Hierarchical Control of Swarms During Evacuation

Kristýna Janovská and Pavel Surynek[a]
Faculty of Information Technology, Czech Technical University in Prague, Thakurova 9, 160 00 Praha 6, Czech Republic
{*janovkri,pavel.surynek*}*@fit.cvut.cz*

Keywords: Evacuation, multi-agent pathfinding, hierarchical control of swarms, conflict-based search.

Abstract: The problem of evacuation is addressed from the perspective of agent-based modeling (ABM) in this paper. We study evacuation as a problem of navigation of multiple agents in a known environment. The environment is divided into a danger and a safe zone while the task of agents is to move from the danger zone to the safe one. Unlike previous approaches that model the environment as a discrete graph with agents placed in its vertices our approach adopts various continuous aspects such as grid-based embedding of the environment into 2D space continuous line of sight of an agent. In addition to this, we adopt hierarchical structure of multi-agent system in which so called *leading* agents are more informed and are capable of performing multi-agent pathfinding (MAPF) via centralized algorithms like conflict-based search (CBS) while so called *follower* agents are modeled using simple local rules. Our experimental evaluation indicates that suggested modeling approach can serve as a tool for studying the progress and the efficiency of the evacuation process.

1 INTRODUCTION

We address the problem of evacuation (Kurdi et al., 2018) from the perspective of agent-based modeling (ABM) (Wilensky and Rand, 2015; Zia and Ferscha, 2020) and multi-agent path finding (MAPF) (Silver, 2005; Ryan, 2007; Surynek, 2009; Standley, 2010). The evacuation is understood as a navigation problem for agents that need to relocate themselves from the danger zone to the safe zone in a known environment. Typically the environment is modeled as a discrete graph where vertices represent locations and edges define the topology and each agent has the full knowledge of the graph. An agent can be located in a vertex at a time and can instantaneously move across an edge to a vertex in its neighborhood. Similar setup is used in multi-agent path finding, a problem from which we borrow the centralized algorithmic approach and environment modeling.

The basic characteristic of evacuation is that usually we do not care about the precise goal locations of agents. Any location in the safe zone is acceptable as a goal location for an agent. This is an important difference from MAPF where each agent has its specific goal location. In addition to this, centralized control of all agents is hardly reachable in practice hence we need to assume that agents are controlled locally to a significant extent. Individual agents in evacuation do not share the knowledge of locations of other agents. This precludes direct application of MAPF algorithms in solving evacuation problems at the level of individual agents. On the other hand, fine-grained behaviour of individual agents can be captured using the ABM approach.

Previous works often model the evacuation problem as a network flow (Chalmet et al., 1982; Even et al., 2014; Arbib et al., 2018) in a discrete graph. Such approaches however model the problem at a too coarse level where for example interactions (collisions) between individual agents in a limited space are not modeled nor the individual behaviour of agents. Hence these approaches are rather suitable for evacuation modeling at the level of cities and road networks (Kamiyama et al., 2006). More detailed modeling of agents in required for evacuation inside buildings where ABM techniques are more suitable (Liu et al., 2016; Liu et al., 2018; Zafar et al., 2016).

On the other hand, ABM models often lack centralized aspect which often results in insufficient performance of the evacuation process (Trivedi and Rao, 2018).

1.1 Contribution and Organization

We attempt in this work to integrate centralized planning from MAPF and an ABM-based framework for modeling detailed agents' behaviour.

[a] https://orcid.org/0000-0001-7200-0542

Agent behaviour in our approach uses a **hierarchical model** in which some usually more informed agents follow the evacuation plan constructed by the centralized algorithm. Conflict-based search (Sharon et al., 2014), an optimal MAPF algorithm, is used in our case. We call these more informed agents *leaders*. We can assume, that the *leader* agents can be equipped with some electronic device that informs them about the plan and the current situation.

Less informed agents are called *followers* in our hierarchical model and they are controlled locally using various rule-based methods. These agents are assumed to follow the *leader* agents using the local rules.

At the level of environment modeling, our approach is a synthesis between discrete graph-based models and continuous models from which we borrow continuous reasoning about the topology of the environment. Namely, the environment is modeled as a grid embedded in the 2D space where agents are placed in the cells of the grid. Agent themselves have a volume and when using local rules they consider continuous line of sight to localize a *leader* agent.

We demonstrate that our hierarchical ABM approach for evacuation can serve as a tool for studying the progress and the efficiency of the evacuation process.

The paper is organized as follows: we first introduce formal definition of discrete evacuation and basic concepts from related MAPF. Then our novel hierarchical agent-based model of evacuation is introduced. Various models of behaviour of agents are described. Finally, we analyze performance of the evacuation model in diverse scenarios empirically.

2 BACKGROUND

Evacuation is an anonymized form of the multi-agent path finding (MAPF) problem. It takes place in an undirected graph $G = (V,E)$, where each vertex is marked either as safe or endangered, that is $V = S \cup D$, where S are safe vertices and D are endangered vertices. The goal vertex of an agent is not just one vertex, but any vertex from a set of safe vertices.

The task of is to find a set of paths for a set of agents $A = \{a_1, a_2, ..., a_k\}$ that navigates them from endangered vertices D to safe vertices S. Each agent from A starts in its own vertex, so that no more than one agent is present in a vertex at a given time. The solution of this problem is a plan $\pi = [c_0, c_1, ..., c_m]$, $c_m(a) \in S\ \forall a \in A$ and $c_t : A \to V$ is the configuration of agents in vertices at time t.

The following definition summarizes the evacuation problem as introduced in (Selvek. and Surynek., 2019).

Definition 1. Discrete multi-agent evacuation *(MAE) is a 5-tuple $\mathcal{E} = [G = (V,E), A, c_0, D, S]$, where G represents the environment, A is a set of agents, $c_0 : A \to V$ is the initial configuration of agents, D and S such that $D \subseteq V$, $S \subseteq V$, $V = D \cup S$ with $D \cap S \neq \emptyset$, and $|S| \geq k$ represent a set of endangered and a set of safe vertices respectively.*

We usually try to minimize the evacuation time, the total number of steps from the beginning of the evacuation until the moment when the last agent reaches a safe vertex. We call this value the *makespan*.

In our model, the *leader* agents have access to centralized plan obtained by MAPF planning algorithm. Specifically we use the Conflict-Based Search (CBS) algorithm, a makespan optimal MAPF algorithm. It is an algorithm, in which for each agent the shortest path is found from the starting position to the target position so that it does not collide with other agents during the course, that is, no two agents are in the same location at the same time. The algorithm is makespan optimal considering assuming a solution exists (Sharon et al., 2014) (existence of solution can be checked by a different algorithm, however in environment with enough free space a solution always exists).

The algorithm consists of two levels - at the higher level, as shown in Algorithm 1, the construction of a conflict tree takes place, where individual nodes contain a set of conflicts for given agents. A conflict is determined by a pair of agents, a vertex, and a time step where the conflict occurred. The conflict tree then branches to two nodes - one for each agent in the conflict with a constraint forbidding the agent to enter the conflict. In the new nodes, a path-finding algorithm is run for the given agent based on the constraint arising from the conflict, determining the new path for this agent, but without the given conflict.

At the lower level, there is a local path-finding algorithm for each agent - in this paper A* (Hart et al., 1968) algorithm is used. The selected path-finding algorithm works with the limitations given by the higher level of CBS.

The lower level sends the best solution for a given agent to the higher level, which evaluates past conflicts (collisions) based on those solutions and constructs a conflict tree.

The algorithm is guaranteed to find the makespan optimal solutions. For details see (Sharon et al., 2014).

Algorithm 1: Higher level of CBS (Sharon et al., 2014)

```
   Input: MAPF instance
 1 R.constraints = $\emptyset$;
 2 R.solution = set of individual agent paths;
 3 R.make $\leftarrow$ compute_makespan(R.solution);
 4 OPEN $\leftarrow$ R;
 5 while OPEN not empty do
 6     P $\leftarrow$ lowest_makespan_node(OPEN);
 7     OPEN $\leftarrow$ OPEN \ {P};
 8     first_conflict $\leftarrow$ validate(P);
 9     if first_conflict = $\emptyset$ then
10         return P.solution;
11     for $agent_i$ in first_conflict do
12         N $\leftarrow$ new_node;
13         N.constraints $\leftarrow$ P.constraints $\cup$
             new_constraint($agent_i$, vertex, time);
14         N.solution $\leftarrow$ P.solution;
15         N.solution.update(new_constraint());
16         N.make $\leftarrow$ compute_makespan(N);
17         Insert N to OPEN;
```

3 MODEL DESIGN

In this section we describe our hierarchical agent model and individual agent types. We also discuss the design of the environment and the combination of discrete and continuous approach.

3.1 Agents

Agents are divided into two types. The motivation for this decision is the evacuation situation in the school environment. Thus, agents can represent teachers and students. We therefore divide them into *leaders* and *followers*. *Leaders* search for a way out of the building, and *followers* form swarms around *leaders*, follow them, and evacuate with their help. We assume that *leaders* have the full knowledge and can be controlled centrally in the principle while the followers can only get information from their neighborhood and are controlled locally.

3.1.1 Follower

The *follower* must be able to observe the *leader* in its immediate neighborhood if no obstacle prevents it from doing so. If there are more *leaders*, it is able to decide which one is closer. It follows either the nearest *leader* or an assigned *leader*. If it loses the sight of its *leader*, it tries to get to the last position where the *leader* was when the *follower* still saw it.

Algorithm 2: Agent function of a follower

```
  Input: MAPF instance, list of leaders'
         positions
1 follower.leader $\leftarrow$ choose_nearest(leaders);
2 if distance(follower, leader) <
   follower.sight_length or not
   follower_sees_leader() then
3     return random_free_neighbour ;
4     if not follower.obeys() then
5         return move(follower, leader, away)
6     else
7         return move(follower, leader,
            towards)
```

In the base design, the *follower* is designed as a purely reflex agent (Russell and Norvig, 2010). That was decided because following the example of evacuation in the school environment, *follower* should follow orders of an authority - in our case the *leader*. Thus, the *follower* does not try to evacuate itself, it only has to find a *leader* who will lead it to the exit. It has an overview only of its immediate surroundings in its field of view. If it sees a *leader* in its proximity and decides to be obedient and follow it, it moves to a free cell that is closer to the *leader*. Otherwise, *follower* tries to move away from the *leader*. If the agent does not find a *leader* in its vicinity, it moves to a random neighbouring free cell or remains in place.

Each model uses a specific version of agent function shown in Algorithm 2, adapted to the needs of individual models.

3.1.2 Leader

The goal of a *leader* is to find the shortest way out of the dangerous area while keeping a certain distance between itself and other *leaders*. Each *leader* has a swarm of *followers* around itself, which it leads to the nearest exit. After each step, *leaders* must respond to changes in the environment, whether they are obstacles or other *leaders*. If the leader is prevented from moving by *followers*, it is able to move them from their positions so that it can perform the next step. A *leader* is designed as an agent with a goal and an environment model. (Russell and Norvig, 2010) It has a constant overview of the map. It searches for the shortest path to the nearest exit on the map, to which it leads his *followers*.

A *leader* in the lower level of the Conflict-Based Search searches for its own path regardless of other *leaders*. For this A* (Hart et al., 1968) algorithm is used, which is modified so that the agent respects the restrictions set by the CBS. When determining

Algorithm 3: Validate

```
Input: Node N
1 conflicts = ∅;
2 for $i \in \{1, 2, ..., len(N.solution)\}$ do
3     for $j \in \{i + 1, i + 2, ..., len(N.solution)\}$ do
4         steps ← min(len(solution[i], len(solution[j]));
5         for $t \in \{1, ..., steps\}$ do
6             geometry ← $compute\_geometry(solution[i], solution[j], N.state)$;
7             if $geometry.in\_sight(a_i, a_j, t)$ then
8                 conflicts ← conflicts ∪ {(i, j, solution[i][t], solution[j][t], t)};
9 return conflicts;
```

the conflict between two *leaders*, we cannot rely only on specific coordinates, but on the area around both agents, which is defined by their parameter of conflict distance, which is a distance the agents need to keep between them.

Conflict is defined as a quintuplet $(a_1, a_2, conflict\ position\ of\ a_1, conflict\ position\ of\ a_2, t)$. The conflict involves two different positions, one for each agent. It was chosen in this way because a conflict can occur for agents even if they are not standing next to each other. It is enough for one of the agents to get into the area too close to the other agent, but for each of them there will be a different coordinate by which they would get into this area. Therefore, each *leader* has a different coordinate restriction. This conflict search is implemented in the *validate* function as shown in Algorithm 3.

In the *validate* function, positions of each pair of *leaders* are compared at each step of their paths. If two *leaders* come too close to each other and there is no obstacle between them, they have a conflict in that particular steps. Further conflicts between those agents are not searched, as their paths will change when the first conflict is removed.

3.2 Environment

The environment where the evacuation takes place is expressed by means of a discrete grid map which is embedded in 2D space so continuous geometric reasoning can be made in the environment. The agents are expressed as circles with a fixed radius chosen on the basis of preliminary experiments. To make the distribution of agents in the environment more realistic, each agent is placed randomly within its square cell.

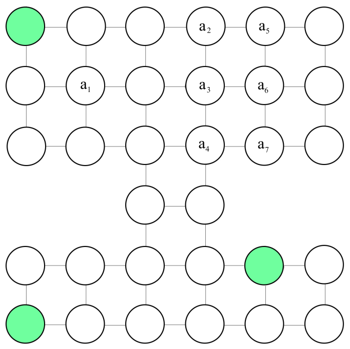

Figure 1: Discrete representation of the environment

Figure 1 depicts the discrete representation of the environment. Vertices marked $a_1 ... a_7$ are vertices occupied by an agent, green vertices represents exits and white vertices are free vertices.

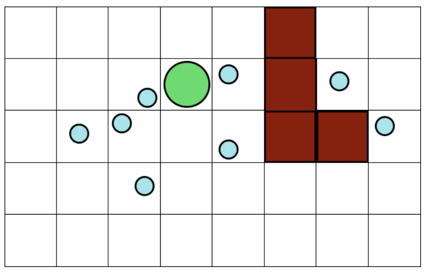

Figure 2: Continuous representation of the environment

In figure 2 we show continuous representation of the environment, blue dots represent *followers*, green dot represents a *leader* and red squares represent walls.

3.2.1 Geometry of the environment

The *follower* can only follow a *leader* if it sees it. We use simulation of line of sight as follows. The *follower F* sees the *leader L* if the distance of the agents given by the Euclidean distance is less than the value of the field of view parameter of the *follower* and at the same time there is no obstacle between the agents, which may be a wall or another *follower*. If the line given by the agents' centers intersects another *follower* or wall, the agent does not see the *leader*.

For *leaders*, the only possible obstacles are walls, which are expressed as squares occupying the entire area of one cell.

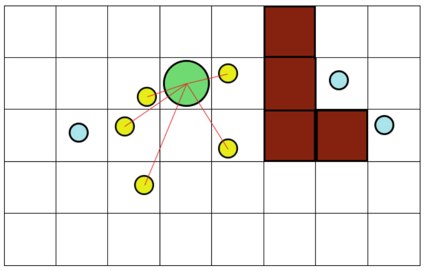
Figure 3: *Followers* depicted yellow see the *leader*

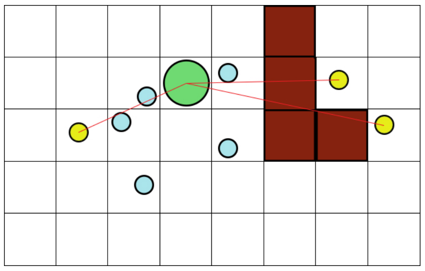
Figure 4: *Followers* depicted yellow don't see the *leader*

4 Models of behavior

Using multi-agent modeling, we examine the evacuation process and determine which of the following evacuation models is the best. Individual models differ in the behavior of both *leaders* and *followers*. Although all models work with hierarchical swarming, the methods of maintaining and guiding these swarms differ in what behavioral rules are used.

4.0.1 Model A – Simple following

In this model, *followers* always choose the nearest *leader* in sight to follow. In the basic design of this model, after separation from *leaders*, *followers* do not attempt to evacuate themselves on their own. *Leaders* only try to get to the safe area without conflicts and as quickly as possible, but they do not have an assigned swarm of *followers* and therefore do not check if any agents have separated along the way. This can lead to significant losses of *followers* in individual swarms.

4.0.2 Model B – Assigned swarms

To prevent significant losses of *followers*, each follower has exactly one *leader* assigned to it, which it must follow all throughout the evacuation. At the same time, in the basic design of the model, *followers* never try to evacuate on their own. *Leaders* look after their respective swarms, regularly checking to see if any agents have separated from the swarm, and if the number of lost agents rises above a given acceptable limit, the *leader* stops and waits for a certain number of steps to allow lost agents to rejoin the swarm.

Thus, during the CBS, the *leader* sends only its current location to the algorithm as its path and does not plan a complete path to the exit, as it is not clear how long it will wait on the spot. It is however possible that due to the inability to move, it will get into a conflict with another *leader*.

However, there is an even greater need for effective planning of individual steps, as the evacuation time increases compared to model A due to frequent stops and waiting. Experiments have shown that the fewer *followers* a *leader* can afford to lose, the slower its evacuation will be.

Partially informed *followers* If the initial position of a purely reflex *follower* is unfavorable (no *leader* is in sight and the position is not near the exit), or it loses its *leader*, it is likely that its evacuation will fail. Therefore, it is appropriate to put in place a mechanism to increase the *follower*'s chances of survival.

Followers are unaware of the ongoing evacuation and therefore do not attempt to reach the exit on their own. However, it is possible to leave them k random steps, after which they will start trying to get to an exit, which however may not necessarily be the nearest exit available. They do not take into account any dynamically emerging barriers that other agents present, because unlike *leaders* they do not cooperate and have no way to obtain information about the position of other agents.

Because these are only *followers*, they plan their path using an algorithm simpler than A* used by *leaders*.

The moment the desired *leader* appears in the field of view of the *follower* after a number of steps, the *follower* becomes a part of the swarm of this *leader* and no longer searches for a path to the exit.

We have therefore introduced a modification of all models, which allows *followers* outside of swarms to perform planned movement. If a *leader* (in Model A any *leader*, in Model B one specific *leader*) no longer appears in their field of view after 30 time steps, they can get the opportunity to evacuate themselves. They use the Breadth-First Search algorithm (BFS) to find their path to the exit. A value of 30 was chosen based on preliminary experiments. We assume that agents have the knowledge of the environment (not the positions of other agents) hence individual pathfinding in the environment can be regarded as a local behaviour.

4.0.3 Model C – Plane

The motivation for this model is evacuation in an airplane or on a ship. The *leaders* do not try to evacuate themselves first, but let *followers* be evacuated before them and leave the space last. As in model B, at the beginning of the evacuation, *leaders* divide the individual *followers* into groups that do not change during the evacuation. The *leader* thus keeps track of whether his assigned swarm has already been evacuated and, if so, only then does it leave the area itself.

Thus, at each step, the *leader* first determines the direction in which it is best to move. It then communicates this information to its *followers*, and they move in response to this instruction. *Leader* itself moves last.

Followers recognize three basic types of their location - they can be located in an aisle, on a seat, or near an exit. Based on this, they perform specific movements.

- Aisle - in this case, the *follower* moves in the direction specified by its *leader*. In this case, the *follower* primarily moves to the right / left, but if a wall, seat or a *leader* prevents it from moving, it has the option to bypass him. *Followers* in this model are also partially informed, so if they do not see their *leader* for a given period of time, they try to evacuate themselves through one of the exits, which, however, may not be the nearest exit for them.
- Seat - a seat is a narrow space surrounded on at least two sides by rows of seats. If a *follower* is in a seat, it tries to move to an aisle.
- Near the exit - if a *follower* is located near the exit, it stops listening to the instructions of the *leader* and moves directly to the exit.

5 EXPERIMENTAL EVALUATION

In this section we will discuss modifications performed to improve the course of experiments and evacuations themselves. Then we will discuss selected experiments.

5.1 Experimental setup

5.1.1 Constraining the depth of the conflict tree

The implementation of the model uses a variant of the CBS algorithm that constructs only the first n vertices of the conflict tree, because non-constrained tree created by the algorithm can have thousands of vertices, which leads to a very long computation time and would make the model impractical for real-life applications. In the experiments, CBS was greatly constrained to save computational time. This could also have contributed to creation of conflicts and congestions during the evacuation. This approach may not return a makespan optimal result, it is only suboptimal. However, since the map space is very large, we don't have to worry that the algorithm won't find a solution.

However, if that happens, the agent who did not find a solution will not move. When replanning occurs in the next step, a position where an agent previously had a conflict with another agent may already be unblocked and the agent may continue on its route.

After each agent performs a step, replanning occurs - the CBS algorithm is therefore run repeatedly. This makes it possible to respond to emerging obstacles on the map, which might be other *leading* agents.

5.1.2 Processing of the resulting vertex

The original CBS algorithm returns a conflict-free vertex. However, since a modification limiting the depth of the search was made, in many cases the vertex identified as the best solution contained conflicts. CBS resolves those conflicts that it encounters first. However, it may not get to all the conflicts that are to take place in the next step. Therefore, for the resulting solution, the set of all coming conflicts is found again, and on the basis of this, the solution is replanned for each agent with a conflict. The final result is this modified solution.

5.2 Selected experiments

The most important experiments we carried out studied impact of numbers of evacuated agents, *follower* obedience, level of communication between *leaders* on makespan and other parameters. We will discuss these experiments in this section.

All selected experiments on models A and B were carried out on map *building*, whereas selected experiment on model C used map *plane*.

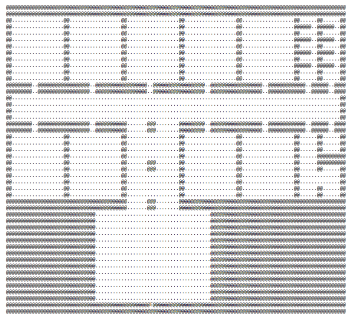
Figure 5: Map *building*

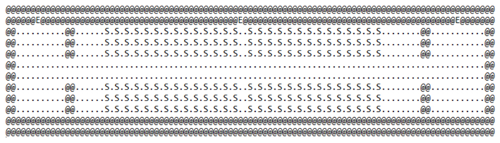
Figure 6: Map *plane*

5.2.1 Comparison between models A, B and usage of partially informed *followers*

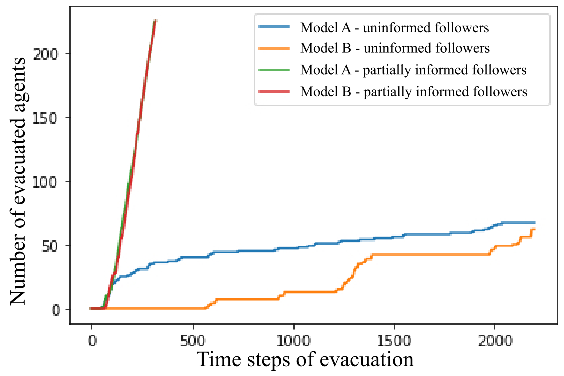

Figure 7: Progress of evacuation depending on the model being used and the usage of partially informed *followers*

Comparison of models A and B using partially informed *followers*. Figure 7 shows that the evacuation process is very similar in both models, with small deviations in favor of model A.

In model B, swarms consist of at least $\lceil \frac{n\_agents}{3} \rceil + 1$ agents, while in model A a *leader* often breaks away from its *followers* and evacuates alone, so in model A, the first agents begin to evacuate themselves earlier than in Model B. These are the solitary *leaders*.

Thus, in model B, unlike model A, a larger number of agents typically arrive at the exit at once. This is reflected in the plot where places with faster and slower growth alternate in this model, while in places with faster growth model B catches up with the numbers of evacuated agents in model A, where the evacuation after the first 130 steps is almost linear.

We can determine that the most important part of the evacuation is, for example, the first half of the evacuation - typically the sooner the agents are evacuated, the better, because there is never an infinite amount of time for the evacuation (Ng and Chow, 2006).

In the first part of the experiment, the numbers of evacuated agents in model A slightly outweighed model B. However, Model B quickly catches up with occasional shortcomings, and variations in evacuated agent numbers are minimal for the remainder of the experiment. In terms of speed, the evacuation time is the same for both models. According to the first metric, model A could be considered more successful. In terms of the rate of evacuation, the models are equal. However, the realism of these models must also be taken into account. We consider model B to be more realistic.

Comparison of models A and B using uninformed *followers*. Figure 7 shows that model A leads in the number of evacuated agents throughout the whole evacuation process. In model B, there are large delays in the evacuation process. Whole parts of swarms always evacuate at once, which is reflected in the plot by significant jumps in the number of evacuated agents.

Because the swarms in Model A are much more scattered, due to the behavior of *leaders*, there are no places where no agents would be evacuated for a long time - in the order of units up to tens of steps. However, after all the agents who were part of a swarm have been evacuated, in both models any additional abandoned agents may evacuate themselves only after approaching the exit by a sequence of random steps.

Comparison of models based on usage of partially informed *followers*. Models using partially informed *followers* have a visible advantage over models working with uninformed *followers*, both in terms of evacuation rate and in terms of the number of evacuees. Model A with uninformed *followers* begins to visibly lose to models with partially informed *followers* after about 100 steps, when the growth in the number of evacuated agents begins to slow down to almost a complete halt.

It is unlikely that models with uninformed *followers* can undergo a complete evacuation because uninformed *followers* do not actively try to evacuate on their own.

5.2.2 Impact of number of evacuated agents on makespan

We observed the development of the makespan - the evacuation time - depending on the number of agents. In models A and B, there are always 9 *leaders*, while the number of *followers* is 1–216. The observed values always differ by 10 agents. In model C, 165 agents are evacuated, out of which 3 are *leading* agents. In all models, *followers* are partially informed.

Model B Outliers in figure 8 are caused by congestions in experiments. Congestion that would prolong the average evacuation time occurred with both

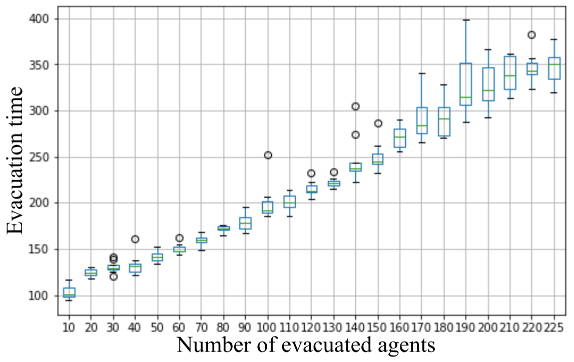
Figure 8: Evacuation time as a function of the number of evacuated agents and the map used in the experiment - model B

a lower number of agents (eg 40 agents) and a higher number (eg 220 agents). However, such cases most often occurred in the range of 100-225 agents.

The value of the average evacuation time increases with the increasing number of agents, but the variability of these values also increases. With a higher number of agents, congestion is more common because several swarms can enter narrow corridors at once and block each other's path. *Leaders* also have to wait for more *followers*, which again seems problematic if the escape route of agents from narrow spaces is blocked by other swarms or *followers* from the same swarm, who are currently waiting for the movement of a standing *leader*. This phenomenon is partially eliminated with a lower number of *followers*, as there are not so many agents in the corridors at once, and they do not block each other's path. However, it is not possible to expect the complete elimination of a certain deviation in evacuation times, because *followers* can significantly prolong the evacuation time by their disobedient behavior.

Due to the problems observed starting with 100 evacuated agents, we can state that the map used becomes dangerous for the number of agents of 100 or more.

Model A Figure 9 shows that the evacuation time increases with the number of evacuated agents. However, unlike in Model B, congestions do not occur, and with the vast majority of agents, evacuation time variations are in the order of units only.

In contrast to model B, several differences are notable – there are no congestions, nor does the variability of evacuation times increase with the increasing number of evacuees. This is due to the behavior of *leaders*, who throughout the evacuation only proceed smoothly to the exit and do not stop to wait for any *followers*. As a result, there are no accumulations of swarms in one place, and therefore no congestions.

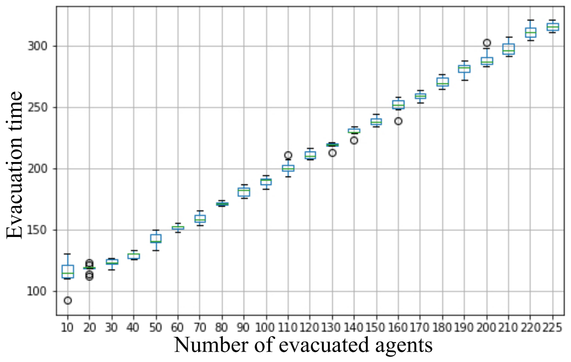
Figure 9: The evacuation time as a function of the number of evacuated agents and the map used in the experiment - model A

The average evacuation times in this model are lower than in model B, except in the case of only 10 agents, that is 9 *leaders* and one *follower*. In this case, *leader* waiting for his respective *follower* helped the evacuation in model B, as the *follower* had a better chance of catching up with his *leader* and did not have to wait for his own path planning to begin, which could have happened in model A.

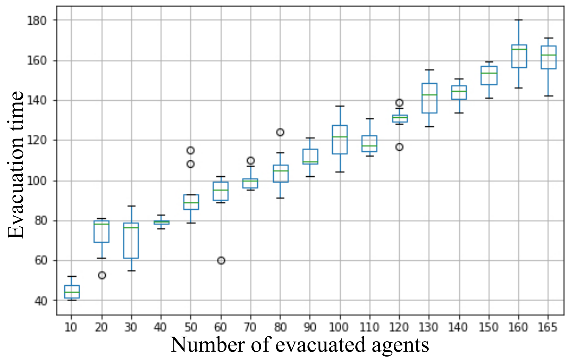
Figure 10: The evacuation time as a function of the number of evacuated agents and the map used in the experiment - model C

Model C As the number of evacuated agents increases, so does the evacuation time. However, between lower and higher numbers of agents, deviations of the evacuation time occur in matter of tens of steps.

In contrast to models A and B, a higher variance of values between the individual numbers of agents occurs in most of the values examined. This may be due to the nature of the environment, in which there are only narrow alleys that support congestions.

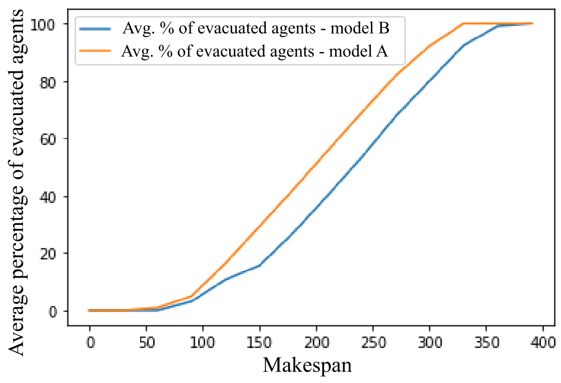

Figure 11: Average percentage of evacuated agents depending on makespan

5.2.3 Impact of makespan on average percentage of evacuated agents

We observed what percentage of evacuees have been evacuated, if we constrain the makespan at a certain number of time steps. We can also choose a percentage of successfully evacuated agents after which the evacuation is deemed successfull. We can also determine what is the minimal time an evacuation needs to take in order to evacuate enough agents.

We can see model A surpassing model B in average percentage of evacuated agents. 10 measurements were taken for each observed value. Observed values were 0-390 time steps, noted every 30 steps. If we deem for example 80% as a sufficient percentage of successfully evacuated agents, evacuation in model A would need around 270 time steps to achieve this number. Evacuation in model B would need to take about 300 steps to evacuate at least 80% of agents.

5.2.4 Impact of minimal swarm size

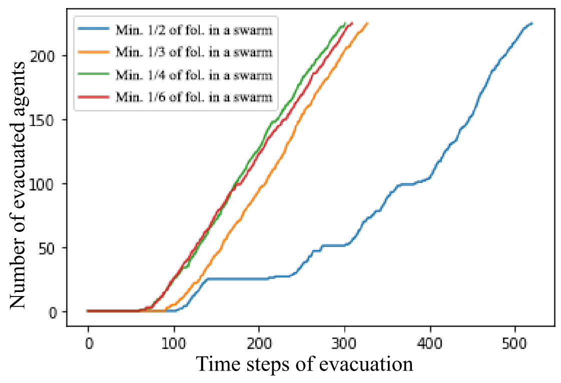

Figure 12: Progress of evacuation as a function of the η parameter

We introduce the minimal swarm size $\eta \in \{\lceil\frac{n\_agents}{2}\rceil, \lceil\frac{n\_agents}{3}\rceil, \lceil\frac{n\_agents}{4}\rceil, \lceil\frac{n\_agents}{6}\rceil\}$, which represents a minimal number of *followers* a *leader* has to keep in a swarm to be able to freely move towards an exit. If the swarm size falls below this limit, the *leader* has to wait for a set amount of time or until the swarm size rises above the set limit again. This experiment was carried out on model B with partially informed *followers*.

It can be seen in figure 13 that although there is a noticeable difference in the evacuation rate of $\eta = \lceil\frac{n\_agents}{2}\rceil$ compared to other cases, the evacuation rate of $\eta = \lceil\frac{n\_agents}{3}\rceil$, $\eta = \lceil\frac{n\_agents}{4}\rceil$ a $\eta = \lceil\frac{n\_agents}{6}\rceil$ is very similar throughout the evacuation, with $\eta = \lceil\frac{n\_agents}{3}\rceil$ lagging slightly behind the two fastest cases. At the same time, it can be noted that in the case of $\eta = \lceil\frac{n\_agents}{2}\rceil$ there was a congestion during the evacuation, which increased the final time of evacuation of this case.

Figure 13 shows that there is a limit to the minimum swarm size below which the evacuation rate no longer differs much. For the performed experiments, the results of the evacuation are very similar for $\eta \leq \lceil\frac{n\_agents}{4}\rceil$, throughout the evacuation process. For higher values of η, the rate of evacuation increased, which was also caused by the congestion, but a slower course of evacuation compared to other cases can be noted in the part of the evacuation preceding this complication.

5.2.5 Impact of *follower* obedience

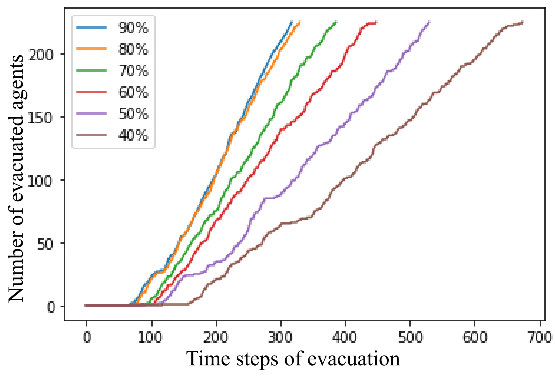

Figure 13: Progress of evacuation as a function of the β parameter

In this experiment we observed the *follower* obedience parameter $\beta \in \{90\%, 80\%, 70\%, 60\%, 50\%, 40\%\}$, which is the probability that a *follower* will decide to follow its *leader* at a given step and will not try to break away from the swarm. This experiment was carried out on model B with partially informed *followers*.

As β decreases, the evacuation time increases by hundreds of time steps. *Followers* with lower β more often do not follow the steps of their *leaders*, thus hindering evacuation. With a lower β, as a result of their disobedience, *followers* more often break away from their swarm and deliberately move away from it. As a result, they will lose sight of their *leaders*, and until they begin to evacuate on their own, they will drop behind significantly. This leads to increasing evacuation time with decreasing β. Lower β can also lead to an increased incidence of congestions.

5.2.6 Impact of communication level between *leaders* on makespan

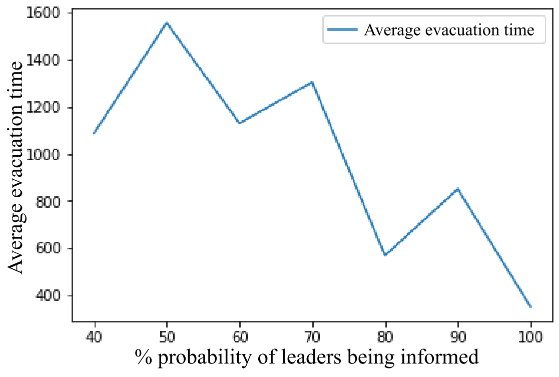

Figure 14: The average evacuation time as a function of the α parameter

Here we observed the parameter $\alpha \in \{100\%, 90\%, 80\%, 70\%, 60\%, 50\%, 40\%\}$, the percentage probability of *leaders* being informed, therefore participating in the CBS. This experiment was carried out on model B with partially informed *followers*.

Unlike *followers*, if a *leader* is informed is decided only once, before the start of path planning. Thus, the same *leaders* always participate in the CBS. 6 experiments were performed for each value of α.

As the value of α decreased, there were more and more conflicts, often multiple, which greatly prolonged evacuation times, as the emergence of these conflicts caused congestions, which lasted hundreds of time steps in a conflict of three or more swarms. The average $\alpha = 90\%$ that exceeds $\alpha = 80\%$ or $\alpha = 70\%$ that exceeds $\alpha = 60\%$ indicates that even a lower number of uninformed agents can cause significant problems .

Thus, this experiment showed the advantage of Conflict-Based Search – $\alpha = 100\%$ has a noticeable advantage over lower values, and although congestions can occur even at this value, their occurrence is more rare. It is therefore important that *leaders* communicate with each other and try to avoid conflicts.

5.2.7 Impact of corridor width

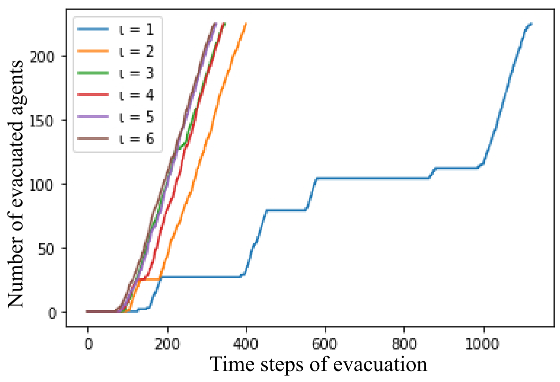

Figure 15: Progress of evacuation as a function the ι parameter

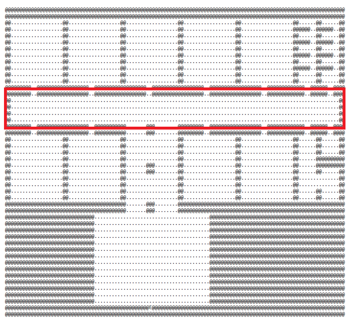

Figure 16: Used map *building* with highlighted corridor whose width has been tested

We observed how the course of evacuation will develop depending on the width of escape routes $\iota \in \{1, 2, 3, 4, 5, 6\}$, ie corridors in which the largest occurrence of congestions occurred in previous experiments. This experiment was carried out on model B with partially informed *followers*.

Figure 15 shows the slowest evacuation process was at $\iota = 1$. $\iota = 2$ lags behind by several tens of steps. The pairs $\iota = 3$, $\iota = 4$ and $\iota = 5$, $\iota = 6$ reached a comparable evacuation time. As the value of ι decreases, the evacuation time increases. It can be seen that the biggest problems are caused by $\iota = 1$, so this setting is inappropriate. We chose $\iota \geq 3$ as an appropriate setting for this parameter.

5.2.8 Summary of results

Experiments have shown the importance of coordinated agent behavior. Conflict-Based Search has proven to be an effective tool for constructing plans for informed *leader* agents which in combination with swarms of the *follower* agents that do not communicate with each other resulted in an efficient evacuation algorithm for large groups of agents.

Another observation is that it is also important to maintain order inside the swarm, which affects the evacuation of the swarm as a whole. It is therefore necessary for *followers* to listen to their *leaders*.

Furthermore, the number of evacuees has been shown to affect evacuation time not only because more agents need to be evacuated, but also because more agents lead to a higher risk of congestions, as the space is easier to fill, which may prevent individual swarms from evacuating.

Experiments also show that in order to evacuate as quickly as possible, it is important that the *leaders* take into account *followers* who may be behind them and wait for them. However, if *followers* are lost and cannot rejoin the swarm, they should try to evacuate themselves.

6 RELATED WORK

This work follows (Selvek. and Surynek., 2019), which also deals with simulation of evacuation in buildings using a multi-agent system. The authors propose a local algorithm for LC-MAE evacuation planning, which is based on sub-optimal algorithms for MAPF. The paper also studies how the course of evacuation affects the presence of uninformed agents, those who plan by themselves in isolation, among informed agents, that plan centrally.

In (Mas et al., 2015) authors simulate evacuation using ABM focusing on the evacuation of cities during the tsunami. The authors present the benefits of simulating evacuations during a tsunami using ABM. They also mention the obstacles that realistic modeling of human behavior poses for successful ABM simulation. Apart from the evacuation simulation, the model proposed by the authors is used to estimate the number of victims, analyze the behavior of evacuees, reveal the limits of the use of shelters or evaluate the use of means of transport.

Evacuation during a natural disaster is also discussed in (Tsurushima, 2021), in which the author bases his model on a video [1] of an actual evacuation during the 2011 Tōhoku earthquake. Based on the video, the behavior of agents, which is also reproduced as a group behavior, is divided into *fleeing* and *falling* to the ground depending on a distance from the exit. The author performs simulations in the environment and settings that correspond to the video.

In (Galea et al., 2003), the authors describe the rules for the evacuation of an aircraft and propose the airEXODUS evacuation model, which is a modification of the EXODUS software tool, which is used to simulate the evacuation of a large number of people from complex environments. According to the authors, this model is able to predict with high accuracy the results of certification tests of an aircraft, but also to predict the sequence of events that may occur during these tests.

[1]https://www.youtube.com/watch?v=tejlDDKeg8s

In (Chen et al., 2020) authors deal with the issue of hierarchical swarm management. They apply this technique to swarms of autonomous drones, in which the hierarchy of leader and follower aircraft applies. The authors deal with the problem of controlling the formation of drones. They propose solutions using group hierarchical swarm control, which achieve coordination outside and inside individual groups of aircrafts. The architecture proposed by the authors reduces the complexity of coordinated planning, as there is no need to plan routes for all aircraft in individual swarms. The authors also propose rules for the control of the formation for the pilot aircraft and the follower aircraft in each group, which guarantee the stability of the entire swarm, even with possible restrictions.

The evacuation and behavior of children in the school environment, which is also the motivation of this work, is discussed in (Chen et al., 2018). Based on experiments on groups of children, the authors reveal several typical behaviors related to distance, obstacles or clogging of space. They also examine group behavior scenarios, which they further compare with the behavior of individuals. They reveal problems during a group evacuation, where children stop and play during the evacuation, or wait for their friends instead of their own evacuation. They also point to the fact that group behavior has an effect on the child's path choice. However, the observations did not take place during the crisis and the children were not helped or influenced by teachers' behavior.

It is important to note that evacuation modeling includes as diverse approaches as fluid-dynamic models that regard the evacuation as fully continuous process solved via differential equations (Sikora et al., 2011). Cellular automata represent another popular tool for studying evacuation (Bazior et al., 2018). Often fine grained modeling of interactions between individual agents such as resolving collisions between agents is studies through the concept of cellular automaton.

7 CONCLUSION

Many studies focus on modelling evacuation using only local algorithms. We have decided to propose a combination of more informed centralized approaches with local approaches to multi-agent evacuation in a hierarchical model via ABM techniques. The designed models can be used in testing efficiency of evacuation plans, building safety, and evacuation progress depending on various parameters.

In our model, agents were divided into two types - *leaders*, who controlled their swarms and guided the swarm to a safe area using a modified global Conflict-Based Search algorithm, a popular algorithm for multi-agent path finding, and *followers*, who aim to follow the *leaders* to the safe zone.

We compared different models of behavior of both types of agents and experimentally verified their impact on the progress of evacuation. The results of our work pointed out the importance of communication between *leaders* in this type of evacuation. Implementing partially centralized approach has increased the efficiency of the evacuation over scenarios using only local approach, where *leaders* didn't communicate. We also showed how mistakes or disobedience of the *follower* agents affect the progress of evacuation and we identified the problems that can occur during evacuation.

In future work, we plan to expand the experiments and propose further modifications of our proposed model, so that the idea of hierarchical control of swarms during evacuation could be tested in a wider range of situations. We also plan to verify how realistic our models are. For this we plan to acquire data from evacuations that are unfortunately rare and difficult to obtain. Another option is to perform tests using volunteers.

ACKNOWLEDGEMENT

This work has been supported by GAČR - the Czech Science Foundation under the grant registration number 19-17966S, and by the VýLet 2021 project sponsored by the Faculty of Information Technology, Czech Technical University in Prague.

REFERENCES

Arbib, C., Muccini, H., and Moghaddam, M. T. (2018). Applying a network flow model to quick and safe evacuation of people from a building: a real case. In *Proceedings of the GEOSAFE Workshop on Robust Solutions for Fire Fighting, RSFF 2018, L'Aquila, Italy, July 19-20, 2018.*, pages 50–61.

Bazior, G., Palka, D., and Was, J. (2018). Cellular automata based modeling of competitive evacuation. In *Cellular Automata - 13th International Conference on Cellular Automata for Research and Industry, ACRI 2018, Como, Italy, September 17-21, 2018, Proceedings*, volume 11115 of *Lecture Notes in Computer Science*, pages 451–459. Springer.

Chalmet, L. G., Francis, R. L., and Saunders, P. B. (1982). Network models for building evacuation. *Fire Technology*, 18(1):90–113.

Chen, H., Wang, X., Shen, L., and Cong, Y. (2020). Formation flight of fixed-wing uav swarms: A group-based hierarchical approach. *Chinese Journal of Aeronautics*.

Chen, L., Tang, T.-Q., Song, Z., Huang, H.-J., and Guo, R.-Y. (2018). Child behavior during evacuation under non-emergency situations: Experimental and simulation results.

Even, C., Pillac, V., and Hentenryck, P. V. (2014). NICTA evacuation planner: Actionable evacuation plans with contraflows. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 1143–1148. IOS Press.

Galea, E., Blake, S., Lawrence, P., and Gwynne, S. (2003). The airexodus evacuation model and its application to aircraft safety.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, SSC-4(2):100–107.

Kamiyama, N., Katoh, N., and Takizawa, A. (2006). An efficient algorithm for evacuation problem in dynamic network flows with uniform arc capacity. *IEICE Trans. Inf. Syst.*, 89-D(8):2372–2379.

Kurdi, H. A., Al-Megren, S., Althunyan, R., and Almulifi, A. (2018). Effect of exit placement on evacuation plans. *European Journal of Operational Research*, 269(2):749–759.

Liu, C., li Mao, Z., and min Fu, Z. (2016). Emergency evacuation model and algorithm in the building with several exits. *Procedia Engineering*, 135:12 – 18. 2015 International Conference on Performance-based Fire and Fire Protection Engineering (ICPFFPE 2015).

Liu, H., Xu, B., Lu, D., and Zhang, G. (2018). A path planning approach for crowd evacuation in buildings based on improved artificial bee colony algorithm. *Applied Soft Computing*, 68:360 – 376.

Mas, E., Koshimura, S., Imamura, F., Suppasri, A., Muhari, A., and Adriano, B. (2015). Recent advances in agent-based tsunami evacuation simulations: Case studies in indonesia, thailand, japan and peru.

Ng, C. and Chow, W. (2006). A brief review on the time line concept in evacuation.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition.

Ryan, M. R. K. (2007). Graph decomposition for efficient multi-robot path planning. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Ar-*

tificial Intelligence, Hyderabad, India, January 6-12, 2007, pages 2003–2008.

Selvek., R. and Surynek., P. (2019). Engineering smart behavior in evacuation planning using local cooperative path finding algorithms and agent-based simulations. In *Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - Volume 2: KEOD,*, pages 137–143. INSTICC, SciTePress.

Sharon, G., Stern, R., Felner, A., and Sturtevant, N. R. (2014). Conflict-based search for optimal multi-agent pathfinding.

Sikora, W., Malinowski, J., and Kupczak, A. (2011). Model of skyscraper evacuation with the use of space symmetry and fluid dynamic approximation. In *Parallel Processing and Applied Mathematics - 9th International Conference, PPAM 2011, Torun, Poland, September 11-14, 2011. Revised Selected Papers, Part II*, volume 7204 of *Lecture Notes in Computer Science*, pages 570–577. Springer.

Silver, D. (2005). Cooperative pathfinding. In *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference, June 1-5, 2005, Marina del Rey, California, USA*, pages 117–122.

Standley, T. S. (2010). Finding optimal solutions to cooperative pathfinding problems. In Fox, M. and Poole, D., editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press.

Surynek, P. (2009). A novel approach to path planning for multiple robots in bi-connected graphs. In *2009 IEEE International Conference on Robotics and Automation, ICRA 2009, Kobe, Japan, May 12-17, 2009*, pages 3613–3619. IEEE.

Trivedi, A. and Rao, S. (2018). Agent-based modeling of emergency evacuations considering human panic behavior. *IEEE Trans. Comput. Soc. Syst.*, 5(1):277–288.

Tsurushima, A. (2021). Reproducing evacuation behaviors of evacuees during the great east japan earthquake using the evacuation decision model with realistic settings. In Rocha, A. P., Steels, L., and van den Herik, H. J., editors, *Proceedings of the 13th International Conference on Agents and Artificial Intelligence, ICAART 2021, Volume 1, Online Streaming, February 4-6, 2021*, pages 17–27. SCITEPRESS.

Wilensky, U. and Rand, W. (2015). *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo*. The MIT Press.

Zafar, M., Zia, K., Muhammad, A., and Ferscha, A. (2016). An agent-based model of crowd evacuation integrating agent perception and proximity pressure. In *Proceedings of the 14th International Conference on Advances in Mobile Computing and Multi Media, MoMM 2016, Singapore, November 28-30, 2016*, pages 12–19. ACM.

Zia, K. and Ferscha, A. (2020). An agent-based model of crowd evacuation: Combining individual, social and technological aspects. In *Proceedings of the 2019 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, SIGSIM-PADS 2020, Miami, FL, USA, June 15-17, 2020*, pages 129–140. ACM.