Evolutionary Algorithms for Optimizing Emergency Exit Placement in Indoor Environments*

Carlos Cotta[1,2][0000−0001−8478−7549] and José E. Gallardo[1,2][0000−0002−0646−2535]

[1] Dept. Lenguajes y Ciencias de la Computación, ETSI Informática, Campus de Teatinos, Universidad de Málaga, 29071 Málaga, Spain
[2] ITIS Software, Universidad de Málaga, Spain
{ccottap,pepeg}@lcc.uma.es

Abstract. The problem of finding the optimal placement of emergency exits in an indoor environment to facilitate the rapid and orderly evacuation of crowds is addressed in this work. A cellular-automaton model is used to simulate the behavior of pedestrians in such scenarios, taking into account factors such as the environment, the pedestrians themselves, and the interactions among them. A metric is proposed to determine how successful or satisfactory an evacuation was. Subsequently, two metaheuristic algorithms, namely an iterated greedy heuristic and an evolutionary algorithm (EA) are proposed to solve the optimization problem. A comparative analysis shows that the proposed EA is able to find effective solutions for different scenarios, and that an island-based version of it outperforms the other two algorithms in terms of solution quality.

Keywords: Pedestrian Evacuation · Cellular Automata · Greedy Heuristics · Evolutionary Algorithms.

1 Introduction

In the event of an emergency, the rapid and orderly evacuation of crowds from enclosed spaces is essential to minimize casualties and ensure public safety. Needless to say, it can also become a critical challenge requiring meticulous planning at different levels, in order to avoid panic, bottlenecks, and potential harm to people in a potentially chaotic scenario [9]. There are different factors that need being taken into account depending on the specificities of each situation (e.g., what the particulars of the environment are, what the typical size and composition of the crowd is, and so on), and the level at which the planning is done (e.g., architectural decisions, signaling, etc.). In this work we are specifically concerned about the placement of emergency exits in the most convenient way to facilitate the efficient evacuation of the crowd.

* This work is supported by Spanish Ministry of Science and Innovation under project Bio4Res (PID2021-125184NB-I00 – http://bio4res.lcc.uma.es) and by Universidad de Málaga, Campus de Excelencia Internacional Andalucía Tech.

In order to approach any evacuation optimization problem –such as the one considered here– and attain safe and efficient evacuation plans, understanding and predicting the behavior of pedestrians is of paramount importance. However, pedestrian evacuation is a complex and dynamic process, influenced by many factors, such as the environment, the pedestrians themselves, and the interactions among them. Therefore, modeling pedestrian evacuation is a challenging task that requires a balance between simplicity and realism. There are different tools that can be used for this purpose, depending on the scope of the simulation. Thus, whereas macroscopic approaches will often consider the crowd as a continuous medium whose flow is to be modeled, e.g., see [2,8], microscopic models will focus on the pedestrians –the individual components of the crowd– and model the crowd behavior as an emergent property of the collective behavior of those individual agents. The latter models can be further divided into two major categories, namely models based on social forces (in which pedestrians are particles in a continuous space, subject to different forces resulting from their interaction with the environment and other particles, e.g., [3,12]), and cellular-automaton (CA) models (in which the environment is modeled as a discrete grid, and pedestrians transition between these following some predefined rules, e.g.,[16,18]). We refer to [4,13] for a more in-depth survey of all these approaches.

We have precisely considered the CA approach in this work, and devised a model for modeling the behavior of a crowd evacuating an indoor environment (see Sect. 3). Using this tool, we aim to find which would be the most appropriate location for emergency exits. This also entails defining appropriate metrics to assess to which extent an evacuation was successful/satisfactory or not. We do this in Sect. 2. Subsequently, we consider different algorithmic approaches to tackle this problem. To be precise, we devise an iterated greedy heuristic and an evolutionary algorithm (EA) for this purpose (see Sect. 4). We conduct an extensive experimentation to analyze the performance of these algorithms (as well as an island-based version of the EA) in Sect. 5. Our main aim in this work is to determine the effectiveness of these approaches for this particular optimization setting, as a stepping stone for devising more powerful approaches and tackling more complex evacuation scenarios. We close this work with a critical outlook of the results and an overview of the following steps in this research.

2 Problem Statement

In order to model the evacuation problem, we need to start by formalizing the indoor space from which the evacuation is attempted. To this end, let $\mathcal{A}$ be this space, which we will assume to be a rectangular area of width w and height h. This rectangular area represents the floor plan of an enclosed space and therefore all its boundaries are assumed to be blocked (i.e., to be non-traversable), except in specific locations which will be denoted as *accesses*. More precisely, we can define an access α as a pair (p_α, w_α), where p_α denotes a point along the perimeter (i.e., a value between 0 and $2(w+h)$, where 0 corresponds to a certain predefined reference point (e.g., the bottom-left corner of $\mathcal{A}$) of the area at

which the access is anchored, and w_α denotes the width of the access along the perimeter, that is, the access extends from p_α to $p_\alpha + w_\alpha$[3]. Now, within $\mathcal{A}$ there may be a number of *obstacles*. Each obstacle $o \subseteq \mathcal{A}$ denotes a non-traversable region (representing real-world objects such as walls or furniture). Therefore, the whole environment can be represented as a tuple (w, h, A, O), where:

- w and h are the width and height of $\mathcal{A}$ respectively.
- $A = \{\alpha_1, \ldots, \alpha_k\}$ is a collection of accesses.
- $O = \{o_1, \ldots, o_m\}$ is a collection of obstacles.

This environment is crowded with n pedestrians (they represent the users of said environment, i.e., residents, workers, customers, etc. depending on what it is being modeled) distributed along traversable areas of $\mathcal{A}$. At time $t = 0$, an emergency is declared and the evacuation of the place begins. Let $\mathbb{M}$ be a model that can be used to predict the behavior of pedestrians in this context, and how the evacuation process would then be conducted (cf. Sect. 3). Let $\rho_i(t)$ represent the position coordinates of the i-th pedestrian at time t, and let T be the maximum time up to which the model is simulated. Then, we can split the collection of pedestrians into two sets:

- *evacuees* $\xi^+ = \{\ i \mid 1 \leqslant i \leqslant n, \exists t_i \leqslant T : \exists \alpha \in A : \rho_i(t_i) \in \alpha\}$, i.e., all pedestrians i who manage to reach an access before T.
- *non-evacuees* $\xi^- = \{1, \ldots, n\} \setminus \xi^+$, i.e., the pedestrians who could not reach an access before T. Given a non-evacuee i, we can define $d_i = \min_{\alpha \in A} \|\alpha - \rho_i(T)\|$, i.e., their distance to the nearest exit at the end of the simulation.

In order to quantify the extent to which the evacuation is successful, different metrics could be used. We consider the following hierarchy of objectives:

1. The first goal is to minimize the number of non-evacuees $|\xi^-|$. This has the highest priority.

The next levels of the hierarchy depend on whether the first goal could be accomplished or not. In the first case ($\xi^- = \emptyset$), we consider:

2a. Minimize the time at which the last pedestrian left the area, i.e., minimize $t^* = \max_{1 \leqslant i \leqslant n} t_i$.
3a. Minimize the average time at which pedestrians left the area, i.e., minimize $\bar{t} = \frac{1}{n} \sum_{1 \leqslant i \leqslant n} t_i$.

If the evacuation was however not complete, then:

2b. Minimize the minimum distance between a non-evacuee and an access, i.e., minimize $d^* = \min_{i \in \xi^-} d_i$.
3b. Minimize the average distance between non-evacuees and accesses, i.e., minimize $\bar{d} = \frac{1}{n} \sum_{i \in \xi^-} d_i$.

[3] Note that since the perimeter is closed, the sum is to be understood as cycling back to 0 when reaching $2(w + h)$.

This hierarchy of goals can be combined into a single numerical value by using appropriate weights that ensure that any comparison respects said hierarchy. To be precise, let $\sigma(\mathcal{A}, S)$ be a tuple containing the evacuation status of each pedestrian and the corresponding value of d_i or t_i at the end of the simulation, given that $S = [\rho_1(0), \ldots, \rho_n(0)]$ are the initial positions in $\mathcal{A}$ of the pedestrians at time $t = 0$. Then, we define:

$$\begin{aligned} f(\sigma(\mathcal{A}, S)) = |\xi^-| + [\xi^- = \emptyset] \left(\tfrac{1}{T} \max_{1 \leqslant i \leqslant n} t_i + \tfrac{1}{nT^2} \sum_{1 \leqslant i \leqslant n} t_i \right) + \\ + [\xi^- \neq \emptyset] \left(\tfrac{1}{D} \min_{i \in \xi^-} d_i + \tfrac{1}{nD^2} \sum_{i \in \xi^-} d_i \right) \end{aligned} \tag{1}$$

where $[\cdot]$ are Iverson brackets, and $D = \sqrt{w^2 + h^2}$ is the diagonal of the area. Now, we can formally define the Optimal Evacuation Problem (OEP) as:

Instance: a tuple $(\mathcal{A}, \mathbb{S}, k, \omega)$, where
- $\mathcal{A} = (w, h, A, O)$ is the environment.
- $\mathbb{S} = \{S_1, \ldots, S_l\}$ is a collection of initial configurations of n pedestrians, i.e., for all $1 \leqslant i \leqslant l$, $|S_i| = n$.
- $k \in \mathbb{N}$ is a non-zero value that indicates the number of emergency exits whose location is sought.
- $\omega > 0$ is the width of emergency exits.

Solution: a collection $E = \{e_1, \ldots, e_k\} \subset [0, 2(w + h)]$, where each e_i represents the location of an emergency exit and such that

$$\psi(E) = \frac{1}{l} \sum_{1 \leqslant i \leqslant l} f(\sigma(\mathcal{A}', S_i)) \tag{2}$$

is minimal, where $\mathcal{A}'$ is obtained from $\mathcal{A}$ by adding $\{(e_1, \omega), \ldots, (e_k, \omega)\}$ to the existing accesses.

Having defined the problem, let us turn our attention to how pedestrian behavior is modeled in next section.

3 A CA for modeling pedestrian evacuation

Cellular automata are simple and powerful tools to simulate complex systems, as they can capture the emergence of global patterns from local interactions. In this section, we describe the details of our CA model for pedestrian evacuation.

3.1 State of the CA

The state of the CA is the state of each cell in the environment (represented by a regular lattice of square cells). Each cell can be in one of three states:

- `empty`: The cell is empty and can be occupied by a pedestrian.
- `occupied`: The cell is occupied by a pedestrian.

– `obstacle`: The cell is occupied by an obstacle and cannot be occupied by a pedestrian.

Some cells in the environment are marked as *exit* cells. These are the cells that the pedestrians want to reach to leave the environment. We assume that pedestrians are rational and will try to find the shortest path to the nearest exit. However, the presence of obstacles and other pedestrians can affect their movement and make them choose alternative paths. To capture this behavior, we define two concepts for each cell: the *static field* and the *crowd repulsion*. The former is a measure of how close a cell is to an exit. The latter is a measure of how crowded the neighborhood of a cell is, taking into account obstacles and other pedestrians. We use these two concepts to calculate the *desirability* of a cell, which is the probability that a pedestrian will move to that cell.

The static field of a cell is computed using Dijkstra's algorithm, which is a well-known algorithm for finding the shortest path between two nodes in a weighted graph [6]. We consider the environment as a graph, where nodes are cells and edges are connections between neighboring cells. The weight of an edge is the geometric distance between the cell centers, if the target cell is not an obstacle and infinity otherwise. Formally, we define the graph as $G = (V, E)$, where V is the set of cells in the environment and E is the set of edges between neighboring cells. The weight function is $w : E \to \mathbb{R}^+$, such that $w(v_i, v_j)$ is the geometric distance between cells v_i and v_j, as defined before. Let $\mathcal{SP}_{i,j}$ be the length of the shortest path from cell (i, j) to its nearest exit as computed by Dijkstra's algorithm. The static field of a cell (i, j) is then defined as:

$$\mathcal{SF}_{i,j} = 1 - \frac{\mathcal{SP}_{i,j}}{\mathcal{SP}_{\max}} \tag{3}$$

where $\mathcal{SP}_{\max}$ is the larger shortest path from any cell in the environment to its nearest exit. This definition makes the static field be in [0,1] and only depend on the relative distance of a cell to its nearest exit. The higher the static field, the closer the cell is to an exit. Notice that, as this field is static, it does not change over time and is only computed once before the simulation.

The crowd repulsion of a cell is computed using the number of reachable cells in its neighborhood. A cell is reachable if it is currently empty and not blocked by an obstacle. For each occupied cell (i, j), let $\mathcal{N}_{i,j}$ be the set of reachable cells in its neighborhood. The repulsion of a cell (i, j) is defined as the inverse of one plus the number of reachable cells in this neighborhood:

$$\mathcal{R}_{i,j} = (1 + |\mathcal{N}_{i,j}|)^{-1} \tag{4}$$

where $|\cdot|$ denotes the cardinality of a set. This definition makes the repulsion be in (0,1] and depend on how crowded the neighborhood of a cell is. The higher the repulsion, the more crowded the neighborhood is.

The desirability of a cell is computed using a combination of the static field and the crowd repulsion. We introduce two parameters to weight the importance of these two factors: the *field attraction bias* ϕ and the *crowd repulsion bias* ζ.

The field attraction bias reflects how strongly the pedestrians are attracted to the exit cells, while the crowd repulsion reflects how strongly the pedestrians are repelled by the crowded cells. We firstly define the attraction of a cell (i, j) as:

$$\mathcal{A}_{i,j} = \exp(\phi \cdot \mathcal{SF}_{i,j} - \zeta \cdot \mathcal{R}_{i,j}) \tag{5}$$

In this way, the attraction of a cell is a positive number that increases with the static field and decreases with the crowd repulsion. The higher the attraction, the more desirable the cell is. However, we can make the pedestrian behavior more realistic and adaptive by reducing the reliance on the global knowledge of the environment and by making use of the information available in the local neighborhood. As the attraction of a cell is not enough to capture this behavior, we need to consider instead the *desirability* of a cell, which is defined as the gradient of its attraction. The desirability of a cell reflects how the attraction changes locally by comparing the attraction of the cell with the minimum attraction in its reachable neighborhood. Let $\mathcal{A}_{\min_{i,j}}$ denote the minimum attraction in neighborhood of cell (i, j), which is defined as:

$$\mathcal{A}_{\min_{i,j}} = \min_{(k,l)\in\mathcal{N}_{i,j}} \mathcal{A}_{k,l} \tag{6}$$

Then, the desirability of cell (i, j) is defined as:

$$\mathcal{D}_{i,j} = \epsilon + \mathcal{A}_{i,j} - \mathcal{A}_{\min_{i,j}} \tag{7}$$

where ϵ is a small number which is added to avoid the desirability being zero ($\epsilon = 10^{-5}$ in our implementation). In this way, the desirability of a cell is a positive number that increases with the gradient of the attraction. The higher the desirability, the more likely a pedestrian will move to that cell. The desirability of a cell is the main input of the local rule that updates the state of each cell on each time step. The local rule is based on a probabilistic transition function that determines the probability of a pedestrian moving from one cell to another.

3.2 Update Procedure

The update procedure is the procedure that is used to update the state of the CA on each time step. The procedure is as follows:

1. We start by marking as empty in the next state the cells that are currently occupied by pedestrians, as they may change depending on their movement.
2. We then mark the cells that are occupied by obstacles in the current state as obstacle in the next state. These cells will not change, as they cannot be occupied by pedestrians.
3. We also mark the exit cells that are occupied by pedestrians in the current state as empty in the next state. This models the evacuation of the pedestrians through the exits. We assume that once a pedestrian reaches an exit, they leave the environment and do not come back.

4. For any other cell that is occupied by a pedestrian in the current state, we compute the desirabilities of reachable neighboring cells. We use the desirability as the probability of a pedestrian moving to that cell and randomly select one neighboring cell according to these probabilities. If the selected cell is not occupied by another pedestrian in the next state, we mark it as occupied by the pedestrian in the next state. This means that the pedestrian moves to that cell. Otherwise, we mark the current cell as occupied by the pedestrian in the next state, i.e., the pedestrian stays in the same cell. This way, we avoid collisions between pedestrians and ensure that each cell can have at most one pedestrian. To ensure fairness among pedestrians, we shuffle the order in which we process occupied cells on each time step.

We consider that each cell in the environment is a square and we denote by cl its side length. We denote the time elapsed for each time step as Δt. The speed of a pedestrian that moves to a neighboring cell on each time step is then $cl/\Delta t$. We call this the *reference speed* of a pedestrian, and denote it by v. However, not all pedestrians may move at the same speed (for instance, some pedestrians may move slower than the reference speed, due to physical or psychological factors). To model this, we introduce for each pedestrian a parameter called *velocity percent* (v_p), which is a percentage of the reference speed. For example, if $v_p = 0.5$ for a pedestrian, their speed would be $0.5v$. We model this by letting v_p be the probability of a pedestrian moving to a neighboring cell on each time step so that, on average, their speed would be $v_p \cdot v$.

3.3 Transition Function

The transition function is the function that determines the probability of a pedestrian moving from one cell to another. The function is based on the desirability of the neighboring cells. The function is defined as follows:

$$T(c_i, c_j) = \begin{cases} P_{i,j} \cdot v_p, & \text{if } c_j \text{ is empty or an exit in the current state} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

where c_i and c_j are two neighboring cells, $P_{i,j}$ is the probability of agent in cell c_i to move to cell c_j based on its desirability:

$$P_{i,j} = \frac{\mathcal{D}_{c_j}}{\sum_{c \in \mathcal{N}_{c_i}} \mathcal{D}_c} \tag{9}$$

and v_p is the velocity percent of the pedestrian in cell c_i. The transition function returns the probability of the pedestrian in cell c_i moving to cell c_j on the next time step. The function is zero if cell c_j is blocked or already occupied by another pedestrian in the current state, or if the pedestrian in cell c_i does not move in this time step, which happens with probability $1 - v_p$. The transition function is applied to each occupied cell in the current state, after shuffling the order of the cells. The result of the function and the procedure to avoid collisions (step

Algorithm 1: Greedy constructive heuristic

```
Data: an instance OEP(A, S, k, ω)
E ← ∅;
η ← ⌈2(w + h)/ω⌉;
for i ← 1 to k do
    p ← rand(0, 2(w + h));
    best ← ∞;
    for j ← 1 to η do
        cur ← ψ(E ∪ {p});
        if cur < best then best ← cur; e ← p;
        p ← p + ω;
        if p > 2(w + h) then p ← p − 2(w + h);
    end
    E ← E ∪ {e};
end
return E
```

4 in Sect. 3.2) is used to update the state of the CA on the next time step. The update procedure is repeated until all the pedestrians have evacuated or a maximum number of time steps (corresponding to time T) is reached.

4 Algorithms for Emergency Exit Optimization

As indicated in Sect. 2, a solution to problem instance $\mathrm{OEP}(\mathcal{A}, \mathbb{S}, k, \omega)$ is a set $E = \{e_1, \ldots, e_k\} \subset [0, 2(w+h)]$. The mapping between solutions and their associated objective functions values is not just non-linear, but also not available in closed form, and only computable via a stochastic simulation. Thus, it is complex to design low-level heuristics to construct such solutions. We can however engineer a constructive approach on top of the simulations, based on greedy principles. The core of this approach is shown in Algorithm 1.

This procedure starts by picking a random initial point p along the perimeter. Then all points $p, p+\omega, p+2\omega, \ldots, p+\eta\omega$ are potential candidates to place an exit, where the addition is assumed to wrap around the length of the perimeter, and η is picked so as to ensure that we cover the whole perimeter. For each candidate, we simulate the system with an emergency exit in the corresponding location (in addition to any other exits that might have been considered in previous steps), and keep the one that returns the best value of the objective function. This is repeated as many times as needed (i.e., k times) to construct the solution. Notice that this procedure involves computing the value of the objective function $\eta \cdot k$ times. Also, this is a randomized procedure and therefore can be iterated as many times to desired to obtain different greedy solutions. We will denote this latter iterated procedure as greedy.

As an alternative to this greedy heuristic, we consider an EA approach. This is a real-coded EA in which individuals are vectors of k values in the range $[0, 2(w+h)]$. We can initially generate such vectors by sampling uniformly at random the

Algorithm 2: Set-based recombination

Data: two sets $E = \{e_1, \ldots, e_k\}$ and $E' = \{e'_1, \ldots, e'_k\}$
$C \leftarrow E \cup E'$; $S \leftarrow \emptyset$;
for $i \leftarrow 1$ **to** k **do**
  $e \leftarrow$ PICK (C); `// makes random selection`
  $S \leftarrow S \cup \{e\}$; $C \leftarrow C \setminus \{e\}$;
end
return S

search space. Notice that we do not introduce any constraint regarding the non-overlap of exits. Having two overlapping exits is equivalent within the simulation to having a single exit of width $2\omega-$overlap. We pose that this is less convenient than having two exits back to back without overlapping, or those two exits strategically placed somewhere else. For this reason, we expect evolution will get rid of those suboptimal solutions without the need of introducing an explicit constraint. As to mutation, we have opted for a Gaussian perturbation of a single exit, whose amplitude is a certain percentage γ of its current value, i.e.,

$$e' \leftarrow e \cdot (1 + \gamma \mathcal{N}(0, 1)) \tag{10}$$

where $\mathcal{N}(0, 1)$ is a normally distributed random value of mean 0 and variance 1. As usual, the value of the variable will wrap around $[0, 2(w + h)]$. As for recombination, we have opted for a discrete set-based approach, since standard operators for continuous variables require a meaningful matching between homologous variables in the parental solutions which is not possible (or at least non-trivial) in this problem. Our recombination algorithm is depicted in Algorithm 2. It creates a set of candidate locations from the union of the individuals being recombined, and makes a sequence of random picks without replacement from this candidate set. The resulting operator is therefore transmitting and assorting, but not necessarily respectful [15]. Besides these operators, our EA uses binary tournament solution, and elitist generational replacement. We have also considered an island version of this EA [1], which divides the population into a number of separate demes (arranged following a certain topology – a bidirectional ring in our case) which evolve in partial isolation, and periodically migrate the best solution to neighboring demes, who accept these in substitution of their current worst solutions. We will denote our EA and our island-based EA as EA and iEA respectively. All algorithms are available in our GitHub repository[4].

5 Experimental Results

The different algorithms described in the previous section have been put to test on a collection of problem instances with different features. These instances and the remaining experimental parameters are described in Sect. 5.1. Subsequently, the numerical results will be reported and analyzed in Sect. 5.2.

[4] https://github.com/Bio4Res/pedestrian-evacuation-optimization

5.1 Experimental Setup

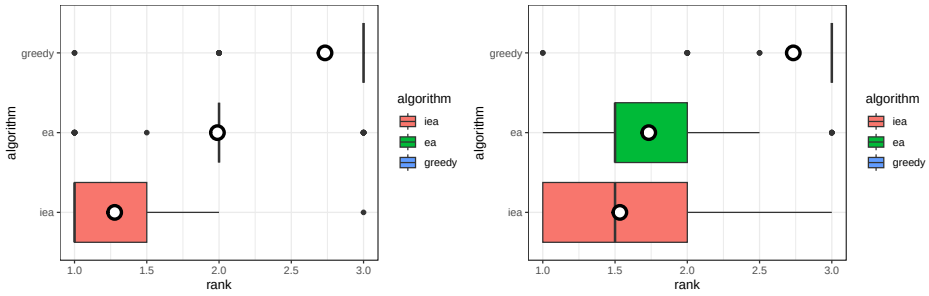
To evaluate the performance of different algorithms, we have generated several environments that simulate evacuation scenarios. Our instance generator discretizes the evacuation area in the same fashion our CA does (see Sect. 3), and places obstacles randomly in the domain, avoiding overlaps and ensuring a minimum distance between them. The obstacles are rectangular and their dimensions are randomly generated as follows: the width of the obstacle can be either one or two cells, if the obstacle is vertical, or between one and 25 cells, if the obstacle is horizontal. The height of the obstacle is inversely proportional to the width, and it can be between one and half of the rows of the domain. The orientation of the obstacle is also randomly chosen, with a 50% probability of being vertical or horizontal. The position of the obstacle is randomly selected, with the condition that the obstacle does not exceed the boundaries of the domain, and that there is a minimum distance of two cells between the obstacle and any other obstacle, so that the agents can always move around them. The purpose of the obstacles is to create a realistic, diverse, and challenging environment for the agents, by obstructing their movement and forcing them to find alternative paths.

We have generated three sets of instances, each containing five environments with different characteristics depending on the number $|O|$ of obstacles:

- *low-density*: $|O| \in \{20, \ldots, 30\}$. A low density of obstacles implies that the agents have more space to move and less chances of colliding with them.
- *mid-density*: $|O| \in \{50, \ldots, 75\}$. A medium density of obstacles means that the agents have less space to move and more chances of colliding with them, but still have some room for maneuvering and finding alternative paths.
- *high-density*: $|O| \in \{100, \ldots, 150\}$. A high density of obstacles results in the agents having very little space to move and very high chances of colliding with them, facing a lot of congestion and bottlenecks in their movement.

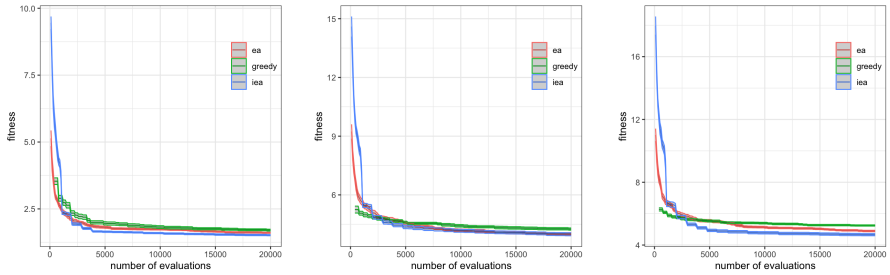
In all cases, the width and height are picked from [40, 50] and [20, 30], the side of the square cells is 0.5m and no exits are initially placed. Hence, evacuation will only proceed through the emergency exits placed by the optimization algorithms. We consider three setting in this regard, namely $k \in \{3, 4, 5\}$ exits. The width of the emergency exits is set to $\omega = 2$m. All the instances are publicly available in our data repository [5]. For each instance, we have randomly generated 1000 initial pedestrian configurations. 20 are used as training set for the optimization algorithms, and the remaining ones are used as test set. In every case we have considered 100 pedestrians. Each of them has a reference velocity $v = 1.3$m/s, a velocity percent $v_p \in [0.5, 1]$, field attraction bias $\phi \in [1.5, 2]$, and crowd repulsion bias $\zeta \in [0.25, 0.5]$. The simulation is run up to $T = 60$s.

Regarding the algorithms, in all cases we consider $maxevals = 20000$. The EA has a population size $\mu = 100$, recombination probability $p_X = 0.9$, mutation probability equivalent to a mutation rate $1/\ell$ per variable, where ℓ is the number of variables, and gaussian mutation amplitude $\gamma = 0.05$. As to the iEA, it considers 4 islands of size $\mu = 25$, and migration frequency of 10 generations. No fine tuning of these parameters has been attempted. For each algorithm, floor plan and number of exits sought, we perform 20 runs.



(a) (b)

Fig. 1: (a) Rank distribution of the different algorithms on the training set. (b) Rank distribution of the best solution of each algorithm on the test set.



(a) (b) (c)

Fig. 2: Evolution of fitness in three of the instances. (a) low-density (b) mid-density (c) high-density

5.2 Results

Table 1 shows the summary of results over the 20 runs of the algorithms. As it can be seen there is a general superiority of iEA over all types of instances and number of exits, and even more clearly for $k \geqslant 4$ exits. This superiority is not just clear on a head-to-head basis with respect to EA and greedy on specific instances, but it is also globally significant. To show this, we rank each algorithm on each problem instance, and determine the distribution of ranks – see Fig. 1a. These ranks show statistically significant differences according to Quade test [14] (Quade $F = 37.351$, p-value $= 1.803e{-}12$). Subsequently, we conduct Holm test with Bonferroni correction [7,10] using iEA as control algorithm. The test is passed against both EA and greedy with p-value $= 7.433e{-}4$. These results indicate that the evolutionary search, and in particular the island-based EA, is capable of effectively navigating the search space and finding solutions that perform satisfactorily on the training set. Fig. 2 shows an example of the evolution of fitness as a function of the number of solution evaluations for the

Table 1: Results of the algorithms (out of 20 runs) on the training set. Each column depicts the best (x^*), median ($\tilde{x}$), mean ($\bar{x}$) and standard error of the mean ($\sigma_{\bar{x}}$). For each instance, the algorithm with the best mean is marked with a star ($\star$), and the remaining algorithms are marked with a symbol that denotes whether the differences are statistically significant at $\alpha = 0.01$ (■), $\alpha = 0.05$ (•), and $\alpha = 0.1$ (○) according to a Wilcoxon rank sum test [17].

	greedy			EA			iEA		
instance	x^*	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	x^*	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	x^*	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$
low-density-1-3	8.109	8.410	8.493 ± 0.045 ■	7.111	7.111	7.171 ± 0.023 ⋆	7.111	7.111	7.184 ± 0.058
low-density-2-3	7.511	8.436	8.662 ± 0.216 ■	6.417	6.467	6.520 ± 0.020 ⋆	6.417	6.488	6.534 ± 0.024
low-density-3-3	9.462	9.584	9.606 ± 0.020	9.462	9.660	9.687 ± 0.035 •	9.409	9.660	9.598 ± 0.030 ⋆
low-density-4-3	11.309	11.309	11.759 ± 0.115 ■	9.259	9.556	9.657 ± 0.069 ■	9.259	9.309	9.438 ± 0.072 ⋆
low-density-5-3	8.510	8.862	8.867 ± 0.035 ■	4.066	4.318	4.411 ± 0.052 ○	4.066	4.315	4.279 ± 0.030 ⋆
mid-density-1-3	13.104	13.482	13.435 ± 0.039 ■	10.709	10.709	10.894 ± 0.118 ⋆	10.709	11.509	11.613 ± 0.210 •
mid-density-2-3	15.306	15.432	15.477 ± 0.046 ■	14.507	14.507	14.507 ± 0.000 ⋆	14.507	14.507	14.507 ± 0.000
mid-density-3-3	24.656	25.007	24.919 ± 0.055 ■	15.555	15.555	15.688 ± 0.061 ⋆	15.555	15.658	15.696 ± 0.057
mid-density-4-3	14.758	15.006	15.016 ± 0.019 ■	11.704	12.058	12.198 ± 0.100	11.704	11.757	12.089 ± 0.128 ⋆
mid-density-5-3	13.656	14.383	14.195 ± 0.107 ■	12.557	12.557	12.557 ± 0.000 ⋆	12.557	12.557	12.642 ± 0.075
high-density-1-3	16.909	17.804	17.736 ± 0.083 ■	15.005	15.005	15.206 ± 0.126 ⋆	15.005	15.005	15.436 ± 0.144
high-density-2-3	17.556	17.607	17.813 ± 0.110 ■	17.556	17.556	18.034 ± 0.477	17.556	17.556	17.556 ± 0.000 ⋆
high-density-3-3	25.757	25.982	26.055 ± 0.072 ■	18.757	19.307	19.341 ± 0.097	18.757	19.005	19.218 ± 0.111 ⋆
high-density-4-3	17.205	17.831	17.629 ± 0.084 ■	15.105	15.356	15.959 ± 0.216 ⋆	15.105	15.306	16.069 ± 0.236
high-density-5-3	13.406	13.508	13.613 ± 0.056 ■	13.006	13.006	13.325 ± 0.095	13.006	13.006	13.129 ± 0.038 ⋆
low-density-1-4	1.316	1.510	1.492 ± 0.025 ⋆	1.333	1.668	1.700 ± 0.060 •	1.263	1.738	1.611 ± 0.051
low-density-2-4	3.015	3.369	3.359 ± 0.040 ■	2.472	2.690	2.802 ± 0.084 •	2.378	2.577	2.588 ± 0.031 ⋆
low-density-3-4	2.216	2.888	2.793 ± 0.076 ■	1.827	2.110	2.152 ± 0.039 ■	1.808	1.967	2.078 ± 0.131 ⋆
low-density-4-4	2.774	3.246	3.171 ± 0.041 ■	2.006	2.145	2.144 ± 0.024	1.869	2.122	2.123 ± 0.034 ⋆
low-density-5-4	1.100	1.170	1.166 ± 0.009 ■	1.100	1.189	1.205 ± 0.023 ■	1.053	1.089	1.106 ± 0.013 ⋆
mid-density-1-4	3.515	3.963	3.926 ± 0.041 ■	2.867	3.342	3.367 ± 0.048 ■	2.961	3.190	3.213 ± 0.045 ⋆
mid-density-2-4	5.261	5.984	5.808 ± 0.078 ○	5.261	5.637	5.769 ± 0.118	5.261	5.470	5.618 ± 0.106 ⋆
mid-density-3-4	6.161	6.513	6.523 ± 0.046 ■	5.610	6.111	6.100 ± 0.041 ■	5.610	5.860	5.870 ± 0.044 ⋆
mid-density-4-4	5.011	5.188	5.221 ± 0.046 ■	2.929	3.120	3.156 ± 0.026 ■	2.666	3.062	3.005 ± 0.041 ⋆
mid-density-5-4	5.114	5.345	5.351 ± 0.030 •	4.967	5.188	5.266 ± 0.044	4.915	5.263	5.247 ± 0.036 ⋆
high-density-1-4	4.611	5.090	5.151 ± 0.082 ■	4.110	4.487	4.564 ± 0.055	4.110	4.440	4.484 ± 0.068 ⋆
high-density-2-4	7.507	7.663	7.742 ± 0.057 ■	7.112	7.509	7.492 ± 0.047 ⋆	7.112	7.360	8.889 ± 0.523
high-density-3-4	7.009	7.259	7.229 ± 0.041 ■	6.859	7.209	7.189 ± 0.038 ■	6.712	6.985	6.980 ± 0.031 ⋆
high-density-4-4	5.206	5.487	5.614 ± 0.073 ■	4.711	4.944	5.098 ± 0.090 ⋆	4.519	4.786	5.424 ± 0.193
high-density-5-4	5.513	5.863	5.898 ± 0.032 ■	5.014	5.816	5.787 ± 0.070 •	4.662	5.640	5.523 ± 0.083 ⋆
low-density-1-5	0.985	1.053	1.055 ± 0.010 ■	0.968	1.025	1.035 ± 0.012 •	0.948	0.979	1.005 ± 0.012 ⋆
low-density-2-5	1.241	1.455	1.423 ± 0.017 ⋆	1.295	1.561	1.608 ± 0.050 ■	1.151	1.611	1.513 ± 0.052
low-density-3-5	1.417	1.691	1.707 ± 0.035 ■	1.419	1.608	1.609 ± 0.020 •	1.352	1.531	1.520 ± 0.024 ⋆
low-density-4-5	1.130	1.263	1.253 ± 0.019 •	1.192	1.250	1.254 ± 0.012 ■	1.062	1.168	1.198 ± 0.028 ⋆
low-density-5-5	0.942	0.964	0.965 ± 0.002 ■	0.924	0.945	0.950 ± 0.005 ○	0.908	0.935	0.941 ± 0.006 ⋆
mid-density-1-5	1.552	1.812	1.787 ± 0.025 ■	1.265	1.534	1.536 ± 0.029 ○	1.323	1.416	1.489 ± 0.040 ⋆
mid-density-2-5	3.520	3.971	3.980 ± 0.047 •	3.318	3.820	4.067 ± 0.120	3.272	3.744	3.838 ± 0.081 ⋆
mid-density-3-5	3.515	4.309	4.255 ± 0.057 •	3.515	4.036	4.016 ± 0.058	3.513	3.912	3.990 ± 0.078 ⋆
mid-density-4-5	2.064	2.380	2.355 ± 0.050 ■	1.298	1.505	1.522 ± 0.035 ○	1.201	1.366	1.433 ± 0.035 ⋆
mid-density-5-5	2.215	2.393	2.517 ± 0.063 ■	1.467	1.771	1.774 ± 0.032 ■	1.460	1.666	1.657 ± 0.022 ⋆
high-density-1-5	2.112	2.411	2.419 ± 0.051 •	2.012	2.385	2.504 ± 0.084 •	1.760	2.090	2.201 ± 0.078 ⋆
high-density-2-5	4.967	5.410	5.479 ± 0.062	4.811	5.511	5.507 ± 0.093	4.465	5.514	5.384 ± 0.101 ⋆
high-density-3-5	4.911	5.237	5.246 ± 0.039 ■	4.513	4.914	4.895 ± 0.045 ■	4.113	4.611	4.655 ± 0.068 ⋆
high-density-4-5	3.218	3.573	3.599 ± 0.044 ■	2.010	2.511	2.468 ± 0.069 ■	1.905	2.126	2.234 ± 0.072 ⋆
high-density-5-5	2.060	2.410	2.363 ± 0.047 ■	1.520	1.864	1.881 ± 0.042	1.512	1.812	1.829 ± 0.034 ⋆

Table 2: Test results of the best solution found by each algorithm during training. The meaning of symbols is the same as in Table 1.

	greedy				EA				iEA			
instance	x^*	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$		x^*	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$		x^*	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	
low-density-1-3	2.000	9.001	8.816 ± 0.093	■	0.972	8.001	7.866 ± 0.087	⋆	0.972	8.001	7.866 ± 0.087	
low-density-2-3	2.015	8.011	8.278 ± 0.084	■	1.010	7.521	7.680 ± 0.083	⋆	1.010	7.521	7.680 ± 0.083	
low-density-3-3	3.013	11.001	10.839 ± 0.098		3.013	11.001	10.839 ± 0.098		2.009	11.001	10.673 ± 0.097	⋆
low-density-4-3	4.067	12.026	12.687 ± 0.110	■	2.000	10.010	10.399 ± 0.099	⋆	2.000	10.010	10.399 ± 0.099	
low-density-5-3	1.005	9.011	8.962 ± 0.091	■	0.918	6.001	5.958 ± 0.074	⋆	0.918	6.001	5.958 ± 0.074	
mid-density-1-3	5.011	15.001	14.707 ± 0.112	■	3.010	11.010	11.340 ± 0.103	⋆	3.010	11.010	11.34 ± 0.103	
mid-density-2-3	6.010	16.001	15.970 ± 0.110	■	5.010	15.011	15.391 ± 0.112	⋆	5.010	15.011	15.391 ± 0.112	
mid-density-3-3	13.010	25.002	25.242 ± 0.137	■	8.001	17.009	17.296 ± 0.119	⋆	8.001	17.009	17.296 ± 0.119	
mid-density-4-3	4.028	16.001	15.648 ± 0.117	■	4.019	13.001	12.831 ± 0.107	⋆	4.019	13.001	12.831 ± 0.107	
mid-density-5-3	4.011	15.001	15.022 ± 0.112	■	3.000	13.001	12.954 ± 0.101	⋆	3.000	13.001	12.954 ± 0.101	
high-density-1-3	7.010	17.001	16.852 ± 0.117	■	6.011	15.010	15.336 ± 0.108	⋆	6.011	15.010	15.336 ± 0.108	
high-density-2-3	7.011	18.001	18.038 ± 0.118	⋆	7.011	18.001	18.038 ± 0.118		7.011	18.001	18.038 ± 0.118	
high-density-3-3	16.001	27.002	27.097 ± 0.137	■	8.009	20.018	20.685 ± 0.127	⋆	8.009	20.018	20.685 ± 0.127	
high-density-4-3	7.010	18.510	18.585 ± 0.120	■	7.013	17.010	17.227 ± 0.115	⋆	7.013	17.010	17.227 ± 0.115	
high-density-5-3	5.011	14.011	14.365 ± 0.107		5.011	14.011	14.35 ± 0.109	⋆	5.011	14.011	14.350 ± 0.109	
low-density-1-4	0.819	2.000	1.915 ± 0.035	⋆	0.863	2.010	2.098 ± 0.039	■	0.809	2.010	2.056 ± 0.037	■
low-density-2-4	0.896	4.010	3.982 ± 0.059	■	0.852	3.011	3.039 ± 0.050	⋆	0.874	3.011	3.144 ± 0.054	
low-density-3-4	0.853	3.009	2.973 ± 0.049	■	0.863	2.029	2.669 ± 0.049	⋆	0.875	3.001	2.885 ± 0.049	■
low-density-4-4	0.917	4.001	3.875 ± 0.060	■	0.809	2.042	2.714 ± 0.049	⋆	0.906	3.010	3.164 ± 0.053	■
low-density-5-4	0.754	1.062	1.664 ± 0.030	■	0.754	1.021	1.422 ± 0.025		0.787	1.024	1.417 ± 0.024	⋆
mid-density-1-4	0.917	4.020	4.429 ± 0.063	○	0.885	4.020	4.446 ± 0.064	○	0.917	4.014	4.300 ± 0.064	⋆
mid-density-2-4	0.983	6.010	6.329 ± 0.077	⋆	0.983	6.010	6.329 ± 0.077		0.983	6.010	6.329 ± 0.077	
mid-density-3-4	2.000	7.013	7.408 ± 0.082	■	1.009	6.009	6.216 ± 0.075	⋆	1.009	6.009	6.216 ± 0.075	
mid-density-4-4	0.994	6.010	6.166 ± 0.074	■	0.907	4.010	4.201 ± 0.062		0.939	4.010	4.177 ± 0.062	⋆
mid-density-5-4	1.010	7.000	6.691 ± 0.078	■	0.994	7.001	6.904 ± 0.081	■	0.885	6.001	5.617 ± 0.070	⋆
high-density-1-4	0.929	5.014	5.449 ± 0.070	■	0.907	5.010	4.978 ± 0.068	⋆	0.907	5.010	4.978 ± 0.068	
high-density-2-4	1.029	8.010	8.432 ± 0.088		0.972	8.010	8.251 ± 0.089	⋆	0.972	8.010	8.251 ± 0.089	
high-density-3-4	0.962	8.009	8.281 ± 0.088	■	0.972	7.020	7.627 ± 0.083	⋆	1.090	8.001	8.035 ± 0.087	■
high-density-4-4	1.000	6.014	6.496 ± 0.078	■	0.961	6.009	6.021 ± 0.077	⋆	1.005	6.010	6.063 ± 0.074	
high-density-5-4	1.037	7.011	7.151 ± 0.079	■	0.972	6.011	6.102 ± 0.073	■	1.000	6.001	5.851 ± 0.071	⋆
low-density-1-5	0.775	1.022	1.482 ± 0.026	■	0.754	1.010	1.242 ± 0.019	■	0.743	1.003	1.174 ± 0.017	⋆
low-density-2-5	0.743	1.041	1.570 ± 0.027	⋆	0.732	1.042	1.638 ± 0.030		0.786	1.042	1.630 ± 0.030	
low-density-3-5	0.852	2.012	2.217 ± 0.043	■	0.863	2.009	2.087 ± 0.039	•	0.819	2.001	1.988 ± 0.038	⋆
low-density-4-5	0.797	1.027	1.486 ± 0.026		0.775	1.021	1.475 ± 0.026	⋆	0.743	1.027	1.540 ± 0.028	
low-density-5-5	0.742	1.011	1.23 0± 0.019	■	0.655	0.970	1.046 ± 0.012	⋆	0.689	0.971	1.06 0± 0.013	
mid-density-1-5	0.808	2.010	2.119 ± 0.039	■	0.831	2.000	1.922 ± 0.036		0.809	2.000	1.906 ± 0.036	⋆
mid-density-2-5	0.984	5.000	4.713 ± 0.069	■	0.896	4.010	4.231 ± 0.064	⋆	0.917	4.017	4.452 ± 0.066	•
mid-density-3-5	1.000	5.009	5.243 ± 0.069	■	1.000	5.009	5.243 ± 0.069	■	0.972	5.000	4.785 ± 0.067	⋆
mid-density-4-5	0.820	2.029	2.577 ± 0.046	■	0.797	1.044	1.703 ± 0.032	⋆	0.765	1.069	1.844 ± 0.035	■
mid-density-5-5	0.884	3.011	3.119 ± 0.052	■	0.830	2.011	2.182 ± 0.041		0.830	2.011	2.142 ± 0.039	⋆
high-density-1-5	0.863	2.022	2.514 ± 0.045	■	0.842	2.028	2.577 ± 0.045	■	0.863	2.014	2.218 ± 0.042	⋆
high-density-2-5	1.010	7.000	6.700 ± 0.081	■	0.928	6.000	5.803 ± 0.074	⋆	0.950	6.001	5.943 ± 0.074	
high-density-3-5	0.994	6.009	6.148 ± 0.075	■	0.950	6.000	5.755 ± 0.073	■	0.929	5.009	5.201 ± 0.070	⋆
high-density-4-5	0.885	3.036	3.581 ± 0.059	■	0.896	3.000	2.802 ± 0.048	⋆	0.852	3.000	2.819 ± 0.049	
high-density-5-5	0.830	2.021	2.436 ± 0.044		0.830	2.024	2.511 ± 0.045	○	0.831	2.021	2.379 ± 0.042	⋆

three algorithms. As it can be seen, greedy often starts with good quality solutions, typically better than those of EA and iEA for a similar computational effort. However, in the long run the evolutionary approaches are capable of outperforming the greedy heuristic.

Subsequently, we move to the test phase. To this end, we select the solution that has the best fitness on each instance for each algorithm, and evaluate it on all the test cases. Table 2 shows the resulting results. Note that iEA remains superior in general, and both EAs outperform greedy. However, the differences are less marked. This is better seen in Fig. 1b, where the rank distribution of the different algorithms according to the performance of their solution on the test set is shown. Again, these ranks show statistically significant differences according to Quade test (Quade $F = 50.376$, p-value $= 2.618e{-}15$), and iEA remains the algorithm with the best mean rank, so it is chosen as control algorithm for Holm test. Now, the test is passed against greedy (p-value ≈ 0), but not against EA (p-value $= 3.428e{-}1$). We believe this may be an indication that the training set is not large enough and therefore iEA may be overfitting its solutions.

6 Conclusions

Optimizing the placement of emergency exits in indoor environments is not just a problem of importance for public safety, but also poses a challenging optimization task. We have conducted a comparative analysis of two different optimization approaches, namely an iterated greedy heuristic and an evolutionary algorithm (in two variants, both panmictic and island-based). This analysis indicates the superiority of the evolutionary approaches, both on the training and test phases, underpinning the need for powerful global optimization techniques in this context. It also hints at the need of using larger training sets, which of course will have a toll on computational cost. This makes a strong case for directing effort into solutions of computational nature (such as parallel computing) and solutions of algorithmic nature (e.g., lightweight simulations or surrogate models [11]).

In addition to the research directions sketched above, it is clear that the evacuation scenario can be enriched with additional layers of complexity. While we have here assumed situations of orderly evacuation as an initial base case, we can go on to consider situations in which the cause of the emergency does pose a visible threat (e.g., a rampant fire, or ongoing explosions) that might disrupt the evacuation process or the flow of people. Such scenarios may be in need of more sophisticated approaches, and this work has paved the way for hybrid approaches that combine greedy components within an evolutionary search engine. Work is in progress in this area.

Acknowledgments The authors thank the Supercomputing and Bioinnovation Center (SCBI) of the University of Malaga for their provision of computational resources (the Picasso supercomputer http://www.scbi.uma.es).

References

1. Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. IEEE Transactions on Evolutionary Computation **6**(5), 443–462 (2002)
2. Bellomo, N., Bellouquid, A., Knopoff, D.: From the Microscale to Collective Crowd Dynamics. Multiscale Modeling & Simulation **11**(3), 943–963 (Jan 2013)
3. Cao, R.F., Lee, E.W.M., Xie, W., Gao, D.L., Chen, Q., Yuen, A.C.Y., Yeoh, G.H., Yuen, R.K.K.: Development of an agent-based indoor evacuation model for local fire risks analysis. Journal of Safety Science and Resilience **4**(1), 75–92 (Mar 2023)
4. Chen, J., Shi, T., Li, N.: Pedestrian evacuation simulation in indoor emergency situations: Approaches, models and tools. Safety Science **142**, 105378 (2021)
5. Cotta, C., Gallardo, J.E.: Instance dataset for the pedestrian evacuation problem (2023), https://osf.io/cnh7u/
6. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische mathematik **1**(1), 269–271 (1959)
7. Dunn, O.J.: Multiple Comparisons among Means. Journal of the American Statistical Association **56**(293), 52–64 (Mar 1961)
8. Golas, A., Narain, R., Lin, M.C.: Continuum modeling of crowd turbulence. Physical Review E **90**(4), 042816 (Oct 2014)
9. Haghani, M.: Optimising crowd evacuations: Mathematical, architectural and behavioural approaches. Safety Science **128**, 104745 (Aug 2020)
10. Holm, S.: A Simple Sequentially Rejective Multiple Test Procedure. Scandinavian Journal of Statistics **6**(2), 66–70 (1979)
11. Jin, Y.: Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation **1**(2), 61–70 (Jun 2011)
12. Li, Z., Xu, C., Bian, Z.: A force-driven model for passenger evacuation in bus fires. Physica A: Statistical Mechanics and its Applications **589**, 126591 (Mar 2022)
13. Martinez-Gil, F., Lozano, M., García-Fernández, I., Fernández, F.: Modeling, Evaluation, and Scale on Artificial Pedestrians: A Literature Review. ACM Computing Surveys **50**(5), 72:1–72:35 (Sep 2017)
14. Quade, D.: Using Weighted Rankings in the Analysis of Complete Blocks with Additive Block Effects. Journal of the American Statistical Association **74**(367), 680–683 (Sep 1979)
15. Radcliffe, N.J.: The algebra of genetic algorithms. Annals of Mathematics and Artificial Intelligence **10**(4), 339–384 (Dec 1994)
16. Shi, M., Lee, E.W.M., Ma, Y.: A dynamic impatience-determined cellular automata model for evacuation dynamics. Simulation Modelling Practice and Theory **94**, 367–378 (Jul 2019)
17. Wilcoxon, F.: Individual Comparisons by Ranking Methods. Biometrics Bulletin **1**(6), 80 (Dec 1945)
18. Zheng, Y., Li, X.G., Jia, B., Jiang, R.: Simulation of pedestrians' evacuation dynamics with underground flood spreading based on cellular automaton. Simulation Modelling Practice and Theory **94**, 149–161 (Jul 2019)