, walks into the door in the first floor. Similarly, the figure shows a pedestrian in the second floor, who is trying to walk to the first floor, entering the door cell in the second floor. Figure 10 also demonstrates how the pedestrians leave the boundary cells. The door cells that became occupied will become vacant later to simulate the movement of pedestrians to the other floor. Pedestrians that leave the floor will appear on the other floor.

4. Case study applications

4.1. Emergency evacuation

Emergency planning is an important application and necessary step in building design.⁴¹ It is done for important reasons such as preventing collapse during evacuation or

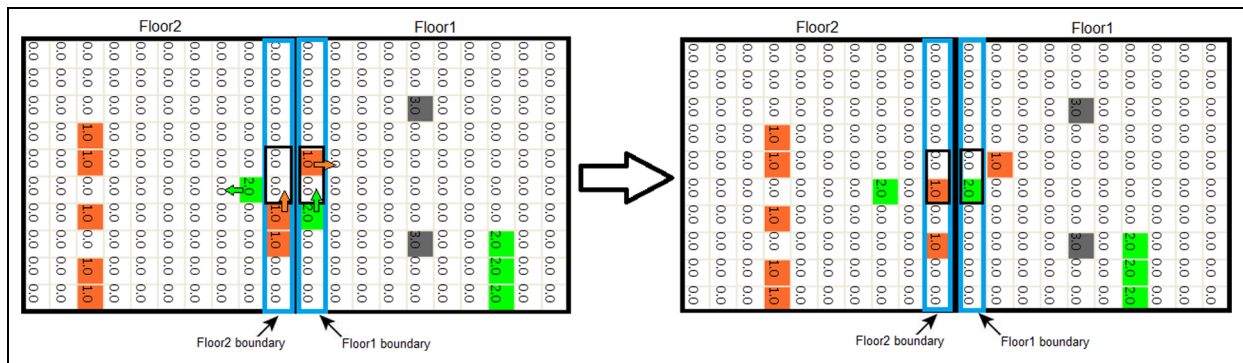


Figure 10. Moving towards door.

reduction of building evacuation time. This can be achieved sometimes through little changes in the building design such as changing the location of stairways or adding sufficient emergency exits to ensure that the building can be evacuated rapidly. In such cases, simulations would be helpful. By creating a virtual version of the building, it is possible to test many different designs to get metrics, such as the evacuation time, and find the best design. In this way, potential problems can be avoided and fixed before construction begins.

In this section, we will present a model of the evacuation in buildings with many floors. We will show how to use the model to test different building designs. We use a three-dimensional Cell-DEVS model to define the behavior in a multi-floor building; a two-dimensional Cell-DEVS model runs on each floor, with the third dimension used to model multiple floors.

We first determine the pathways that people would take to get to the nearest exit, and initialize the model by placing people at random. After this stage, each cell will have a direction of movement based on a shortest-path algorithm. The direction of movement in a cell points directly to the exit or to the next cell on the path to the nearest exit as can be seen in Figure 11. The rules that implement the

initialization of the pathways are discussed later in this section.

Furthermore, people are distributed randomly throughout the building in this stage. Each cell accommodates only one person at a time. The area of a cell is assumed to be 0.4 m^2 . The model can be initialized with different densities of people (saturation). A saturation of 10% means that there is a probability of 0.1 for each cell to be occupied at the start of the simulation.

Once the pathways have been determined, the actual evacuation begins. Each pedestrian will attempt to move through the building one cell at a time based on the direction that is stored in their current cell, as shown in Figure 12.

If the cell that a pedestrian should move to is occupied, then in most cases the pedestrian will try to sidestep to the left or to the right, if one of these cells is available and no one is moving into them. Pedestrians following the pathways are always given higher priority over sidestepping pedestrians. If the pedestrian is blocked and no cells are vacant for them to sidestep to, then the pedestrian waits in their current cell. On each floor, a person moves only in four directions; N, S, E, W (although this could be easily extended). Depending on their floor, people will either move to the nearest exit out of the building or to the nearest staircase down to the next floor. Upon reaching the next floor, they exit the staircase and head to the nearest exit or the next set of stairs down, and so on.

Table 1 shows all the cell states used in this model. As we can see, values from 3 to 10 represent the pathways. Even numbers in that range represent occupied cells, whereas odd numbers represent vacant cells. For example, a cell with a value of 10 means that the cell is occupied and the person will move to the cell to the west when it is available. When the person moves, the state of the cell will change to 9.

The number 2 represents an exit of the building, and when pedestrians enter a cell with that value, they disappear from the simulation and it is assumed they have left

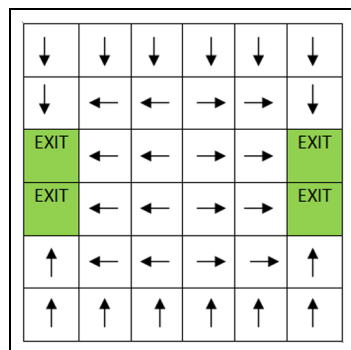


Figure 11. Sample direction of movement in cells.

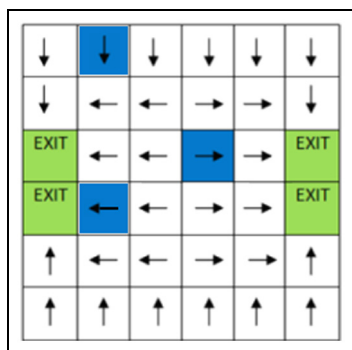


Figure 12. Pedestrian movement.

the building. The numbers from 11 to 14 represent stairways. A set of two stairway cells can only hold two people, one at the top and one at the bottom. If both the top and bottom of the stairway cells are occupied then no one else can enter until an opening appears. A person enters the top of stairwell when the pathway tells it to and when the top of the stairway has the value of 11 (unoccupied). If the bottom of the stairway is vacant, i.e., the cell that is one level below has a value of 13, the person will move down to the bottom of the stairway (one floor below). To exit the stairway, a neighboring cell must have an odd value between 3 and 10, representing an empty pathway.

The local transition function rules can be categorized into three groups: rules that define the pathways of the model and initialize it with people; rules that control the movement of people through the pathways, stairways, and the exit; and rules that control the sidestepping of pedestrians when the next cell on the pathway is occupied. In the following, we provide examples on each of the groups above.

- 1) Rules that define the pathways of the model and initialize it with people.

The first set of rules, shown below, is used to define the pathways in the first floor at the beginning of the simulation, and initialize the cells with people.

```
rule : { if (uniform(0,1) < 0.1, 4, 3) } 0 {
  (0,0,0)=0 and (1,0,0)>1 and (1,0,0)<11
}
rule : { if (uniform(0,1) < 0.1, 8, 7) } 0 {
  (0,0,0)=0 and (-1,0,0)>1 and (-1,0,0)<11
}
rule : { if (uniform(0,1) < 0.1, 10, 9) } 0 {
  (0,0,0)=0 and (0,-1,0)>1 and (0,-1,0)<11
}
rule : { if (uniform(0,1) < 0.1, 6, 5) } 0 {
  (0,0,0)=0 and (0,1,0)>1 and (0,1,0)<11
}
```

When executed, they initialize the cells that are not defined as exit, staircase, or wall cells. The rules initialize each one of these cells with a value from 3 to 10 (based on the values shown in Table 1). The direction of movement in each cell will depend on its location. Each one of these cells will point to the next cell on the shortest path to the exit. For example, the second part of the first rule (between the second group of curly brackets) states that if the cell state is zero (has no direction) and the cell to the south has a direction (value between 1 and 11), the direction of this cell will be south. Whether the cell is vacant or not will be decided randomly with a probability of 0.1 (10% density). For example, the first part of the first rule (between the first group of curly brackets) states that if a uniformly generated random number is less than 0.1, the cell state will be 4 (occupied with south direction), otherwise it will be 3 (vacant with south direction). The rules can also be used to generate pedestrians with any density, by only changing the value 0.1 that appears after the inequality in the rules above.

The following set of four rules are similar to the first set, however, they are used for floors with only stairwells (floor two and above).

```
rule : { if (uniform(0,1) < 0.1, 6, 5) } 0 {
  (0,0,0)=0 and (0,1,0)>10 and (0,1,0)<13
}
rule : { if (uniform(0,1) < 0.1, 10, 9) } 0 {
  (0,0,0)=0 and (0,-1,0)>10 and (0,-1,0)<13
}
```

Table 1. Cell states.^a

Cell state	Cell color	State name	Cell state	Cell color	State name
1		Wall	8		North occupied
2		Exit	9		West
3		South	10		West occupied
4		South occupied	11		Top of stairs
5		East	12		Top occupied
6		East occupied	13		Bottom of stairs
7		North	14		Bottom occupied

^aColor available online.

```
rule : { if (uniform(0,1) < 0.1, 8, 7) } 0 {
(0,0,0)=0 and (-1,0,0)>10 and (-1,0,0)<13 }
rule : { if (uniform(0,1) < 0.1, 4, 3) } 0 {
(0,0,0)=0 and (1,0,0)>10 and (1,0,0)<13
}
```

2) Rules that control the movement of people through the pathways, stairways, and the exit.

The movement of people through the pathways, stairways, and the exits is performed using many sets of rules. As examples, we will present the rules that move people between cells within the same floor. The following set of rules are used to vacate the cell the pedestrian is about to leave.

```
rule : 9 {Delay} {(0,0,0) = 10 and odd
((0,-1,0)) }
rule : 5 {Delay} {(0,0,0) = 6 and odd
((0,1,0)) and (0,2,0) != 10}
rule : 3 {Delay} {(0,0,0) = 4 and odd
((1,0,0)) and (1,-1,0) != 6 and (1,1,0) !=
10 and (1,1,0) != 14 and (1,-1,0) != 14 }
rule : 7 {Delay} {(0,0,0) = 8 and odd((-1,0,0))
and (-2,0,0) != 4 and (-1,1,0) != 10 and (-1,-1,0) != 6
and (-1,1,0) != 14 and (-1,-1,0) != 14 }
```

The first rule states that if the current value of the cell is 10 (occupied with a pedestrian moving west) and the cell to the west is vacant, then the next state of this cell will be 9. Note how the following rules in the set control the priority of movement, for example, the second rule controls movement to the east, and it makes sure no pedestrian moving west is vying for the same cell, which gives west movement higher priority. Variable delays are used in this model, and complex rules are used to control the delays of movement rules, which models movement with variable speeds. The rules control the movement depending on factors. For instance, the delay for horizontal movement could decrease as the cells get closer to the exit. This simulates acceleration of pedestrians as they are moving towards the exit. Multiple cell delays were used. These delays are in the range 277 ms to 425 ms to model unimpeded free walking speeds in the range 1.19 ± 0.25 m/s. These ranges around the speed of 1.19 m/s which is the average walking speed in evacuation found by Gwynne and Rosenbaum⁴² and adopted by the Society of Fire Protection Engineering. Again this simulates a free walking speed that does not include the time during which a pedestrian is staying at their cell when there is no other cell available to move to. When considering the time during which pedestrians wait at their cells, the average speed of pedestrians considerably decreases as the density of pedestrians decreases. This is

validated and compared to empirical results as can be seen in the Appendix.

The following set of rules control the movement into a cell in the same floor.

```
rule : {(0,0,0)+1} {Delay} {(0,0,0)=3
or (0,0,0)=5 or (0,0,0)=7 or (0,0,0)=9)
and (0,1,0) = 10 }
rule : {(0,0,0)+1} {Delay} {(0,0,0)=3
or (0,0,0)=5 or (0,0,0)=7 or (0,0,0)=9)
and (0,-1,0)=6 and (0,1,0) != 10 }
rule : {(0,0,0)+1} {Delay} {(0,0,0)=3
or (0,0,0)=5 or (0,0,0)=7 or (0,0,0)=9)
and (-1,0,0) = 4 and (0,1,0) != 10 and (0,-1,0) != 6 }
rule : {(0,0,0)+1} {Delay} {(0,0,0)=3
or (0,0,0)=5 or (0,0,0)=7 or (0,0,0)=9)
and (1,0,0) = 8 and (0,1,0) != 10 and (0,-1,0) != 6
and (-1,0,0) != 4 }
```

The first rule states that if the core cell is a vacant pathway (3, 5, 7, or 9), and the cell to the east is 10 (occupied with a person moving west into the core cell), the next state of the core cell will be occupied pathway with the same direction (4, 6, 8, or 10, respectively). The other 3 rules are the similar but with pedestrians coming from other directions. As mentioned above, other rules are used in this group to control the movement of pedestrians through the stairways and the exits.

3) Rules that control the sidestepping of pedestrians when the next cell on the path way is occupied.

When the next cell on the pathway is not available, the pedestrian will try to sidestep to the left or to the right. This is important and it models a more natural behavior of pedestrians. The following set of rules provides an example of sidestepping rules.

```
rule : 9 {Delay} {(0,0,0) = 10 and #Macro
(WestBlocked) and #Macro(SouthBlocked)
and #Macro(NorthVacant) and (-2,0,0) !=
4 and (-1,1,0) != 10 and (-2,0,0) != 10 and
(-1,-1,0) != 6 and (-1,1,0) != 4 and (-1,-1,0) != 4
and (-1,1,0) != 14 and (-1,-1,0) != 14 }
rule : {(0,0,0)+1} {Delay} {#Macro
(VacantPathway) and (0,1,0) != 10 and (-1,0,0) != 4
and (-1,0,0) != 10 and (1,0,0) = 10 and (0,-1,0) != 6
and #Macro(10SemiBlocked) and (0,1,0) != 4 and (0,-1,0) != 4
and (0,-1,0) != 14 and (0,1,0) != 14 }
```

The first rule checks if the current state of the cell is 10 (pedestrian moving west), and the next cell on the pathway is blocked, the cell to the south is blocked, and the cell to

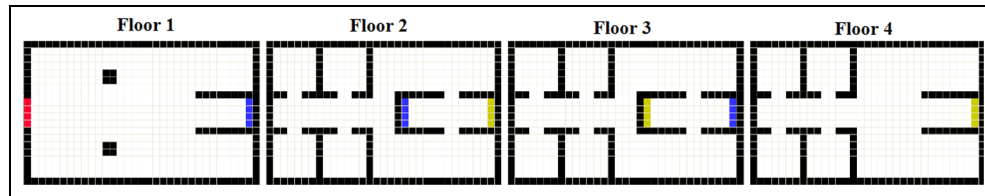


Figure 13. First design of the four-floor building.

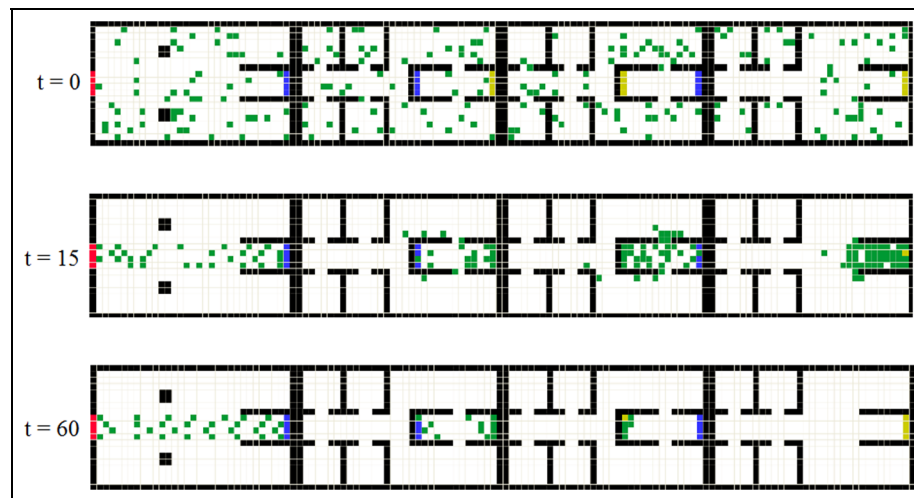


Figure 14. First design evacuation at different simulation times.

the north is vacant, and no one else is moving or sidestepping to the cell, then the next state of the cell will be 9. The second rule states that if the core cell is a vacant pathway (3, 5, 7, or 9), and that the cell to the south is a blocked west walker, and no one else other than that walker is moving into the core cell, then the next state of the core cell will be occupied pathway with the same direction of movement (4, 6, 8, or 10, respectively). Note that macros are used in the rules above to reuse components and help manage the lengthy rules. Note also that variable delays are also used with sidestepping movement.

We performed multiple tests for the verification and validation of this model. The verification and validation of this model is available in the appendix. These tests are suggested for evacuation models by Ronchi et al.⁴³ and the International Maritime Organization (IMO) Guidelines.⁴⁴ In the following, we will use our model to study the evacuation times for different designs of a four-floor building. The building will be modeled with $20 \times 33 \times 4$ Cell-DEVS model.

Figure 13 shows the first design of a commercial four-floor building. Each floor is modeled with 20×33 cell area. The 20×33 cell rectangles from the left to the right represent the first, second, third, and fourth floors, respectively. As shown in Figure 13, this building has an exit in

the middle of the west side of the first floor, and 1 stairway in the middle of the east side of the building. Simulations showed that it takes 91 seconds to evacuate this building when initialized with pedestrians with 10% density.

Figure 14 shows screenshots from the simulations of the evacuation, at times $t = 0$ seconds, $t = 15$ seconds, and $t = 60$ seconds, respectively. As can be seen from the figure, at the beginning of the simulation, the building is initialized with 10% density and pedestrians are scattered throughout the building. At $t = 15$ seconds, we can see that the pedestrians are moving towards the exit and the stairway. At $t = 60$ seconds, the figure shows that only few pedestrians left at the upper floors and all pedestrians in the first floor are moving towards the exit.

Figure 15 shows the second design of the building. This design has 1 exit at the same location as in the previous design, and 2 stairways at the east side of the building instead of 1 stairway. The stairways are at the edges of the east side of the building.

Simulations showed that it takes 63 seconds to evacuate the building with the second design with 10% density. This is shorter than the evacuation time of the first design. There are a couple of deficiencies that can be noticed about the second design. First, although another stairway was added, there is still 1 exit available. Second,

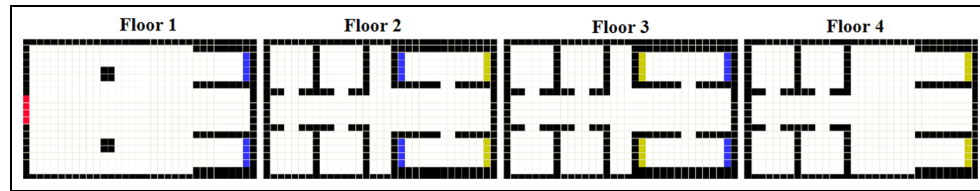


Figure 15. Second design of the four-floor building.

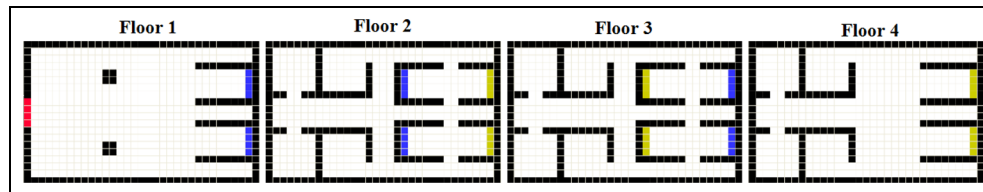


Figure 16. Third design of the four-floor building.

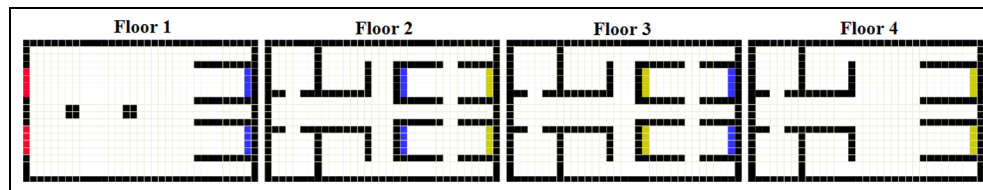


Figure 17. Fourth design of the four-floor building.

pedestrians can only access each stairway from one side as the other side of the stairway is against the wall. Furthermore, the stairways in this design are at the edges of the east side of the building (as opposed to the middle as in the first design), which means that all pedestrians leaving the stairways at the first floor will have to walk slightly longer to get to the exit. Moreover, pedestrians leaving the offices at the upper floors will need to walk slightly longer to get to the stairways. To improve this design, the stairways are pushed slightly towards the middle of the east side of the building, and the location of the doors for the eastern rooms in each floor was changed to be closer to the stairways, as can be seen in Figure 16. By enabling pedestrians to access each stairway from two sides, and changing the doors locations for some offices, many pedestrians will be able to access stairways faster. Furthermore, pedestrians leaving the stairways at the first floor will need to walk shorter distance to get to the exit. Simulations showed that it takes 54 seconds to evacuate this building, which is less than that of the previous designs.

To further speed up the evacuation time, one more design was tested, which can be seen in Figure 17. As this figure shows, this design has two exits at the west side of the first floor, and two stairways at the east side of the building. Simulations showed that it takes 53 seconds to

evacuate this building, which is slightly less than that of the previous design. We can notice that the fourth design did not provide much improvement over the third design. This is because the building is not heavily populated. Simulations results with higher building occupancy showed that the last design improves evacuation time significantly over the other designs, as can be seen in Figure 18.

Figure 18 shows evacuation time versus building occupancy for the building designs presented above. The last design always has the least evacuation time. Furthermore, we can see that the benefit of having an additional exit increases by increasing the initial building occupancy. We can also notice from Figure 18 that the improvement by designs 2 and 3 (additional stairway) over design 1 decreases with increasing the building occupancy. In fact, it can be seen that after 20% occupancy, designs 2 and 3 have longer evacuation time than design 1. The reason is that with high initial building occupancy, adding a staircase while having one exit will cause the pedestrian to get faster to the first floor. This causes cluttering of people which slows down the evacuation process. This is referred to as the “faster-is-slower effect”,⁴⁵ caused by impatience. Since clogging is connected with delays, pedestrians that move faster could eventually cause a smaller average speed of leaving.

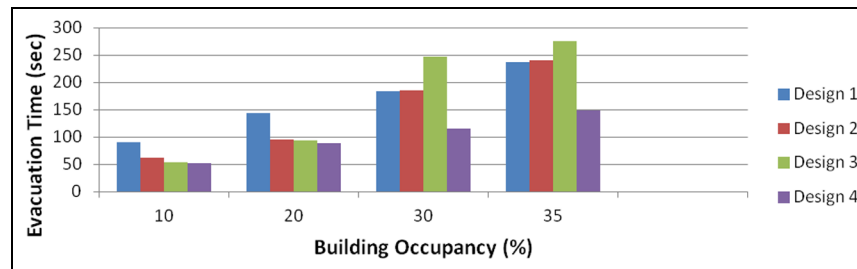


Figure 18. Evacuation time vs building occupancy for multiple building designs.

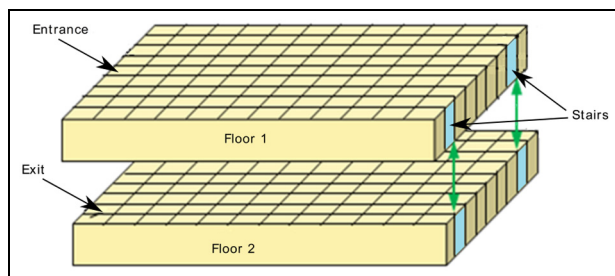


Figure 19. The building used in the occupancy model.

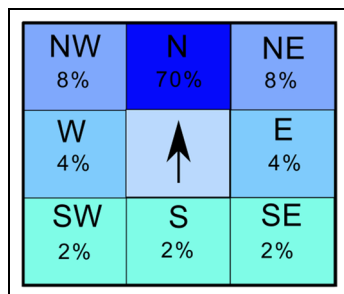


Figure 20. Possible intent probability distributions with pathway value of 6 (North).

Such results can be used to determine the potential maximum allowable occupancy given a safe evacuation time constraint. The results above show that the proposed model is capable of recreating evacuation events and testing different building designs.

4.2. Occupancy analysis

As another application, we present another three-dimensional Cell-DEVS model.⁴⁶ The objective of this model is to study the occupancy level of a multi-dimensional transportation/building, taking into consideration random movement in eight directions, and random movement delays for each pedestrian.

This model has the following new features over the previous models:

- People move with different direction probabilities.
- People wait randomly in their cells before moving to the next cell.
- People move in more than 4 directions within each floor.






















As a case study, we consider the Copenhagen Zoo New Elephant House, where people can move randomly in different directions and wait randomly when visiting in the two floors of this building. A diagram that illustrates the building used in this model is shown in Figure 19. CD++ version 3.0 is used to implement this model, which supports state variables and multi ports for each cell. Each cell can have different variables, and communicates with its neighborhood via multi ports. There are two floors that are connected with staircases. Each cell has five state variables: *Movement*, *Phase*, *Pathway*, *Layout*, and *Hotzone*.

People walk in from the main entrance on floor1, go downstairs to floor2 and then leave the house through the exit. Each floor has pathways for pedestrians to follow. The cell size used is 0.4 m². To implement random movement, pedestrians tend to follow the pathways with high probability, but they can also move randomly to other directions. Figure 20 illustrates an example of potential probabilities in the eight directions when the pathway points north. The darker the cell, the more likely a pedestrian will move into it. Random waiting time is also used at each cell, i.e., a pedestrian will wait for a random delay before moving into the next cell. This models pedestrians moving with different speeds. The higher the value of the HotSpot of a cell, the longer pedestrians tend to stay in that cell.

As mentioned above, each cell has five state variables: Movement, Phase, Pathway, Layout, and Hotzone. Table 2 lists all cell states.

- *Phase*: each movement cycle has four phases (Intent, Grant, Wait, and Move). Details about each phase are provided later.
- *Movement*: 0 means the cell is unoccupied, 1 means it is occupied, and other values are also used to keep track of other states related to the four phases.

Table 2. Cell states.^a

Cell state	State name	Color	Cell state	State name	Color
Variable 1: movement			Variable 2: layout		
0	Not occupied		0	Space	
1	Occupied		2	Wall	
10-18	Intent direction		3	Entrance	
20-28	Grant: intent cell		4	Exit	
40-48	Grant: target cell		3.1	Stair(Floor1)	
30-38	Wait: can move		3.2	Stair(Floor2)	
39	Wait: cannot move				
Variable 3: pathway			Variable 4: hot zone		
5	East		0	No wait	
6	North		1	Wait 0-1 cycle	
7	West		2	Wait 0-2 cycles	
8	South		3	Wait 0-3 cycles	
Variable 5: phase					
1	Intent		2	Grant	
3	Wait		4	Move	

^aColor available online.

- *Pathway*: as in the previous model, pathways are used to guide pedestrians. In this model, pedestrians tend to follow the pathways with high probabilities, but they could also move in other directions with certain probabilities.
- *Layout*: represents the type of the cell (wall, entrance, exit, etc.)
- *Hotzone*: reflects popularity levels of the spots. It reflects the potential of a person staying in the cell.

In order to achieve random movement direction and random delay, the movement cycle has four phases; *intent*, *grant*, *wait* and *move*. They occur sequentially in that order every 4 seconds. In occupied cells, pedestrians choose a direction at random during the *intent* phase. If the target cell accepts this pedestrian, the cell changes to *get grant*; otherwise, it turns to *get rejected*. If granted, the pedestrian would *wait* for some random time according to the *Hotzone* variable, and then the pedestrian will vacate the cell at the *move* phase. If rejected, pedestrian would keep waiting, and go through the cycle again.

In the following, we discuss in more detail the different phases of the movement cycle.

- Rules in phase 1 (Intent):

During this phase, the intent direction is determined by two factors: the pathway direction and the direction probabilities. The pathway direction variable has four values; N, S, E, and W. Depending on the pathway direction, a certain probability distribution will be chosen. As an example,

the probability distribution associated with a pathway pointing north is shown in Figure 20. After choosing a distribution, a random number will be generated, and the intended direction will be chosen depending on where the random number falls.

This is implemented with the following rule.

```
rule : {~movement := uniform(0,1);
~phase := 1.1;} 0 (0,0,0)~phase = 1 and (0,
0,0)~movement=1 and (0,0,0)~pathway>=5}
...
rule : {~movement := 11; ~phase := 2;} 0
(0,0,0)~phase = 1.1 and (0,0,0)~pathway = 5
and (0,0,0)~movement > 0.0 and (0,0,0)~
movement <= #Macro(Front) } ...
```

- Rules in phase 2 (Grant):

Extended neighborhood was used in the previous models to avoid collisions between pedestrians. Using extended neighborhood needs more computation demands. In this model, another method is used to avoid collisions. This model uses the *Grant* phase. In this phase, the target cell chooses which one of the vying pedestrians will move into it. When a vacant cell is in the *Grant* phase, it will check how many neighbors are intended to move into the vacant cells. If there is more than one, the vacant cell will choose only one of them to move into it. Depending on which neighbor it has selected, the vacant cell changes its state. The neighbor that is granted the permission to move into the vacant cell will recognize itself from the current state of the vacant cell.

This is implemented with the following rules.

```
rule : {~movement := 41; ~phase := 4;} 1 {
(0,0,0)~movement = 0 and (0,-1,0)~movement
= 11 and $layout != 2} ...
rule : {~movement := ((0,0,0)~movement+
10); ~phase := 3;} 1 { (0,0,0)~movement >=
10 and (0,0,0)~movement <= 18 }
```

- Rules in phase 3 (Wait):

We use this phase to realize different speed of people by random waiting delay. If a person is granted by the target cell, he will stay there for a random amount of time. The range from which the random waiting time is generated increases by increasing the value of the *Hotzone* variable. This simulates the fact that visitors may stay in different cells for different periods of time (to see something interesting). This is implemented with the following rules.

```
rule : {~movement := 31; ~phase := 4;} { 1
+ 4*randInt($hotzone) } { (0,0,0)~phase = 3
and (0,0,0)~movement = 21 and (0,1,0)~
movement = 41 } ...
rule : {~movement := 39; ~phase := 4;} 1 {
(0,0,0)~phase = 3 and (0,0,0)~move-
ment >= 20 and ... }
```

- Rules in phase 4 (Move):

At this phase, the granted pedestrian can move to the target cell. To finish the movement for this cycle, the *movement* of the cell who has a granted pedestrian will become zero, and that of cells with rejected pedestrian will be 1. For the target cell, it changes to 1. Here are the rules for this part:

```
rule : {~movement := 0; ~phase := 1;} 1
{ (0,0,0)~phase=4 and (0,0,0)~movement=30
and (0,0,1)~movement=40 } ...
```

```
rule : {~movement := 1; ~phase := 1;} 1
{ (0,0,0)~phase=4 and (0,0,0)~movement=48
and (-1,-1,0)~movement=38 }
```

We show the simulation results of a $10 \times 22 \times 2$ Cell-DEVS used to model the two-floor Elephant House. The duration of the simulation was 10 minutes, which has approximately 150 cycles of movements (each cycle takes 4 s). Different building design and simulation properties (doors location, staircases number, arrival rate, moving probabilities) were tested in the simulations.

Figure 21 shows our initial test, which simulates the basic behavior of visitors during the rush hours. The house has four entrances, two floors, and two stairwells. The rate at which people are coming is one person/cycle on average. In Figure 21, the blue cells represent pedestrians, light blue cells are waiting visitors, and each green cell represents the cell where a waiting visitor wants to go. At 10 minutes, 35 cells are occupied out of 90 vacant cells in Floor1, and 24 cells are occupied out of 152 vacant cells in Floor2. So the occupancy levels are 38.9% for Floor1 and 15.8% for Floor2, respectively.

We ran a varied number of simulations after making some changes in terms of door location, number of staircase cells, and other parameters, to study the impact of such changes on the occupancy levels. The results of all tests can be seen in Figure 22.

First, we separated the four-cell gate into two two-cell gates. At the end of the 10-minute interval, the occupancy levels are 36.7% for Floor1 and 17.1% for Floor2. These only slightly differ from the results obtained from the first

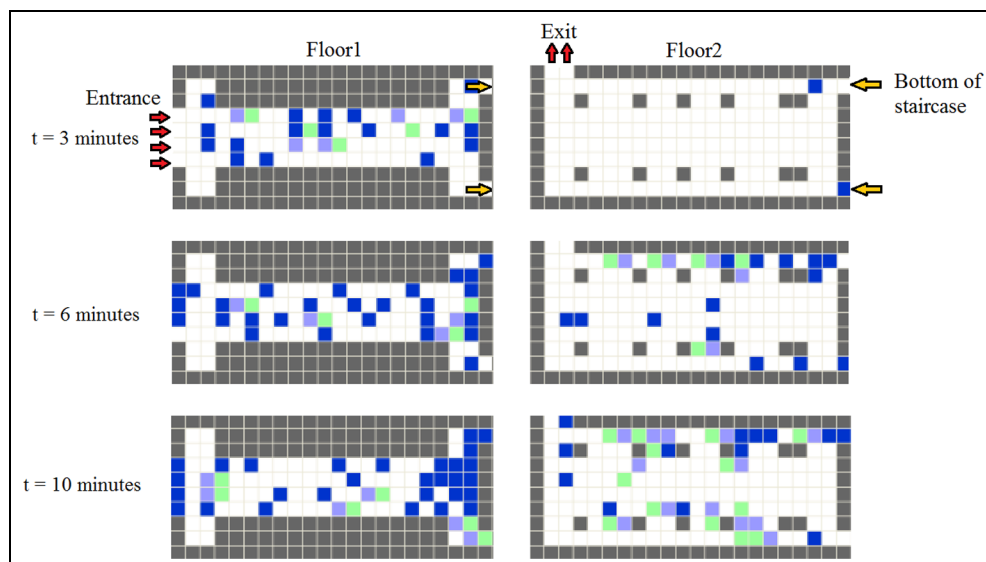


Figure 21. Screenshots from the simulations of the occupancy model.

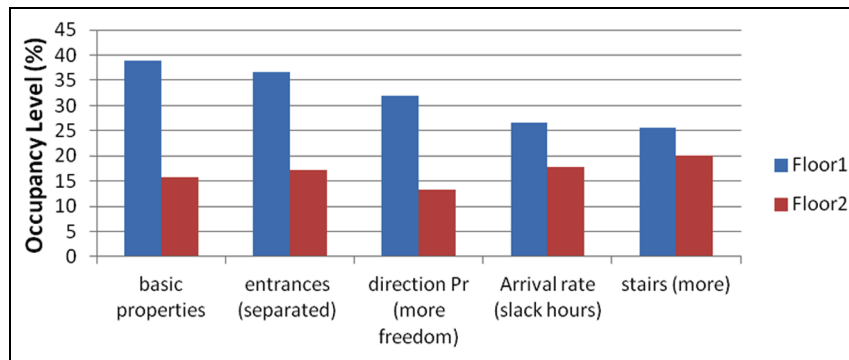


Figure 22. Simulation results of occupancy model.

design above. As it can be noticed, the occupancy of Floor1 is still about twice that of Floor2. We can conclude from this that doors location did not have much impact on the occupancy as doors still generate people at similar rates.

We ran simulations after modifying the first design by extending the staircases between the two floors. Each staircase was extended by one cell (two staircase cells were added at each floor). At the end of the 10-minute period, the occupancy levels are 25.6% for Floor1 and 20% for Floor2. As can be noticed, extending the staircases brought the occupancy rates of the two floors much closer to each other. This is because doing so will eliminate congestions that might happen at the top of the staircases, and get the pedestrians to move faster down to the second floor.

Other tests were performed by changing other parameters (movement direction probabilities and arrival rate) in an attempt to find the impact of such parameters and see which one is the most significant. This can be used by designers to improve the maintenance and management.

In the first test, pedestrians were set to move forward at the rate of 70%. This was changed to 50% to give visitors more freedom moving to other directions. The results show that this did not have significant impact on the occupancy levels. This is probably because people still can reach the stairs/exit at similar rates.

The first test was under hot hours. A new test was performed with lower arrival rates to mimic slower hours. We prolonged the interval between two incoming visitors. The results showed significant decrease of Floor 1 (from 38.9% to 26.7%). These results indicate that pedestrians' arrival rates have obviously a significant effect on occupancy levels. There was only slight difference in the occupancy of Floor2. This is probably because the rate at which people are coming to Floor 2 stays at a steady full rate (but congestion at Floor1 has decreased).

5. Conclusion

Modeling and analysis of crowd behavior is an area of increasing interest owing to its importance in many applications. In many situations, it could be very difficult or

costly to study the behavior of crowd by real-life experiments, which makes M&S of crowds very useful in these situations. In this work, we propose an entity-based approach for M&S of crowd using Cell-DEVS, to overcome the shortcomings of the existing models. We propose Cell-DEVS entity-based models for M&S of one-, two-, and three-dimensional movement of crowds. Furthermore, we extend the models above, and propose two more advanced models as real-life applications. The first application is for modeling of evacuation of multiple-floor buildings. The second model is used to analyze the occupancy levels of multiple-floor buildings. Obtained results show that the Cell-DEVS models are very suitable for entity-based simulation of the crowd. Simulation results obtained from the study case models above show the capability of the proposed models for recreating pedestrians' behavior and providing architects with important input that can be useful in the design process.

Acknowledgements

The authors thank Sixuan Wang, Ronnie Farrell, Mohammad Moallemi, Wang Xiang, and Michael Van Schyndel who developed various models presented in this paper.

Funding

This research has been partially funded by NSERC.

References

1. Zhan B, Monekosso DN, Remagnino P, et al. Crowd analysis: a survey. *Mach Vision Applic* 2008; 19: 345–357.
2. Zhou S, Chen D, Cai W, et al. Crowd modeling and simulation technologies. *ACM Trans Model Comput Simul* 2010; 20: 1–35.
3. Hughes RL. The flow of human crowds. *Annu Rev Fluid Mech* 2003; 35: 169–182.
4. Helbing D and Molnár P. Social force model for pedestrian dynamics. *Phys Rev E* 1995; 51: 4282–4286.
5. Klügl F and Rindsfuser G. Large-scale agent-based pedestrian simulation. *Multiagent System Technol* 2007; 4687: 145–156.

6. Burstedde C, Klauck K, Schadschneider A, et al. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Phys A: Stat Mech Applic* 2001; 295: 507–525.
7. Wolfram S. *Theory and applications of cellular automata*. Singapore: World Scientific, 1986.
8. Wainer G. *Discrete-event modeling and simulation: a practitioner's approach*. Boca Raton: CRC/Taylor & Francis Group, 2009.
9. Zeigler B, Praehofer H and Kim T. *Theory of modeling and simulation*. San Diego, CA: Academic Press, 2000.
10. Wainer G. Improved cellular models with parallel Cell-DEVS. *Int Trans Soc Comput Simul* 2000; 17: 73–88.
11. Wainer G and Giambiasi N. Application of the Cell-DEVS paradigm for cell spaces modeling and simulation. *Simulation* 2001; 71: 22–39.
12. Wainer G and Giambiasi N. Cell-DEVS/GDEVs for complex continuous systems. *Simulation* 2005; 81: 137–151.
13. Wainer G. Applying Cell-DEVS methodology for modeling the environment. *Simulation* 2006; 82: 635–660.
14. Liu Q and Wainer G. Parallel environment for DEVS and Cell-DEVS models. *Simulation* 2007; 83: 449–471.
15. Wainer G and Castro R. A survey on the application of the Cell-DEVS formalism. *J Cell Automat* 2010; 5: 509–524.
16. Wang S and Wainer G. A simulation as a service methodology with application for crowd modeling, simulation and visualization. *Simulation* 2014; 91: 71–95.
17. Bradley GE. A proposed mathematical model for computer prediction of crowd movements and their associated risks. In: *Proceedings of the international conference on engineering for crowd safety*, London, UK, March 17–18, 1993.
18. McPhail C. *The myth of the madding crowd*. New York, NY: A. de Gruyter, 1991.
19. Treuille A, Cooper S and Popović Z. Continuum crowds. In: *Proceedings of the 33rd international conference on computer graphics and interactive techniques*, Boston, Massachusetts, July 30–August 3, 2006.
20. Xia Y, Wong SC and Shu CW. Dynamic continuum pedestrian flow model with memory effect. *Phys Rev E* 2009; 79: 1–20.
21. Dogbe C. On the Cauchy problem for macroscopic model of pedestrian flows. *J Math Anal Applic* 2010; 372: 77–85.
22. Kisko TM, Francis RL and Nobel CR. *EVACNET4 user's guide*. University of Florida, 1998.
23. Zhang WM, Huang L and Wang B. Application of the EVACNET4 model to evacuation in high-rise building. *Fire Sci Technol* 2009; 11.
24. Helbing D, Farkas I and Vicsek T. Simulating dynamical features of escape panic. *Letters to Nature* 2000; 407: 487–490.
25. Hoogendoorn S and Bovy P. Pedestrian route-choice and activity scheduling theory and models. *Transp Res Part B: Methodological* 2004; 38: 169–190.
26. Campanella MC, Hoogendoorn SP and Daamen W. Improving the Nomad microscopic walker model. In: *12th IFAC symposium on control in transportation systems*, Fairfield, NJ, September 2–4, 2009.
27. Yuen J and Lee E. The effect of overtaking behavior on uni-directional pedestrian flow. *Safety Sci* 2012; 50: 1704–1714.
28. Ronald N, Sterling L and Kirley M. An agent-based approach to modelling pedestrian behaviour. *Int J Simul Systems, Sci Technol* 2007; 8: 25–38.
29. Pluchino A, Garofalo C, Inturri G, et al. Agent-based simulation of pedestrian behaviour in closed spaces: a museum case study. *J Artif Soc Social Simul* 2014; 17: 1–14.
30. Liu S, Lo S, Ma J, et al. An agent-based microscopic pedestrian flow simulation model for pedestrian traffic problems. *IEEE Trans Intell Transp Systems* 2014; 15: 992–1001.
31. Von Neumann J and Burks A. *Theory of self-reproducing automata*. Urbana IL: University of Illinois Press, 1966.
32. Burks WA. *Essays on cellular automata*. Urbana, IL: University of Illinois Press, 1971.
33. Kari J. Theory of cellular automata: A survey. *Theoret Comput Sci* 2005; 334: 3–33.
34. Bandini S, Manzoni S and Vizzari G. Crowd modeling and simulation: the role of multi-agent simulation in design support systems. In: *Innovations in design and decision support systems in architecture and urban planning*. Dordrecht: Springer, 2006, 105–120.
35. Tao W and Jun C. An improved cellular automaton model for urban walkway bi-directional pedestrian flow. In: *Measuring technology and mechatronics automation international conference*, Zhangjiajie, Hunan, April 11–12, 2009.
36. Ji X, Zhou X and Ran B. A cell-based study on pedestrian acceleration and overtaking in a transfer station corridor. *Phys A: Stat Mech Appl* 2013; 392: 1828–1839.
37. Masuda T, Nishinari K and Schadschneider A. Cellular automaton approach to arching in two-dimensional granular media. *Lect Notes Comput Sci* 2014; 8751: 310–319.
38. Vihas C, Georgoudas I and Sirakoulis G. Cellular automata incorporating follow-the-leader principles to model crowd dynamics. *J Cellular Automat* 2013; 8: 333–346.
39. Was J, Lubas R and Mysliwiec W. Proxemics in discrete simulation of evacuation. *Lect Notes Comput Sci* 2012; 7495: 768–775.
40. Steffen B and Chraïbi M. Multiscale simulation of pedestrians for faster than real time modeling in large events. *Cell Automat Lec Notes Comput Sci* 2014; 8751: 492–500.
41. Boukerche A, Zhang M and Pazzi RW. An adaptive virtual simulation and real time emergency response system. In: *IEEE international conference on virtual environments, human-computer interfaces and measurements systems*, Hong Kong, May 11–13, 2009.
42. Gwynne S and Rosenbaum E. Employing the hydraulic model in assessing emergency movement. In: Hurley MJ, Gottuk DT, Hall JR Jr, et al. (eds) *SFPE handbook of fire protection engineering*. New York: Springer, 2013.
43. Ronchi E, Nilsson D, Kuligowski E, et al. The process of verification and validation of building fire evacuation models. *NIST Tech Note* 2013; 1822.
44. International Maritime Organization. Guidelines for evacuation analyses for new and existing passenger ships. London: MSC/Circ.1238, 2007.
45. Henderson L. The statistics of crowd fluids. *Nature* 1971; 229: 381–383.
46. Wang S, Wainer G, Rajus VS, et al. Occupancy analysis using building information modeling and Cell-DEVS simulation. In: *Proceedings of 2013 SCS/ACM/IEEE symposium on theory of modeling and simulation*. San Diego, CA, April 7–10, 2013.

47. Zhang J and Seyfried A. Empirical characteristics of different types of pedestrian streams. *Procedia Engng* 2013; 62: 655–662.
48. Farrell R, Moallemi M, Wang S, et al. Modeling and simulation of crowd using cellular discrete event systems theory. In: *Proceedings of the 2013 ACM SIGSIM conference on principles of advanced discrete simulation*. Montreal, Canada, May 19–22, 2013.
49. Wang S, Schyndel MV, Wainer G, et al. DEVS-based building information modeling and simulation for emergency evacuation. In: *Proceedings of the 2012 winter simulation conference*. Berlin, Germany, December 9–12, 2012.
50. Helbing D, Farkas I, Molnár P, et al. Simulation of pedestrian crowds in normal and evacuation situations. In: Schreckenberg M and Sharma S (eds) *Pedestrian and evacuation dynamics*. Heidelberg: Springer, 2002, pp.21–58.

Appendix. Verification and validation for the building evacuation model

The rules used in the building evacuation model have been recently used for a project on crowd modeling of a building in Ottawa, Canada. We cannot present the data sets in this paper yet because of privacy issues, but there has been a second round of thorough validation of this model used in this real-world project. Furthermore, we performed multiple tests for the verification and validation of this model. In this Appendix, we present these tests and the results obtained.

A.1. Verification tests

In this section, we present some of the verification tests that were conducted on the evacuation model in Section 4.1. These tests are suggested for evacuation models to test the movement and navigation of pedestrians in the model.⁴³

The first test is to verify the simulation of an occupant maintaining an assigned walking speed over time. The test is based on IMO Test 1 from the IMO Guidelines.⁴³ Figure 23 shows the setup for this test. The figure shows a Cell-DEVS model for a corridor 2 m wide and 40 m long. A pedestrian is placed at the right side of the corridor and is assigned a walking speed of 1 m/s walking along the corridor. The expected result is that the occupant should reach the end of the corridor in 40 s. The cell size is assumed 0.4 m². By assigning a cell delay of 400 ms and running the simulations, it was verified that the pedestrian covers

the distance of the corridor in 40 s. Furthermore, we ran multiple simulations for the verification models above, with different free walking speeds in the range of 1.19 ± 0.25 m/s, as in the evacuation model, and the walking periods achieved were in the range of 27.78 s and 42.55 s, as expected.

The second test is to verify whether the model is able to correctly simulate the boundaries of a scenario. It does so by testing to verify that occupants successfully navigate around a corner. Figure 24 shows snapshots of the simulations for this test. As can be seen in Figure 24, a corridor with a corner is used in this test. Twenty persons are placed in one end of the hallway. The pedestrians should walk to the other end of the corridor and are expected to successfully navigate around the corner without penetrating the boundaries. Figure 24 shows screenshots from the simulations. As can be seen, the pedestrians are placed in the beginning of the simulations in the left bottom corner. The second screenshot shows that the simulations verify that pedestrians successfully navigate around the corner.

In addition to the tests presented above, other tests from Ronchi et al.,⁴³ such as the horizontal counter-flows and the exit route allocation, were also performed to verify our model.

Zhang and Seyfried fundamental diagram experiments

A series of experiments were conducted by Zhang and Seyfried⁴⁷ in which the fundamental diagram was measured by controlling the density of pedestrians in a corridor, by varying the entrance and exit widths. For further details on the experiments such as the setup of the experiments, snapshots of the experiments, and the obtained results, the reader is referred to Zhang and Seyfried.⁴⁷

We simulated the experiments conducted by Zhang and Seyfried using our Cell-DEVS model in Section 4.1. We then obtained and analyzed the fundamental diagram from our simulations. We focus on unidirectional movement as our model is for building evacuation where the crowd moves usually as a flow to the exits, as opposed to flows moving in opposite directions.

Figure 25 shows one of the models used to simulate the experiments. The figure shows a setup with corridor with 2 m entrance and 3 m exit. The models are used to simulate and reproduce the experiment quantitatively. We validate our model by the empirical data of that experiment.

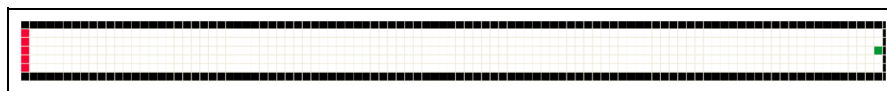


Figure 23. Setup for the speed in a corridor verification test.

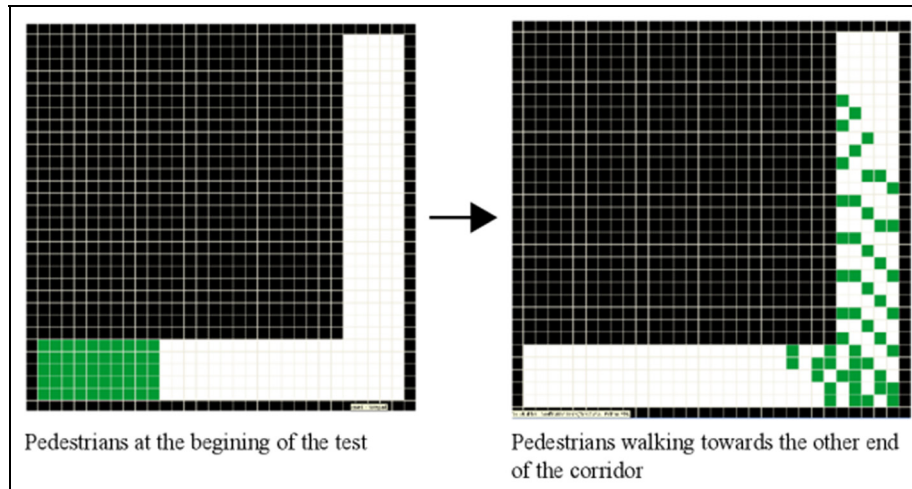


Figure 24. Screenshots from our simulations for the experiment in Ronchi et al.⁴³

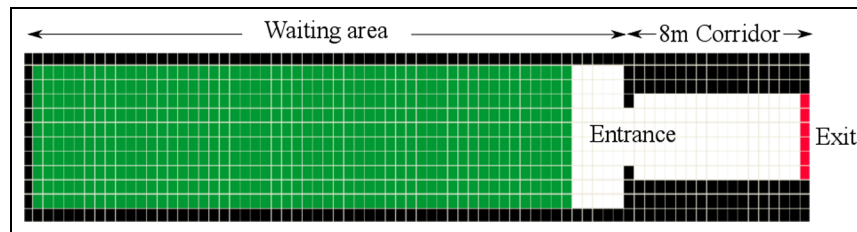


Figure 25. Cell-DEVS model for the Zhang and Seyfried experiments.

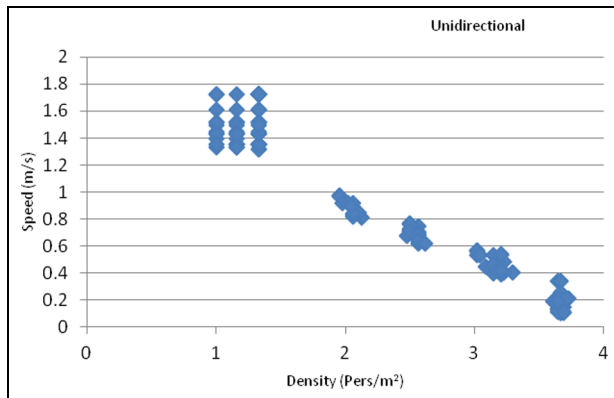


Figure 26. Velocity–density relation obtained from the simulations of the Zhang and Seyfried experiments for a unidirectional crowd.

We simulate different cases where the entrance width varies from 1.5 to 3 m, and the exit width varies from 1 to 3 m. Different delays are used to simulate different free walking speeds in the range 1.55 ± 0.18 m/s as in Zhang and Seyfried experiments. The cell size in the model is assumed to be 0.5 m^2 to simulate maximum density of 4 persons/ m^2 , as per the results in Zhang and Seyfried.

The individual velocity of a person is calculated as per the statistical method of Zhang and Seyfried, where the individual velocity, v_i , is calculated by

$$v_i = \frac{l}{t_i^{\text{out}} - t_i^{\text{in}}} \quad (1)$$

where t_i^{in} and t_i^{out} are the entrance and exit times, respectively, of the measured section for the pedestrian i , and l is the length of measured section. The measurements are taken during the period (30,70) seconds of the simulations in each corridor.

Figure 26 shows the results obtained from our simulations of the Zhang and Seyfried experiment. As can be seen in the figure, the average speed of the pedestrians is in the range 1.55 ± 0.18 m/s for low densities as expected. Figure 26 shows that the results obtained from the simulations are close to the results obtained by the Zhang and Seyfried experiments for unidirectional pedestrian flow. From Figure A4, it can be seen that as the density increases, the speed of pedestrians decreases considerably, as the pedestrians will wait more time in their cells waiting for other cells in the pathway to become vacant. For example, at densities around 2 persons/ m^2 , velocities close to 1 m/s are achieved, which is close to the results by Zhang

and Seyfried. Similarly, at densities around 3 persons/m², velocities around 0.5 m/s are achieved, which is close to the results by Zhang and Seyfried. Further closer results can be even achieved by adjusting the cell delays according to the initial densities of crowds.

Author biographies

Ala'a Al-Habashna received a BEng (with excellent assessment) in computer engineering from Mu'tah University, Karak Governorate, Jordan, in 2006, and an MEng in electrical and computer engineering from Memorial University of Newfoundland, St. John's, Canada, in 2010. He received the fellowship of the School of Graduate Studies at Memorial University of Newfoundland. From 2011 until 2013, he worked with Inmarsat, Mount pearl, Canada, as a technical communications specialist. He is now a PhD candidate at Carleton University, Ottawa, Canada. His current research interests include discrete-event modeling and simulations, signal detection and classification, cognitive radio systems, communication networks architecture and protocols, and multimedia communication over wireless networks.

Gabriel A. Wainer (SMIEEE, SMSCS) received a PhD (1998, with highest honors) at the University of Marseille, France. In July 2000, he joined the Department of Systems and Computer Engineering at Carleton University, where he is a full professor. He has authored three books and over 300 research articles; he edited four other books and helped organizing a large number of conferences, being one of the founders of SimAUD, SIMUTools and TMS-DEVS. He was Vice-President Conferences and Vice-President Publications of SCS (Society for Modeling and Simulation International), where he is a current member of the Board of Directors. He is the Special Issues Editor of SIMULATION, member of the Editorial Board of IEEE Computational Science and Engineering, Wireless Networks (Elsevier), and Journal of Defense Modeling and Simulation (SCS). He received the IBM Eclipse Innovation Award, the SCS Leadership Award, and various Best Paper awards; also, the First Bernard P. Zeigler DEVS Modeling and Simulation Award (2010), the SCS Outstanding Professional Award (2011), Carleton University's Mentorship Award, the SCS Distinguished Professional Achievement Award (2013) and Carleton University's Research Achievement Award (2005 and 2014).