Optimization-Driven Workplace Evacuation using Evolutionary Algorithms and Derivative-Free Methods

Carlos Cotta
ccottap@lcc.uma.es
ITIS Software, University of Málaga
Málaga, Spain

ABSTRACT

We consider the problem of optimizing the evacuation of a workplace environment by deciding the best arrangement of emergency exits. To this end, we consider a simulation-based approach that relies on the use of cellular automata to model the collective behavior of the crowd. In order to obtain problem instances more akin to realistic workplace environments, a problem instance generator based on L-attributed grammars is devised and described in detail. Subsequently, we consider the use of evolutionary algorithms, an iterated greedy heuristic, and Nelder-Mead method to solve the problem. In-depth experimentation is reported. It is shown that the evolutionary algorithm is superior during training and that Nelder-Mead method is also competitive during test, raising interesting prospects for future hybrid strategies.

CCS CONCEPTS

• **Computing methodologies** → **Genetic algorithms**; **Randomized search**; • **Theory of computation** → **Design and analysis of algorithms**.

KEYWORDS

Workplace Evacuation, Cellular Automata, Attribute Grammar, Evolutionary Algorithms, Nelder-Mead Algorithm.

ACM Reference Format:
Carlos Cotta. 2024. Optimization-Driven Workplace Evacuation using Evolutionary Algorithms and Derivative-Free Methods. In *2024 8th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence (ISMSI) (ISMSI 2024), April 24–25, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3665065.3665079

1 INTRODUCTION

According to OECD data, people actively working spend around 1750 hours a year at their jobs [16], which represents approximately 31% of their waking hours. This figure is consistent with other sources [17] and suggests that about a third of our lives are spent in work settings. Since such environments are not exempt from risks and emergencies, it should become clear that workplace evacuation planning is of paramount relevance. This planning must consider numerous dimensions and procedures in order to minimize potential risks, and mitigate the impact of unforeseen events. We are here particularly concerned with the design of evacuation routes, and more precisely with the strategic arrangement of emergency exits to facilitate fast and safe evacuation. This falls within the general framework of crowd evacuation optimization [8].

Crowd evacuation problems can be approached from many perspectives, usually emanating from the precise tools used to model the environment and the behavior of the crowd during an emergency; see, for example, [2]. Cellular automata (CA) are one of these mathematical tools that can be used in this context, and that shine because of their simplicity, computational efficiency, and the richness of the behaviors they can simulate [6, 10]. We have previously approached the problem from this direction, defining a CA model that incorporates a parameterizable balance between rational search for the nearest exit and crowding aversion [3]. However, this model was tested on random unstructured environments that arguably lacked the features of realistic scenarios. To account for this, we are here considering environments that resemble real-world workplaces. More precisely, we devise a problem instance generator based on L-attributed grammars [11] that can be used to produce diverse problem instances that resemble office floors. We have also extended the algorithm suite for solving the problem with derivative-free methods, and used these to tackle the optimization problem in scenarios of orderly evacuation.

The remainder of this work is organized as follows: we start by providing a formal definition of the problem (Section 2.1) and how our CA models the behavior of the crowd evacuating the environment (Section 2.2). Then, we carefully detail the problem instance generator (Section 3.1) and describe the optimization methods considered to solve the problem (Section 3.2). Subsequently, we report the results of an extensive experimentation (Section 4). We close this work with a discussion of key findings and avenues for future research (Section 5).

2 PRELIMINARIES

2.1 Problem Definition

Generally speaking, the problem we are considering is determining where to place emergency exits in a certain enclosure so that the latter can be quickly evacuated in case of an emergency. Thus, modeling this problem involves two aspects: (i) formalizing the indoor space from which the evacuation will be attempted, and (ii) defining appropriate metrics for assessing whether the outcome of the process is satisfactory or not. Regarding the first aspect, we will describe the geometry of the environment $\mathcal{E}$ as a tuple (w, h, A, O), where w, h are the dimensions of a rectangular floor plan within which $\mathcal{E}$ is embedded, A is a collection of pre-existing accesses that

ISMSI 2024, April 24–25, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-1729-1/24/04
https://doi.org/10.1145/3665065.3665079

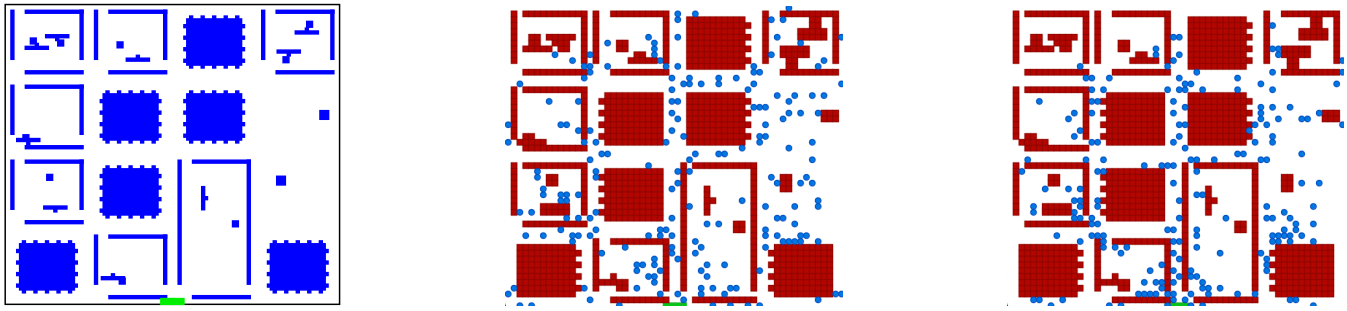


Figure 1: (a) Example scenario created with the attribute grammar described in Section 3.1. An exit is depicted in green at the bottom of the scenario. Snapshots at different time stamps of a CA-based simulation of this scenario using $n = 200$, $c = 0.5$m, $\phi = 4$ and $\psi = 0$. (b) $t = 0$s (c) $t = 10$s.

allow one to exit the environment, and O is a collection of obstacles representing areas of the floor plan that cannot be traversed. Note that A and O might be empty.

Now, let $k \in \mathbb{N}^+$ be a strictly positive integer that indicates the number of emergency exits that we are interested in placing. Let $\Sigma = \{e_1, \ldots, e_k\}$ be the collection of such exits (which we consider to be doors in some parts of the perimeter of the environment), and let $\mathcal{E}_\Sigma$ be the environment resulting from adding these emergency exits to the list of accesses. To determine whether this extended environment is well-suited for evacuation, we will follow a simulation-based approach, whereby we will populate the environment with a crowd of people ξ and simulate the evacuation by some means (the model we consider for this purpose will be described in next subsection). Whatever this procedure is, at the end we gather some statistics to determine how successful the evacuation was. Obviously, the main factor is the number of people $\xi^+ \subseteq \xi$ that could exit the environment within some allocated time. This is a value that should be maximized, or conversely, the number of people $\xi^- = \xi \setminus \xi^+$ who could not evacuate the place (non-evacuees, henceforth) should be minimized. In the event of a fully satisfactory evacuation, we would also be interested in making the evacuation as fast and safe as possible. Although this can be related to the average evacuation time for all members of the crowd, it is important to also consider the evacuation time of the last person to leave, since this is the person who marks the difference between a full evacuation and a partial evacuation. However, if the evacuation was not completely successful, it is important to measure to what extent this occurred. Following an analogy with the previous case, we can consider the average distance of non-evacuees to their nearest exit, and the minimum distance from any of them to an exits (which again marks the difference between having one more non-evacuee or not). We can then integrate everything into a single scalar value f_Σ defined as

$$\begin{aligned} f_\Sigma = |\xi^-| + \quad & [\xi^- = \emptyset] \cdot \left(w_1^+ \max_{i \in \xi} t_i + w_2^+ \sum_{i \in \xi} t_i \right) + \\ + \quad & [\xi^- \neq \emptyset] \cdot \left(w_1^- \min_{i \in \xi^-} d_i + w_2^- \sum_{i \in \xi^-} d_i \right) \end{aligned} \tag{1}$$

where t_i (resp. d_i) is the time of evacuation (resp. distance to closest exit) of the i-th person, and the weights $w_{1|2}^{+|-}$ are picked to prioritize the different metrics ($w_1^+ = T^{-1}$, $w_2^+ = (nT^2)^{-1}$, $w_1^- = D^{-1}$, $w_2^- = (|\xi^-|D^2)^{-1}$, where T is the maximum allowed evacuation time, and D is the diagonal length of $\mathcal{E}$) [3].

2.2 Crowd Modeling

The crowd behavior has been modeled using a probabilistic cellular-automaton (CA) [13] approach. To this end, the environment considered is first discretized in a grid of square cells (let c be the side length of such cells). As a result, we obtain a lattice of cells that are either blocked or traversable. This lattice is populated with a crowd of agents, each of whom will occupy a traversable cell. Initially, the position of each agent is selected at random among all traversable cells (with the obvious restriction that at most one agent can be placed on any given square). Subsequently, agents will move (or try to move) to other traversable squares using a random sweep policy [18]. Such moves are determined according to the *desirability* of each cell. This is a numerical quantity (whose calculation is described below) that determines how appropriate is a position from the point of view of the agent, taking into account both physical and psychological factors. Let $\hat{\mathcal{D}}_{ij}$ be the desirability of a square cell indexed by (i, j): the agent at position (i, j) will then move to any empty (unoccupied by another agent) traversable cell in its Moore neighborhood with probability proportional to its desirability. This procedure is repeated until all agents have reached any cell square marked as exit, or until the maximum time T is reached (the real time to which a single CA step corresponds is calibrated taking into account the cell side length and a known displacement speed v of agents).

The computation of the desirability of a cell rests on two key notions: the *attraction* and the *repulsion*:

- The attraction measures the goodness of each cell for evacuation purposes. It is determined on the basis of a static floor field $\mathcal{A}$, namely, a potential field calculated according to the geometry of the environment. The value $\mathcal{A}_{ij}$ of the static floor field at position (i, j) is $\mathcal{A}_{ij} = 1 - d_{ij}/\max_{rs} d_{rs}$, where d_{ij} is the distance from cell (i, j) to its nearest exit (computed using Dijkstra's algorithm [5]).

$$
\begin{array}{rlll}
w > 8\text{:} & E[x,y,w,h] & \to_3 & E[x,y,w/2,h]E[x+w/2,y,w/2,h] \\
h > 8\text{:} & E[x,y,w,h] & \to_3 & E[x,y,w,h/2]E[x,y+h/2,w,h/2] \\
w > 8 \wedge h > 8\text{:} & E[x,y,w,h] & \to_4 & E[x,y,w/2,h/2]E[x+w/2,y,w/2,h/2]E[x,y+h/2,w/2,h/2]E[x+w/2,y+h/2,w/2,h/2] \\
w \leqslant 8 \vee h \leqslant 8\text{:} & E[x,y,w,h] & \to_1 & R[x+2\zeta, y+2\zeta, w-4\zeta, h-4\zeta] \\
\text{true:} & R[x,y,w,h] & \to_3 & O[x,y,w,h] \\
\text{true:} & R[x,y,w,h] & \to_1 & \mathit{solid}[x+\zeta+\rho\cdot(w-5\zeta), y+\zeta+\rho\cdot(h-5\zeta), 3\zeta, 3\zeta] \\
w < 8 \wedge h < 8\text{:} & R[x,y,w,h] & \to_2 & \mathit{table}[x,y,w,h] \\
w > h\text{:} & O[x,y,w,h] & \to_1 & \mathit{office}_1[x,y,w,h] \\
w \leqslant h\text{:} & O[x,y,w,h] & \to_1 & \mathit{office}_2[x,y,w,h]
\end{array}
$$

Figure 2: Production rules of the attribute grammar for generating office-like environments.

- The repulsion measures the psychological aversion to enter intro closely packed multitudes. Repulsion $\mathcal{R}_{ij}$ of a cell (i, j) is calculated as $\mathcal{R}_{ij} = (1 + \omega_{ij})^{-1}$, where ω_{ij} is the number of accessible neighbors (i.e., traversable and unoccupied) of the cell (i, j).

These two measures are combined into a raw-desirability value $\mathcal{D}_{ij} = \exp(\phi \cdot \mathcal{A}_{ij} - \psi \cdot \mathcal{R}_{ij})$, where ϕ, ψ are two coefficients weighting the contribution of each measure. The actual desirability $\hat{\mathcal{D}}_{ij}$ is subsequently computed as the excess raw-desirability of a position with respect to the minimum desirability of any other cell the agent could move to (we consider the use of additive smoothing by adding some very small ϵ so that all cells have a non-zero probability).

Using this model, agents are capable of navigating the environment towards the exit. Figure 1 shows two snapshots of a simulation in which it can be seen that starting from a random distribution at $t = 0$ (Figure 1b), the agents start to organize into evacuation paths towards the exit (Figure 1c).

3 MATERIALS AND METHODS

3.1 Environment Generation with Attribute Grammars

The workplace environments we are focusing on in this work respond to the the typical configuration of company office spaces, that is, an indoor enclosure featuring areas with open layout as well as cubicles or office rooms, and including furniture such as meeting tables, desks, and cabinets. To avoid the need to manually design such environments for the experimentation, we have devised a problem instance generator that can be used to produce as many instances as required. This is done using attribute grammars [12].

An attribute grammar is an extension of standard generative grammars that captures semantic information by defining attributes of terminal/nonterminal symbols and augmenting production rules with semantic rules stating how the values of these attributes are computed. In our case, we consider an L-attributed grammar [11], defined by an underlying context-free grammar $G(S, N, T, R)$ with inherited attributes (i.e. the attributes of a symbol are computed from those of its ancestor and/or left siblings in the parse tree). We will use the following notation:

- $A[\alpha_A]$ denotes a terminal/nonterminal symbol $A \in N \cup T$ with attributes $\alpha_A = \alpha_1, \ldots \alpha_{n_A}$.
- $\phi(\alpha_A) : A[\alpha_A] \to_w X_1[f_1(\alpha_A)] \ldots X_m[f_m(\alpha_A)]$ is a production rule where $A \in N$ is a nonterminal, $X_1 \ldots X_m$ can be terminals or nonterminals, $f_i(\cdot)$ is a function indicating how to compute the attribute values of X_i, w is the weight of the rule (used to control the relative frequency of application of a certain rule whenever several rules are applicable), and $\phi(\cdot)$ is a logical predicate stating a precondition for the application of the rule.

Specifically, paying attention to the scenarios we aim to generate, we consider a grammar defined by:

- The nonterminal symbols $N = \{E, R\}$, where
  - E represents a generic subspace that can be recursively subdivided in two of four pieces. This symbol is also the initial axiom.
  - R represents a virtual room, that is, a space that cannot be further subdivided but used in some specific way.
  - O represents an office, that is, a space surrounded with walls (but accessible through doors from all sides) and containing some furniture.
- The terminal symbols are $T = \{\mathit{solid}, \mathit{table}, \mathit{office}_{1,2}\}$, where
  - *solid* represents a small solid object of square shape.
  - *table* represents a conference table with chairs around it.
  - office_1, and office_2 represent the particular arrangement of furniture within an office, with desks aligned horizontally or vertically respectively.

All symbols have four attributes, namely x, y, w, h, representing a position (x, y), a width w, and a height h. Let W and H be two constants that represent the total width and height of the environment. Then, the initial axiom is $E[0, 0, W, H]$. Subsequently, the production rules of the grammar are given in Figure 2. Therein, a few additional constants are used, such as

- ζ: size of a small object such as a chair. It is also used as a reference unit for defining corridors whose width is some multiple of this size (e.g. see rule #4 in Figure 2), as well as other elements described below.
- ρ: a random ephemeral constant uniformly distributed in [0,1).

Each $\mathit{table}[x, y, w, h]$ terminal translates into a conference table equivalent to a $\mathit{solid}[x + 3\zeta, y + 3\zeta, w - 6\zeta, h - 6\zeta]$ surrounded by chairs (solid objects of size $\zeta \times \zeta$) spaced ζ apart. As to the offices, their surrounding walls have a thickness ζ, doors have size 2ζ, and the interiors comprise a desk (of size $4\zeta \times \zeta$), a chair, and a cabinet (of size $2\zeta \times 2\zeta$). This furniture is randomly located inside the $[x, y, w, h]$ area, and with 50% probability the office is assumed to be double-desk, being the second set of furniture located in a reflected position with respect to the first one.

Table 1: Training results on each instance (20 runs). 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 (⋆), and the remaining algorithms are marked with a symbol that denotes that the differences are statistically significant at $\alpha = 0.01$ (■), $\alpha = 0.05$ (•), and $\alpha = 0.1$ (○) according to a Wilcoxon ranksum test [19].

	ea				greedy				nelder-mead			
instance	x^*	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$		x^*	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$		x^*	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	
office-1	0.469	0.482	0.480 ± 0.001		0.478	0.481	0.481 ± 0.001		0.468	0.480	0.479 ± 0.001	⋆
office-2	0.432	0.438	0.438 ± 0.001	⋆	0.445	0.451	0.451 ± 0.001	■	0.437	0.446	0.446 ± 0.001	■
office-3	0.461	0.470	0.471 ± 0.001	⋆	0.464	0.473	0.472 ± 0.001		0.467	0.478	0.477 ± 0.001	■
office-4	0.480	0.490	0.489 ± 0.001	⋆	0.485	0.496	0.496 ± 0.001	■	0.488	0.500	0.499 ± 0.001	■
office-5	0.415	0.424	0.424 ± 0.001	⋆	0.423	0.431	0.431 ± 0.001	■	0.418	0.432	0.432 ± 0.002	■
office-6	0.466	0.471	0.471 ± 0.000	⋆	0.469	0.477	0.476 ± 0.001	■	0.471	0.481	0.480 ± 0.001	■
office-7	0.409	0.415	0.415 ± 0.001	⋆	0.412	0.416	0.417 ± 0.001		0.411	0.424	0.422 ± 0.001	■
office-8	0.523	0.528	0.528 ± 0.001	⋆	0.526	0.532	0.531 ± 0.001		0.527	0.542	0.541 ± 0.001	■

We have used this framework for generating a number of test instances (using $\zeta = .25m$, and picking values for the initial area from $W \in \{20, 30\}$ and $H \in \{25, 30\}$ or vice versa) which can be used for experimentation (an example is shown in Figure 1a). These instances are publicly available in our data repository[1].

3.2 Optimization Approaches

Given a pair $(\mathcal{E}, k)$ defining an instance of the problem, the solution sought is the placement of k exits in the perimeter of $\mathcal{E}$ so as to minimize the objective function described in Equation (1). Here, we assume that exits have a fixed size $l = 2$m, and therefore it suffices to indicate for each exit an anchor point along the perimeter. For simplicity, we can express each anchor point as a number in $[0, 1]$, indicating the displacement with respect to a certain reference point as a fraction of the perimeter length. In this way, our solution is a vector $\langle \lambda_1, \ldots, \lambda_k \rangle \in [0, 1]^k$. To solve this continuous optimization problem, we have considered the following approaches:

- ea: an evolutionary algorithm that uses binary tournament, elitist generational replacement, set-based recombination (picking a subset of k elements from the union of positions provided by each parent), and Gaussian mutation (using a certain fixed amplitude σ) [3].
- greedy: an iterated greedy method that repeatedly constructs a solution by incrementally placing exits in the location that provides the best value for the objective function (picking a random starting point along the perimeter and successively considering all locations spaced by the emergency exit length; this procedure is repeated for each exit to be placed, and the whole heuristic is iterated until exhausting the budget of function calls) [3].
- nelder-mead: an iterated version of Nelder-Mead method [15]. The Nelder-Mead method is a derivative-free optimization technique that optimizes a k-dimensional continuous function by keeping a polytope of $k + 1$ tentative solutions and repeatedly picking a new test solution –performing operators of reflection, expansion, contraction, and shrinking– to replace one of them (that with the worst value of the objective function), until all vertices in the polytope have converged to very similar values of the objective function or a maximum number of iterations is reached. The iterated version considered here runs the Nelder-Mead method from a random polytope until convergence, and then repeats the procedure until the maximum number of function calls has been reached.

All algorithms considered in the experimentation are available in our code repository [2]. The next section describes the results obtained from their deployment on the generated problem instances.

4 EXPERIMENTAL RESULTS

We have selected a dataset composed of 8 problem instances generated according to the procedure described in Section 3.1. The CA parameters considered in the experimentation are $n = 50$ pedestrians, cell side length $c = 0.5$m, attraction bias $\phi = 4$, repulsion bias $\psi = 0$, speed $v = 0.77$m/s, and maximum evacuation time $T = 60$s. Notice that the values of ψ, ϕ correspond to a scenario of rational agents that proceed to an orderly evacuation of the environment. Regarding the ea, it uses population size $\mu = 128$, recombination probability $p_X = 0.9$, and mutation probability equivalent to $p_M = 1/k$ (where k is the number of exits) with Gaussian amplitude $\sigma = 0.05$. As to nelder-mead, it uses the parameters suggested by [7], namely reflection $\alpha = 1.5$, expansion $\gamma = 2.75$, contraction $\beta = 0.75$, shrink $\delta = 0.5$. The tolerance for convergence is 1% and the maximum number of evaluations per cycle is 1000. In all cases, the problem instances are solved for $k = 4$ emergency exits, and 25 training cases are used within each call to the objective function, for a maximum number of 20000 function calls.

The results on the training set are shown in Table 1. As can be seen, ea generally provides the best numerical results, clearly outperforming the remaining algorithms in most problem instances. This general superiority can also be seen in Figure 3, which depicts the evolution of the best fitness as a function of the number of calls to the objective function. Notice how the greedy algorithm can start producing better solutions, but it is outperformed in the long run. Of course, it is interesting to see how these training results can be

[1] https://osf.io/cnh7u/

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

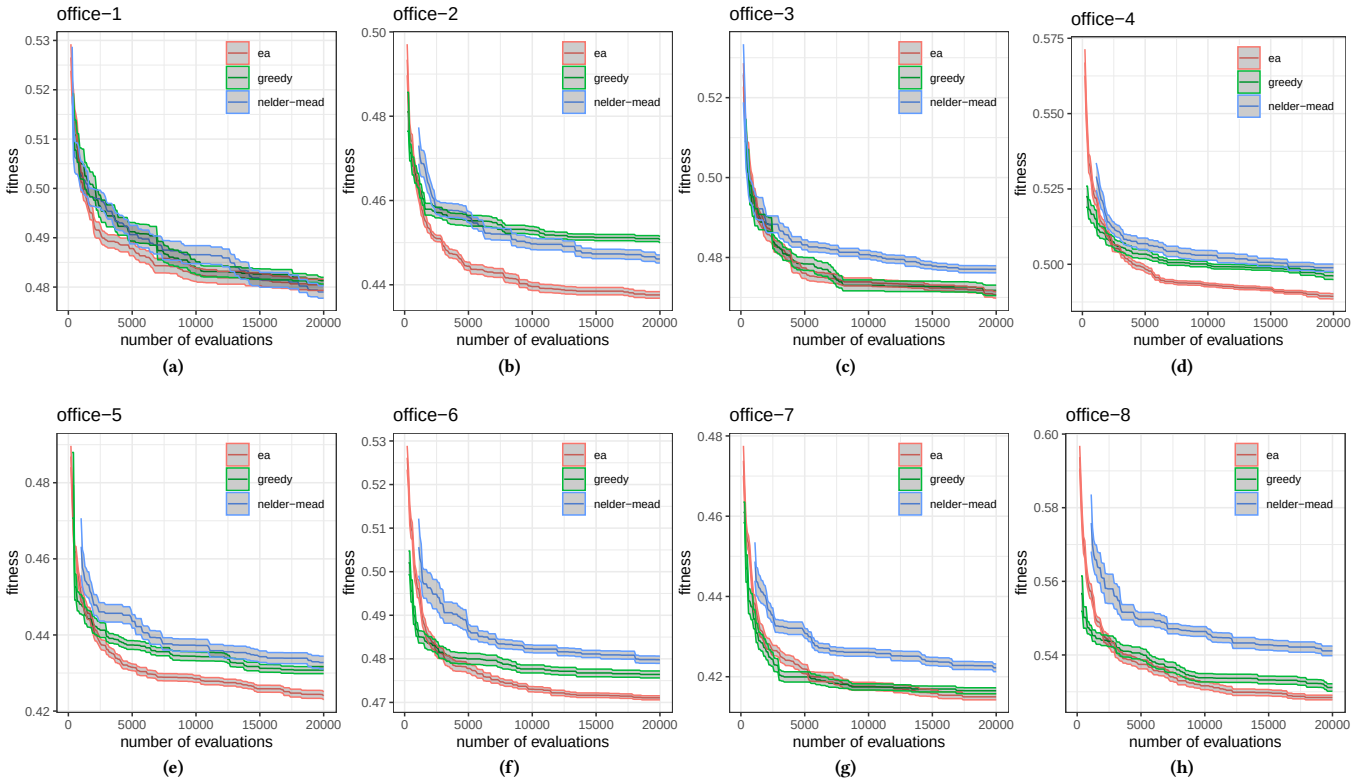
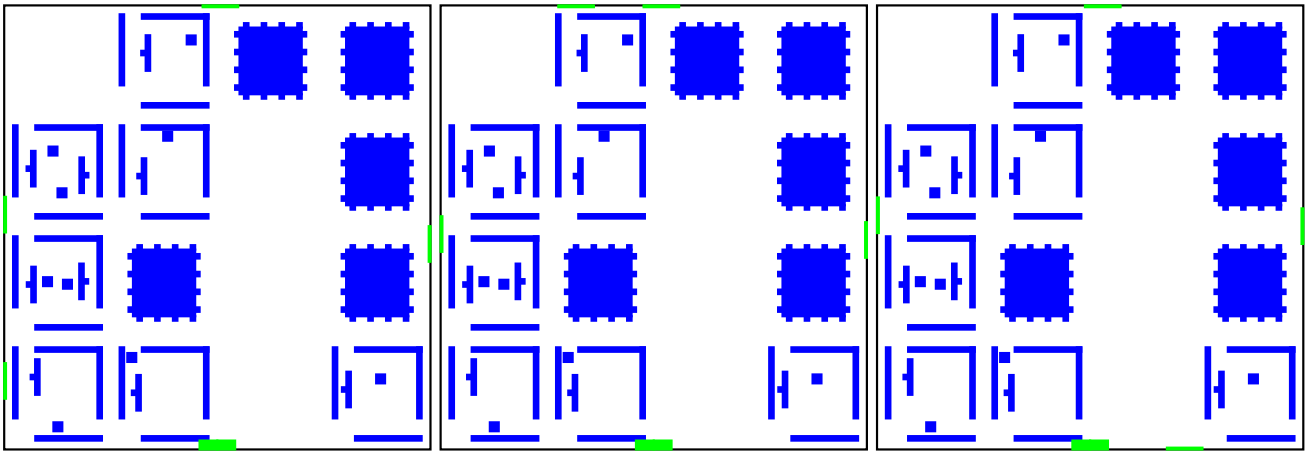


Figure 3: Evolution of the best fitness in each of the problem instances considered. The curves represent the mean best fitness (averaged for 20 runs), and the shaded area around each curve depicts the standard error of the mean.

Table 2: Results of the best solution of each algorithm (according to fitness) on the test set. The legend is the same as in Table 1.

	ea				greedy				nelder-mead			
instance	x^*	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$		x^*	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$		x^*	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	
office-1	0.361	0.481	0.492 ± 0.002	★	0.372	0.491	0.498 ± 0.002	●	0.361	0.492	0.502 ± 0.002	■
office-2	0.328	0.459	0.465 ± 0.002	■	0.339	0.458	0.461 ± 0.002	■	0.339	0.437	0.450 ± 0.002	★
office-3	0.360	0.491	0.503 ± 0.002	■	0.350	0.481	0.492 ± 0.002		0.361	0.481	0.491 ± 0.002	★
office-4	0.383	0.503	0.516 ± 0.002	★	0.405	0.514	0.529 ± 0.002	■	0.405	0.513	0.520 ± 0.002	○
office-5	0.307	0.447	0.453 ± 0.002	★	0.350	0.458	0.462 ± 0.002	■	0.339	0.447	0.454 ± 0.002	
office-6	0.383	0.492	0.500 ± 0.002	★	0.350	0.502	0.508 ± 0.002	●	0.372	0.502	0.513 ± 0.002	■
office-7	0.317	0.437	0.448 ± 0.002	■	0.328	0.437	0.443 ± 0.002	●	0.339	0.436	0.438 ± 0.002	★
office-8	0.405	0.546	0.555 ± 0.002	★	0.394	0.546	0.556 ± 0.002		0.404	0.546	0.560 ± 0.002	

generalized. The results on a test set of 975 cases are shown in Table 2, picking the best solution found during training by each algorithm on each problem instance. Although ea continues to provide very good results, the differences are less marked here and, in particular, nelder-mead shows some very competitive results (unlike greedy, which seems to generally have overfitted the training data). It is interesting, at any rate, to observe the different solutions provided by each algorithm. Figure 4 shows the best solutions found by each algorithm on one of the problem instances. All problem instances featured a regular exit in the middle of the bottom wall (shown with a thicker green rectangle) and determined the location of the remaining $k = 4$ exits. Quite interestingly, all algorithms seem to have allocated 3 of the exits to roughly symmetrical locations covering the four walls of the enclosure and differ on the location of a fourth one. Although the training set appeared to favor placing it on the left wall, as done by ea(Figure 4a), the location at the bottom that reinforces the escape route along the large open area looks much more efficient from a general point of view (Figure 4c). A larger training set could have been required to capture this during the training stage.



(a) (b) (c)

Figure 4: Best solution found during training on the the office-2 instance by (a) ea, (b) greedy, and (c) nelder-mead.

5 CONCLUSIONS

Workplace evacuation planning is an important endeavor to which metaheuristic approaches can be of great help. This has been illustrated in this work considering the specific case of evacuating office environments. The definition of a problem instance generator based on a generative grammar adds a lot of flexibility to the experimentation since it allows constructing diverse datasets of the size and features required. We are currently working on analogous problem generators for other real-world scenarios.

Regarding the numerical results, it is clear that the algorithms considered have shown that effective optimization can take place in this domain. Experimentation also suggests that it would be convenient to consider even larger training sets. Needless to say, this has associated computational implications in terms of time complexity, and hints the need for solutions of computational nature (such as, for example, GPU computing [1]) or algorithmic nature (such as, for example, surrogate models [9]). Along this algorithmic direction, memetic approaches [14] that integrate the methods considered [4] may be of great interest, since each of them has been shown to provide a different, complementary approach to the problem.

ACKNOWLEDGMENTS

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.

REFERENCES

[1] Daniel Cagigas-Muñiz, Fernando Diaz del Rio, Jose Luis Sevillano-Ramos, and Jose-Luis Guisado-Lizar. 2022. Efficient simulation execution of cellular automata on GPU. *Simulation Modelling Practice and Theory* 118 (2022), 102519.

[2] Jieyu Chen, Tianxing Shi, and Nan Li. 2021. Pedestrian evacuation simulation in indoor emergency situations: Approaches, models and tools. *Safety Science* 142 (2021), 105378.

[3] Carlos Cotta and José E. Gallardo. 2024. Evolutionary Algorithms for Optimizing Emergency Exit Placement in Indoor Environments. In *Applications of Evolutionary Computation – 27th International Conference, EvoApplications 2024 (Lecture Notes in Computer Science, Vol. 14634)*, Stephen Smith, João Correia, and Christian Cintrano (Eds.). Springer, Berlin, 194–208.

[4] Sanjoy Das. 2009. Nelder-Mead Evolutionary Hybrid Algorithms:. In *Encyclopedia of Artificial Intelligence*, Juan Ramón Rabuñal Dopico, Julian Dorado, and Alejandro Pazos (Eds.). IGI Global, 1191–1196.

[5] Edsger W Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* 1, 1 (1959), 269–271.

[6] Hairong Dong, Min Zhou, Qianling Wang, Xiaoxia Yang, and Fei-Yue Wang. 2020. State-of-the-Art Pedestrian and Evacuation Dynamics. *IEEE Transactions on Intelligent Transportation Systems* 21, 5 (May 2020), 1849–1866.

[7] Shu-Kai S. Fan and Erwie Zahara. 2007. A hybrid simplex search and particle swarm optimization for unconstrained optimization. *European Journal of Operational Research* 181, 2 (2007), 527–548.

[8] Milad Haghani. 2020. Optimising crowd evacuations: Mathematical, architectural and behavioural approaches. *Safety Science* 128 (Aug. 2020), 104745.

[9] Chunlin He, Yong Zhang, Dunwei Gong, and Xinfang Ji. 2023. A review of surrogate-assisted evolutionary algorithms for expensive optimization problems. *Expert Systems with Applications* 217 (May 2023), 119495.

[10] Feizhou Huo, Yufei Li, Chao Li, and Yaping Ma. 2022. An extended model describing pedestrian evacuation considering pedestrian crowding and stampede behavior. *Physica A: Statistical Mechanics and its Applications* 604 (Oct. 2022), 127907.

[11] Donald E. Knuth. 1968. Semantics of context-free languages. *Mathematical Systems Theory* 2, 2 (June 1968), 127–145.

[12] Donald E. Knuth. 1990. The genesis of attribute grammars. In *Attribute Grammars and their Applications (Lecture Notes in Computer Science, Vol. 461)*, G. Goos et al. (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–12.

[13] Pierre-Yves Louis and Francesca R. Nardi (Eds.). 2018. *Probabilistic Cellular Automata: Theory, Applications and Future Perspectives*. Emergence, Complexity and Computation, Vol. 27. Springer International Publishing, Cham.

[14] Pablo Moscato and Carlos Cotta. 2019. An Accelerated Introduction to Memetic Algorithms. In *Handbook of Metaheuristics*, Michel Gendreau and Jean-Yves Potvin (Eds.). International Series in Operations Research & Management Science, Vol. 272. Springer International Publishing, Cham, 275–309.

[15] J. A. Nelder and R. Mead. 1965. A Simplex Method for Function Minimization. *Comput. J.* 7, 4 (Jan. 1965), 308–313.

[16] OECD.Stat. 2023. Average annual hours actually worked per worker. https://stats.oecd.org/Index.aspx?DataSetCode=ANHRS# Accessed on February 28th, 2024.

[17] International Labour Organization. 2022. Working Time and Work-Life Balance Around the World. Geneve, Switzerland.

[18] Birgitt Schönfisch and André De Roos. 1999. Synchronous and asynchronous updating in cellular automata. *Biosystems* 51, 3 (Sept. 1999), 123–143.

[19] Frank Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin* 1, 6 (Dec. 1945), 80.