Navigating Path-Influenced Environments using Evolutionary Multi-Objective Optimization

Carlo Nübel
carlo.nuebel@ovgu.de
Otto-von-Guericke-University
Magdeburg, Germany

Malte Speidel
malte.speidel@ovgu.de
Otto-von-Guericke-University
Magdeburg, Germany

Sanaz Mostaghim
sanaz.mostaghim@ovgu.de
Otto-von-Guericke-University
Magdeburg, Germany
Fraunhofer IVI
Dresden, Germany

Abstract

This paper explores multi-objective pathfinding in path-influenced environments. These environments contain movable obstacles which can be shifted by the agents. This way, the agents actively change their environment while traversing on their path. Therefore, pathfinding takes on a new dimension. While it has been extensively studied across various domains, finding an optimal path in a path-influenced environment introduces new challenges. In this paper, we propose several real-world inspired problem instances. Then we formally describe this sort of problem as a multi-objective optimization problem and finally evaluate the performance of seven state-of-the-art multi-objective evolutionary algorithms on our problem instances. The results indicate that the evolutionary approach can generate sets of non-dominated solutions for this new problem. The performance of the algorithms in terms of convergence and diversity of the Pareto front highly depends on the way the encountered obstacles are handled, as well as the obstacle distribution on the map. Among the algorithms, AGE-MOEA and SPEA-II demonstrate the best convergence across the majority of problem instances.

CCS Concepts

• **Theory of computation → Shortest paths**; • **Applied computing → Multi-criterion optimization and decision-making**; • **Computing methodologies → Optimization algorithms**.

Keywords

Multi-Objective Optimization, Evolutionary Algorithms, Pathfinding, Path-Influenced Environments

ACM Reference Format:
Carlo Nübel, Malte Speidel, and Sanaz Mostaghim. 2025. Navigating Path-Influenced Environments using Evolutionary Multi-Objective Optimization. In *Genetic and Evolutionary Computation Conference (GECCO '25), July 14–18, 2025, Malaga, Spain.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3712256.3726349

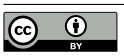
1 Introduction

Finding optimal solutions to pathfinding problems is essential in various domains, including autonomous driving [6], logistics [30], agriculture [5] and medicine [25]. Therefore, pathfinding problems have been extensively studied, with new problems continually being defined. These are often inspired by real-world dynamics, each adding new aspects to the underlying problem: finding the optimal path. One concept of pathfinding problems, as introduced by Heise et al. [19], studies pathfinding in path-influenced environments (PIEs). In such cases, every time an agent takes a step in the environment, and there is an obstacle blocking its path, it changes this environment by moving the obstacle out of its path. This way, the environment dynamically changes while the agent is moving. This concept finds application in various domains where obstacles can be moved for finding a shorter path. Real-world applications of this problem class include warehouse logistics, terrain leveling, snow paving or robotics. In all these examples, the agents' paths actively alter the environment: Whether it is robots rearranging shelves or boxes in warehouses, excavators moving soil, or snowplows clearing roads of snow.

Therefore, path-influenced environments represent a new problem class in which the pathfinding agents in an environment proactively alter this environment to reach the desired destination. Our paper builds upon this concept and has three main goals:

(1) Formally describe this problem as a new multi-objective problem.
(2) Develop new, real-world inspired problem instances, based on the concept of PIEs.
(3) Evaluate the evolutionary approach on the new problem instances.

As this is the first step of defining and analyzing problems based on PIEs, the primary question we aim to address is whether the evolutionary approach can effectively solve this problem, and how the performances of different algorithms compare. In this paper, we present three new environments with different obstacle distributions. We then propose four methods, that the agents can utilize to move the obstacles out of their path. We call these methods shifting strategies. Moreover, we define objectives and modify the operators of existing multi-objective evolutionary algorithms (MOEAs) to be applicable to the problem instances. We model our proposed problem instances as deterministic, single-agent, grid-based pathfinding between two waypoints. Each environment has a different distribution of obstacles, which the agents cannot pass without moving them out of their path. Therefore, the environment is changing dynamically by the path taken, according to the different shifting strategies employed by the agents. The shifting strategy is used to determine in which direction to shift the obstacles along the way. Each combination of environment and shifting strategy effectively defines a new problem instance. We then apply different MOEAs to



This work is licensed under a Creative Commons Attribution 4.0 International License.
GECCO '25, July 14–18, 2025, Malaga, Spain
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1465-8/2025/07
https://doi.org/10.1145/3712256.3726349

these problem instances to determine whether the state-of-the-art algorithms in the multi-objective pathfinding domain can handle this new kind of problem. Finally, we evaluate the results to see which implications these pathfinding problems have and how they can be solved by the different algorithms. The remainder of this paper is structured as follows: Section 2 provides an overview of relevant works in the domain of single and multi-objective pathfinding and offers a classification of pathfinding problems. Section 3 introduces our variants of path-influenced environments, describing the obstacle distribution on each of them as well as the obstacle shifting strategies that can be utilized by the agents. In Section 4 we describe how we apply the evolutionary approach to our new problem class, including the algorithms and operators used. Following this, we present the experimental settings and evaluation metrics, leading to the results showcased in Section 5. Finally, the paper concludes with a summary, and we outline possibilities for future works on this topic in Section 6.

2 Related Work

In the past, several methods have been developed to tackle pathfinding problems. The field of swarm intelligence, for example, offers the ant colony optimization algorithm [15], where the path is found by updating a pheromone matrix, inspired by real-world ants. Further classical approaches to solving single-objective pathfinding problems include Djikstra's algorithm [13], A* [18], and the Bellman Ford algorithm [2] to name a few. For multi-objective pathfinding problems, there exist numerous methodologies to address them as well. Some of them convert the problem back to a single-objective problem, such as the weighted sum approach [23], whereas MOEAs evolve solutions to optimize the objectives simultaneously, resulting in a set of solutions each offering a different tradeoff between objectives [8]. The following selected research items are examples, in which MOEAs are used to optimize different multi-objective pathfinding problems. Firstly, a scalable benchmark suite for multi-objective multi-agent pathfinding was introduced in [32]. In [14], the authors propose an improved MOEA for robots detecting specific points in modelled post disaster coal mining environments. In [22] a hybrid approach between evolutionary algorithms and reinforcement learning was used to optimize a multi-objective orienteering problem. Furthermore, in [16], the authors use a MOEA to optimize paths for inspecting real-world 3D structures. Lastly, in [26] the authors compare differential evolution with NSGA-II for trajectory planning of mobile robots. These examples illustrate a selection of the many applications of the evolutionary approach to multi-objective pathfinding problems.

In general, pathfinding problems can be classified as follows. The first important distinction is between grid-based and graph-based pathfinding. In grid-based environments, space is divided into discrete cells, making it easier to define movement constraints and objectives. In contrast, graph-based pathfinding represents the environment as a network of nodes and edges, which can offer greater flexibility in certain scenarios [17, 29]. Pathfinding can also vary, based on the nature of the environment, specifically whether it is static or dynamic. Static environments feature fixed obstacles, while dynamic environments include obstacles that can move or change over time, adding complexity to the pathfinding process [1]. Closely related to this is the distinction between deterministic and stochastic environments. In deterministic settings, obstacles remain fixed and predictable, whereas in stochastic environments, obstacles may randomly appear or disappear, such as in scenarios where doors open and close unexpectedly [27]. In multi-agent pathfinding, multiple agents must navigate a shared environment without colliding with the other agents. This type of problem can be important for applications in robotic swarm coordination and traffic management [28]. Finally, the objectives for which the paths are optimized also contribute to the diversity of pathfinding problems. Common objectives include minimizing cost, time, or distance. Additionally, more complex goals, such as optimizing for path smoothness and reducing risk, are often considered, depending on the specific requirements of the application [33]. Table 1 summarizes the given classification of pathfinding problems.

Table 1: Classification of pathfinding problems.

Category	Description
Representation	• Grid-based: discrete cells • Graph-based: nodes and edges
Type	• Static: fixed obstacles • Dynamic: moving/changing obstacles
Behaviour	• Deterministic: environment is predictable • Stochastic: environment is subject to random changes
Agent Count	• Single-Agent: One agent in environment • Multi-Agent: multiple agents in shared environment
Objectives	• Single-Objective: one objective • Multi-Objective: 2–3 objectives • Many-Objective: 3 and more objectives

These classifications illustrate the varied and evolving nature of pathfinding challenges, each requiring different strategies and solutions. None of the current pathfinding problems consider how the environment can be actively influenced by the path that is taken by an agent. Instead, most of them assume that the environment is not changed by the path. This is undoubtedly the right approach for many applications, such as finding an optimal path between two geographical points while minimizing the two objectives of travel time and distance, but for others, it is not. A robot moving around boxes in a warehouse, a snow plow clearing a parking lot, a catheter having to push organs to the side to reach a specific point in the body, all those examples have one thing in common: the agents change the environment while traveling along their path. These real-world examples are the inspiration for the problems studied in this paper.

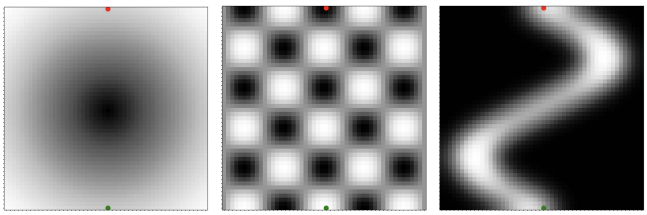
3 Pathfinding in Path-Influenced Environments

In this section, we propose a new multi-objective pathfinding problem in path-influenced environments, which comes with the new challenge that the agents can actively change the environment while moving along the path. The agents do not avoid or collect the

obstacles in the environment, but instead they move them around. For example, robots in a warehouse can reposition boxes or shelves, or snowplows clearing out snow, and this way they change the environment. We provide a generalized formulation of this problem where the agents' goal is to find a path between two waypoints. On their paths, they encounter obstacles with different weight values. To move to the next cell, the agent must first move the obstacle from that next cell to one of the neighboring cells. The obstacle weight is added to the obstacle weight of the neighboring cell. Therefore, in each step of the path of each agent, the environment is changed. The agents' objectives are to minimize the path length and total amount of obstacle weight shifted (details are provided in Section 4).

3.1 Obstacle Environments

We propose three different environments, which are modeled as two-dimensional grid worlds with various distributions of obstacles (Figure 1). We consider quadratic maps as the environment with the size of width × height ($W \times H$). On each map, the starting point is in the bottom middle cell and the agents' goal is the top middle cell. Each cell on all the maps has an obstacle weight between 0 and 1, illustrated by the different colors. A cell with an obstacle weight of 0 is displayed in white, and a cell with a weight of 1 is black. As the first objective is the length of the path and the shortest path on each map is straight up, we expect these paths to be found by the tested algorithms on all of these maps. The second objective, total obstacle weight shifted, however, will cause the agents to avoid areas with many obstacles, which will result in longer paths. This is due to the obstacle distribution on the map, which causes the objectives to be conflicting with each other. The following equations define



(a) Radial gradient environment. **(b) Sinusoidal environment.** **(c) River environment.**

Figure 1: Environments with different obstacle distributions, start point at the bottom middle cell (green), and end point at the top middle cell (red).

the obstacle distribution for each map. The formulas can be used to create obstacle distributions for both square and rectangular maps. Firstly, we define the radial gradient map in Figure 1a. The gradient is based on Equation 1.

$$r(x,y) = 1 - \frac{\sqrt{\left(x - \frac{W}{2}\right)^2 + \left(y - \frac{H}{2}\right)^2}}{D_{\max}} \tag{1}$$

D_{max} is the Euclidean distance from the center of the map to any of the corner points, calculated as defined in Equation 2.

$$D_{max} = \sqrt{\left(\frac{W}{2}\right)^2 + \left(\frac{H}{2}\right)^2} \tag{2}$$

Here we expect to see paths that avoid the obstacle weight in the middle and instead find paths going around it. In Figure 1b we define the sinusoidal map, a grid world with multiple obstacle weight peaks and valleys, following an approximated sinusoidal pattern as defined in Equation 3.

$$s(x,y) = \sin\left(\frac{5\pi x}{W}\right) \cdot \cos\left(\frac{5\pi y}{H}\right) \tag{3}$$

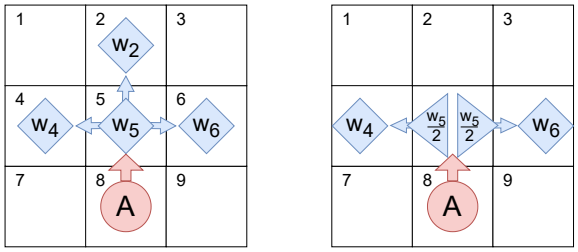
On this map, we expect the final paths to avoid the peaks with high obstacle weights and utilize the valleys. The final map, in Figure 1c, is the river map, representing the shape of a meandering river. The river follows a sinusoidal curve defined in Equation 4.

$$u(y) = \frac{W}{2} + \frac{W}{3} * \sin\left(2\pi \frac{y}{W}\right) \tag{4}$$

The river has a width of seven cells, and we applied Gaussian filtering to its edges to make the transitions between high and low obstacle weights smooth. We expect the different agents to again find the shortest path as well as paths approximating the sinusoidal shape of the meandering river, as the obstacle weight is lowest there.

3.2 Obstacle Shifting Strategies

We introduce four different obstacle shifting strategies. These define in which direction the agents shift each encountered obstacles on the path, as shown in Figure 2. Given the above environments, the agents paths are connected using the Von Neumann neighborhood, meaning the agent can only choose between the four direct neighbors of the current cell to move to. In the scenarios shown in Figure 2, the agent A moves from the bottom middle cell, with index 8, to the center cell, 5. To do so, the agent must move the obstacle weight w_5 from the center cell 5 to one of the neighboring cells, 2, 4, or 6 marked with the blue arrows. He cannot move the weight to the cell from which he came from, cell 8 in this example.



(a) Neighbor-dependent shifts. **(b) Split-in-two (S2) shift.**

Figure 2: Shifting strategies of agent A moving from cell 8 to cell 5 and therefore shifting the weight w_5.

The first three shifting strategies are neighbor-dependent shifts, meaning the shifting direction is dependent on the obstacle weight distribution of the neighboring cells. We use Figure 2a to explain these strategies, starting with the least resistance (LR) shift. This method compares the weights of the adjacent cells 2, 4, and 6. w_5 is then moved to the neighboring cell with the smallest obstacle. Consider, for example, that a robot in a warehouse is stacking crates

on top of each other. For safety reasons, this robot might want to keep the towers as small as possible, and would stack the next crate always on the smallest stack of crates. The second shifting strategy is the maximum resistance (MR) shift, and follows a similar approach. Again, the weights of the neighboring cells are compared. In contrast to the LR shift, the weight w_5 is moved to the cell with the highest obstacle weight. Here, the previously mentioned warehouse robot could have the objective to keep as much of the floor clear of crates as possible. Thirdly, we introduce the equalize neighbor weights (EN) shift. For this shifting strategy, the weight w_5 is split into multiple parts and is distributed to the adjacent cells in a way that all neighboring cells have the same obstacle weight on them. In case w_5 is not large enough to equalize all the neighbors' weights, w_5 is distributed to get as close as possible to an equal distribution. In this case, the distribution is scaled proportionally, based on a normalization factor. The robot could for example be working on a ship, where it is important to distribute the cargo weight equally across the ship's storage space. The fourth obstacle-shifting strategy is not dependent on the neighboring cells' weights. Figure 2b illustrates the split-in-two (S2) shift. Here, w_5 is divided by two and the two parts are shifted to the two neighboring cells that are not in the current movement direction of the agent. If one of the neighboring cells is a wall, all the weight is shifted to the remaining cell. Table 2 offers an overview of the different shifting strategies introduced in this paper and explains the different methods. Our individual problem instances are the combinations of the

Table 2: Summary of the shifting strategies.

Strategy	Explanation
Least Resistance (LR)	• Compare neighboring cells' obstacle weights • Shift to the cell with the lowest weight
Maximum Resistance (MR)	• Compare neighboring cells' obstacle weights • Shift to the attached cell with the highest weight
Split in Two (S2)	• Split weight in half • Distribute weight to the two adjacent cells, that are not in the agents' current movement direction
Equalize Neighbors (EN)	• Compare neighboring cells' obstacle weights • Split and distribute the weight to balance adjacent cell weights as evenly as possible

agents' shifting strategy and the environment. The design of the different obstacle distributions on the maps is aimed at generating environments with varying difficulties, while the real-world applications inspire the shifting strategies. Using the four different shifting strategies on the three obstacle environments results in 12 different problem instances.

4 Multi-Objective Optimization Approach for Pathfinding in PIEs

Our proposed multi-objective optimization approach for pathfinding in PIEs studies two objective functions, path length and total obstacle weight shifted, both of which are to be minimized. We encode each individual using a variable length encoding. An individual X is represented as a list of tuples $X = [c_1, c_2, ..., c_n]$, with n being the total number of steps taken to reach the target cell. The length of the list, n, can be different for each individual. Each tuple c corresponds to the x and y coordinates of a cell in the grid world, $c_i = (x_i, y_i) \quad \text{where} \quad x_i \in \{1, \ldots, W\},\ y_i \in \{1, \ldots, H\}$. To evaluate an individual in the first objective, f_1, we take the length of that individual's list of visited cells, X. Therefore, f_1 is simply n. This provides us the value for the first objective f_1: path length. The second objective, f_2, sums up all the obstacle weights that are shifted on the path. It is calculated by taking the individual's path, and applying it to the obstacle map. As the agent follows the path, the shifting strategy is executed, updating the map and its obstacle values at each step accordingly. During this process, the weight of the obstacles shifted by an agent along the path is summed up, yielding the value for the second objective as defined in Equation 5, where the obstacle weight of a cell is $w(c_i)$.

$$f_2(X) = \sum_{i=1}^{n} w(c_i) \tag{5}$$

As for the different operators used in the MOEAs, we modify the sampling method, and the mutation, and crossover operators to align with our problem description.

4.1 Sampling of Initial Population

The initial population is generated using random sampling. Each agent starts at the starting position and then moves to a randomly chosen neighboring cell. This is done, until the target cell is reached. Per individual, one random path is sampled that must fulfill three conditions:

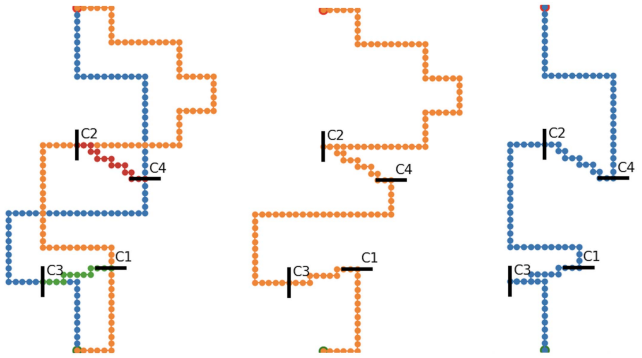
(1) The path must connect the start cell to the target cell.
(2) Every path must be fully connected, as defined by the Von Neumann neighborhood.
(3) Paths cannot exceed the size of the map.

This means each individual can have a different length, resulting in the use of variable length encoding. Therefore, our genetic operators have to be adapted as explained in the following subsections.

4.2 Crossover Operator

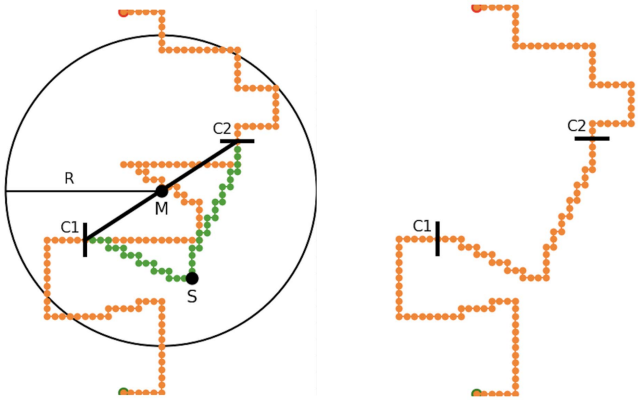
Figure 3 shows an example for the crossover operator. We use a two-point crossover, which generates two children from two selected parents. The first cutoff point for both parents is randomly selected, with the constraint that it must fall outside the last 20% of the parent's path. Figure 3a illustrates an example with two parent individuals (blue and orange paths) and the first cutoff point in each parent ($C1$ and $C3$). The second cutoff point is then positioned 20% further along the path from the first cutoff point ($C2$ and $C4$ in Figure 3a not to scale). This way, it is guaranteed that the second cutoff point is still on the path. The two middle segments between the cutoff points in each parent are swapped by reconnecting each to the other parent using the shortest path, which are displayed as the green and red paths between $C2$ and $C4$ as well as $C1$ and $C3$. This effectively exchanges the middle sections of both parents, creating two new offspring, shown in Figure 3b and Figure 3c. With

this crossover operator, individuals with paths of different lengths can be recombined.



(a) Crossover steps. (b) First child. (c) Second child.

Figure 3: Crossover operator.



(a) Perimeter mutation. (b) Mutated child.

Figure 4: Mutation operator.

4.3 Mutation Operator

We use the Perimeter Mutation Operator proposed by Weise in 2023 [31] as illustrated in Figure 4. First, two cutoff points along the path of the original individual are selected, $C1$ and $C2$. The first point is sampled at random, again with the condition that it cannot be in the last 20% of the path. The second cutoff point is then placed 20% further along the path, similar to the crossover operator. This way, always 20% of an individual's path is exchanged, limiting the impact of the mutation on an individual. This also offers the advantage that individuals with different path lengths are changed to the same extent in relation to their path length. The midpoint, M, between the two cutoff points is then identified. Within a radius R around this midpoint, a random point S, is sampled. This point S is sampled, until it is within the bounds of the height and width of the map. The radius R is set as 40% of the map width. The original cutoff points are then reconnected to this newly sampled point using a greedy shortest-distance approach. This effectively creates a new segment in the path, which is represented as the green path in Figure 4a. The mutated individual is illustrated in Figure 4b. In the crossover operator, the size of the parts of the parents' paths that are exchanged was determined through preliminary experiments. This also holds for the size of the path that is mutated in the mutation operator and the radius in which the random point S is sampled. While it is common to set these sizes relative to the total length of the path in evolutionary algorithms for pathfinding problems, it is important to conduct further analysis on how the chosen values for these parameters affect the performance of the algorithms.

5 Experiments and Results

In the following subsections, we describe the setup for our experiments. We cover the algorithms tested, including their parameters, illustrate the evaluation metrics, and then share our results along with an analysis.

5.1 Experiment Settings and Evaluation Metrics

We conduct experiments for seven algorithms using the four different shifting strategies on the three obstacle maps. Each algorithm runs for 100,000 function evaluations to make them comparable. All of these settings are run 31 times using a different seed for each run to minimize the stochastic influence on our results. We use a population size of 100 individuals. The crossover rate is fixed with a value of $P_c = 0.8$ and our mutation probability is set at $P_m = 0.2$, as these values are commonly used for solving pathfinding problems with MOEAs, e.g., [31]. Furthermore, we use grid environments with a size of 51×51 cells. We implemented our problem using the `pymoo` framework [4], leveraging the pre-implemented algorithms to test on our specific problem. For the operators and parameters not discussed in the previous section, we use the default settings given by the `pymoo` framework.

For this paper, we selected seven different MOEAs that are designed to handle diverse optimization problems. We selected them to test known concepts of multi-objective optimization algorithms on our problem. Using the modifications to the genetic operators as described in Section 4, all the following algorithms are applicable to the defined problem instances. The following algorithms are tested: As one of the most known MOEAs, NSGA-II sorts the individuals in the population into fronts and initially selects them based on the rank of the front and then based on the crowding distance metric to maintain diversity in the population [10]. NSGA-III builds upon the concept of sorting the individuals into fronts, but instead of crowding distance, it uses reference directions to maintain diversity [9]. We generated the reference directions for this algorithm using the Das-Dennis method with 10 partitions [7]. SPEA-II iteratively updates an archive to store non-dominated solutions. Fitness is assigned based on a strength value, which is based on the number of solutions an individual dominates [34]. AGE-MOEA follows the basic concept of NSGA-II as well, but instead of using crowding distance, every solution gets a survival score that evaluates both the diversity and convergence of the solutions based on their distance to neighbors and the ideal point [24]. Another modification made to NSGA-II is used by R-NSGA-II [12]. Here, individuals are ranked based on the Euclidean distance to a reference point. Solutions close to these reference points are prioritized in the survival selection. D-NSGA-II is an adaptation of NSGA-II designed to handle dynamic problems [11]. For every iteration, several solutions are reevaluated to see if their fitness has changed. This would indicate

that the problem has changed and then, the entire population is reevaluated. Finally, SMS-EMOA again is based on NSGA-II, but the crowding distance is replaced with the hypervolume indicator as a survival selection mechanism [3]. Algorithms designed for three or more objectives, such as NSGA-III and R-NSGA-II can still be used on two-objective optimization problems. However, we are aware that due to their design, they can be inefficient on two-objective problems.

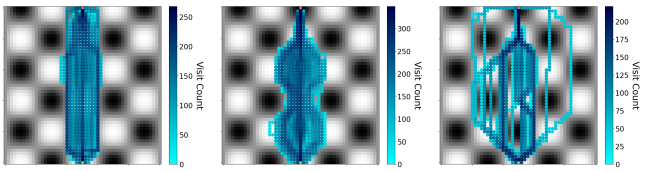
To evaluate our results, first, we visually compare the paths found by NSGA-II on each map using each shifting strategy. Then we use four different evaluation metrics, namely hypervolume (HV), C-metric [35], generational distance (GD^+) and inverted generational distance (IGD^+) [21]. For the hypervolume we calculated one reference point P_{ref} for each map by taking the Nadir point of all solutions S generated by all algorithms over all seeds on that map in the first generation. We then added 10% to each objective value, making the reference points slightly worse than the Nadir points, as defined in Equation 6.

$$P_{ref} = (1.1 \times \max_{s \in S} f_1(s),\ 1.1 \times \max_{s \in S} f_2(s)) \tag{6}$$

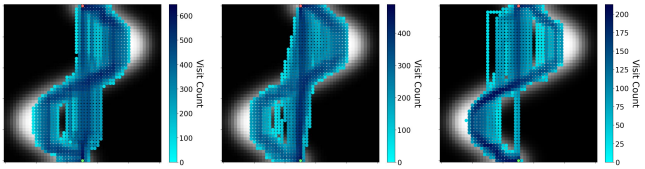
Although using hypervolume based on a reference point that is slightly worse than the Nadir point is not always appropriate for evaluating MOEAs [20], incorporating the additional evaluation metrics helps to ensure the reliability of our findings. We approximated one true Pareto front per combination of map and shifting method needed for the GD^+ and IGD^+ calculation by taking all optimal solutions from the last ten generations of each algorithm over all the different seeds. From this set of solutions, we then again extract the Pareto front and use this as our approximated true Pareto front. In the remainder of this section, the algorithms' performance will be analyzed for diversity, convergence, and convergence speed using the evaluation metrics discussed.

5.2 Results

Figure 5 depicts how different maps and different obstacle handling-strategies can influence the resulting Pareto optimal paths. These paths are found by NSGA-II, used here as an example. Cells with a dark blue color have been visited more often in the final paths, whereas light blue cells have only been visited sparsely. The different visit counts per cell in each sub-figure can be explained by the different amounts of non-dominated paths found by NSGA-II during the experiments. Higher visit counts on a map simply stem from more Pareto optimal paths found during that optimization process. The first three figures in the top row depict three different shifting strategies MR, S2, and EN shift deployed on the sinusoidal map (Figure 5a to Figure 5c). In Figure 5d to Figure 5f, the optimal paths found by NSGA-II using the same three shifting strategies on the river map are displayed. The agents executing the MR shifting strategy on the sinusoidal map (Figure 5a) are pushing the encountered obstacle weights to the side with the peak obstacle weights (black areas on the map). When we consider the outer paths and not the paths going straight up through the middle, these peaks are found to the left or the right of the agent's upward movement direction, but not in the front of the agent. Therefore, the agent is not blocking its path going upward, and we can see that the majority of paths are going straight upwards with an offset to the right and to the left, to avoid the main obstacle peaks of the map. The paths found by NSGA-II on the sinusoidal map using the S2 shift, shown in Figure 5b, explore the valleys (white areas on the map) more thoroughly compared to the MR shift. The dynamic of the S2 shift, splitting the encountered weight in two and pushing it to the left and right neighboring cells in relation to the agents current movement direction, punishes the change of direction because there will always be weight on that next cell due to the previous movement and shift. But, as only half the weight is shifted there, the valleys are still being explored more thoroughly compared to the MR shift on the same environment. Paths found while using the EN shift are exploring more areas of the map compared to the other shifting strategies, as can be seen in Figure 5c. This is due to the dynamics of the EN shift. Part of the obstacle weight is often being shifted into the agents' movement direction. That leads to the agent switching directions more often and not prioritizing moves in movement direction because in each direction the weight was equalized, and it can not always effectively avoid obstacles by continuing in one direction. Therefore, more areas of the map are explored. compared to the other two strategies. In Figure 5d to Figure 5f, where the shifting strategies are applied to the river map, the MR and the S2 shift cover the map to a similar extent, identifying paths both directly through the center and along the obstacle-free meandering river section.



(a) MR shift on sinusoidal map. **(b) S2 shift on sinusoidal map.** **(c) EN shift on sinusoidal map.**



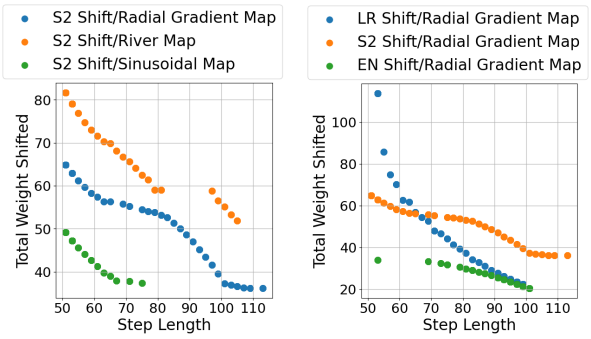
(d) MR shift on river map. **(e) S2 shift on river map.** **(f) EN shift on river map.**

Figure 5: Optimal paths found by NSGA-II on the sinusoidal map and river map using different shifting strategies.

To summarize the visual comparison of paths generated by NSGA-II across our environments with different shifting strategies, it is evident that the evolutionary approach successfully identifies the majority of the expected paths on each map. This observation extends to all seven algorithms across the three maps. Additionally, the shifting strategies impact the diversity of solutions, particularly in terms of the extent to which the paths explore different areas of the map. The reason for this lies in the different ways the obstacles are shifted in each shifting strategy. To make these observations more apparent, Figure 6 depicts the influence of different maps on the same shifting strategy and the influence of three different shifting strategies used on the same obstacle environment. The

solutions in these figures show the objective values of the Pareto optimal paths found across all algorithms. The first evident similarity across the maps is that the total weight shifted decreases with longer paths. This is expected, as longer paths allow agents to avoid more obstacles, reducing the need for shifting obstacles around. The different maps feature varying amounts of obstacles along the shortest path straight through the middle, which leads to different total weights that need to be shifted. This explains the variation in the Pareto fronts regarding the second objective, total weight shifted, shown in Figure 6a. In this figure, we can also observe the following differences. While the S2 shift on the sinusoidal map (green) results in a convex Pareto front, on the radial gradient map (blue), the front has a concave part in the middle. On the River map (orange), however, the Pareto front contains a gap. The concavity in the blue front likely arises from the S2 shift dynamics: When the agent splits weight between neighboring cells and changes direction, it must shift both the new weight and previously shifted weight, potentially causing multiple shifts in obstacle weight before reaching lower-weight areas. The gap in the orange Pareto front for the river map likely stems from the map's design. Two main strategies emerge: a direct, upward path minimizing distance but incurring high obstacle weights and a longer route avoiding obstacles. Intermediate paths are unviable due to the trade-off between path length and total shifted weight, leading to the observed gap. This illustrates how the environment structure and shifting strategy shape the trade-offs between objectives, as reflected in the Pareto fronts. In Figure 6b we illustrate the influence of different shifting strategies on the same map in contrast to the same shifting strategy on different maps as discussed previously. The radial gradient map is used as an example here. It has the highest obstacle weight concentrated in the center of the map, with weights decreasing further away from the middle. When an agent utilizing the LR shift (blue Pareto front) follows the shortest path straight up through the middle, it initially pushes the obstacle weight to the left or right as it moves upward. However, upon reaching the midpoint of the map, the agent begins pushing the weight directly upward in its movement direction. This causes the weight to stack in front of the agent, leading to a cumulative increase in the total obstacle weight shifted with each subsequent step. This behavior explains the high values observed for the objective total weight shifted when using the LR shift strategy on this map. The S2 shift on the radial gradient map behaves as explained previously, while the Pareto front converges best with the EN shift on the radial gradient map. With these examples, we can conclude, that the different maps and different shifting strategies have a large influence on the shape and diversity of the Pareto front.

To compare the performance of the different algorithms on our problem instances in terms of convergence and convergence speed, we look at the HV as well as GD^+ and IGD^+. Independent of the map, in terms of HV, GD^+, and IGD^+, all algorithms converge after approximately 30 generations, as shown in Figure 7a, Figure 7b and Figure 7c. These figures show the results of the three evaluation metrics on the linear gradient map, river map, and sinusoidal map using the S2 shift as an example. The same behavior can be seen for the S3 shift, MR shift, and LR shift. The exception here is the EN shift for which the metrics have not yet converged after 50 generations as shown in Figure 7d, Figure 7e, and Figure 7f. Instead,



(a) Pareto optimal solutions of S2 shift on the radial gradient map, the river map and sinusoidal map.

(b) Pareto optimal solutions of all algorithms using LR shift, EN shift and S2 shift on the radial gradient map.

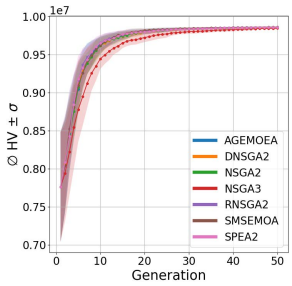
Figure 6: Pareto optimal paths' dependency on the different maps (a) and shifting strategies (b).

it will converge around generation 150. One reason for this is that by equalizing the obstacle weights of the neighboring cells, the maps are more thoroughly explored, as cells with initially low obstacle weights are not identified as such due to the prior equalization. Therefore, paths with low obstacle values are discovered late. While this process takes more time, it results in more diverse solutions, as already shown in Figure 5. In all diagrams in Figure 7 we can also observe that the red line representing the convergence behavior of the metrics of NSGA-III is converging slower during the first 25 generations compared to the other algorithms, but eventually converges to the same level as the others. This can be due to the aforementioned more complex algorithmic design of NSGA-III, as this algorithm was developed for three or more objectives.

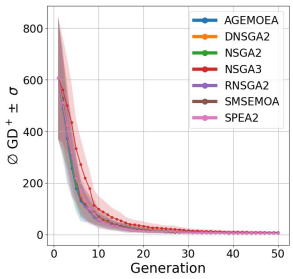
To analyze the different algorithms' convergence behavior more detailed, we use the C-metric. Table 3 shows how many solutions of the Pareto front of the algorithm in the second column dominate the Pareto optimal solutions of all the other algorithms. For example, in the first row, the Pareto optimal solutions of NSGA-III on the radial gradient map using the LR shift are dominating 12.59% of all the other Pareto optimal solutions found by the other six algorithms. We highlight the best and worst converging algorithms in the table for each of the 12 problem instances. The differences in convergence are small, yet across the majority of our problem instances, D-NSGA-II shows the worst convergence, followed by NSGA-III. This result is to be expected because with D-NSGA-II we use an algorithm designed for dynamic problems on a static problem and with NSGA-III we use an algorithm designed for three or more objectives on a two-objective problem. Yet, as our problems are the first using the concept of PIEs, it is worthwhile to look into various existing ways which have proven capable to solving related problems. AGE-MOEA as well as R-NSGA-II on the other hand, show the best convergence.

6 Conclusion and Future Works

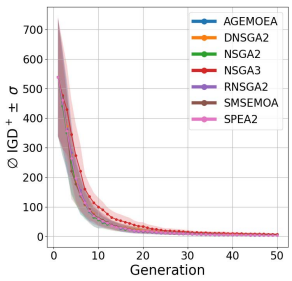
In this paper, we extended the concept of path-influenced environments. These are environments, in which pathfinding agents can interact with obstacles in the environment, pushing them out of



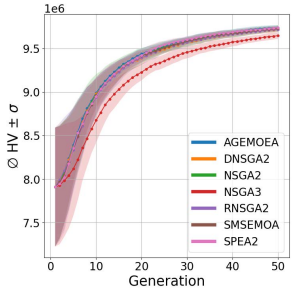
(a) HV on linear gradient map with S2 shift.



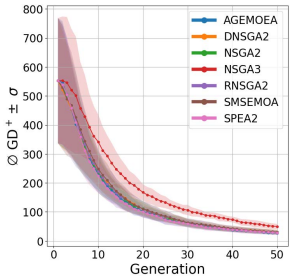
(b) GD^+ on river map with S2 shift.



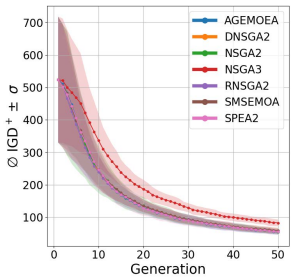
(c) IGD^+ on sinusoidal map with S2 shift.



(d) HV on linear gradient map with EN shift.



(e) GD^+ on river map with EN shift.



(f) IGD^+ on sinusoidal map with EN shift.

Figure 7: Average HV, GD^+ and IGD^+ results with standard deviation for specific combinations of maps and shifting strategies.

their way. Each step taken on the path, therefore, actively changes the environment. Real-world scenarios to which this concept can be applied, include, for example, robotic agents clearing debris after an accident, reorganizing storage space in warehouses or snow plows clearing roads. Based on this concept, this paper has three main contributions. Firstly, we developed new, real-world inspired problem scenarios. This was achieved by introducing three different environments along with four different shifting strategies, with which the agents can interact with the obstacles in the environment. Secondly, we formulated the problem as a multi-objective problem to which the evolutionary optimization approach can be applied. And finally, we evaluated the performance of seven state-of-the-art multi-objective evolutionary algorithms on these instances, each of which has proven useful on different pathfinding problems in the past. Our analysis of the algorithms' performance focused

Table 3: Dominated solutions of the Pareto optimal solutions found by the other algorithms in % to determine convergence performance of algorithms per problem instance (worst and best converging results per scenario highlighted).

Shifting Strategy	Algorithm	Radial Gradient Map	Sinusoidal Gradient Map	River Map
LR Shift	NSGA-III	**12.59**	36.91	23.83
	SPEA-II	**61.84**	45.24	64.85
	R-NSGA-II	49.83	**56.35**	49.99
	AGE-MOEA	59.92	49.21	**71.81**
	DNSGA2	18.57	**15.08**	**10.06**
MR Shift	NSGA-III	**17.64**	29.15	20.68
	SPEA-II	44.35	**71.89**	**81.56**
	AGE-MOEA	**55.97**	40.29	65.65
	DNSGA2	19.33	**14.71**	**6.86**
S2 Shift	SPEA-II	52.30	56.13	**72.68**
	R-NSGA-II	60.43	**70.80**	63.87
	DNSGA2	**15.31**	**3.03**	**7.74**
	SMS-EMOA	**64.32**	48.99	39.29
EN Shift	NSGA-II	**25.12**	58.89	55.58
	AGE-MOEA	23.67	21.39	**78.98**
	DNSGA2	**9.31**	**7.50**	**20.93**
	SMS-EMOA	17.59	**60.83**	53.69

on convergence, diversity, and convergence speed. Therefore, we utilized visual comparisons of the resulting paths along with the performance indicators HV, GD^+, IGD^+, and the C-metric.

Notably, AGE-MOEA and R-NSGA-II performed well on most of the problems, while D-NSGA-II and NSGA-III were less effective because their respective algorithmic design does not fit our problem modeling. However, none of the algorithms tested in this paper were able to handle all problem variations equally well. Instead, the performance of each algorithm was highly dependent on the specific combination of environment and obstacle shifting strategy. Yet, we can conclude, that the evolutionary approach can successfully be applied to the individual problem instances proposed in this paper.

In the future, we will explore the creation of maps with immovable as well as randomly changing obstacles to increase the complexity of the underlying pathfinding problem and introduce upper limits for obstacle weights per cell (e.g., restricting the height of stacked objects). Finally, we want to investigate alternative ways to more efficiently solve this new problem class.

References

[1] Fatin H. Ajeil, Ibraheem Kasim Ibraheem, Mouayad A. Sahib, and Amjad J. Humaidi. 2020. Multi-objective path planning of an autonomous mobile robot using hybrid PSO-MFB optimization algorithm. *Applied Soft Computing* 89 (April 2020), 106076. https://doi.org/10.1016/j.asoc.2020.106076

[2] Richard Bellman. 1958. On a routing problem. *Quart. Appl. Math.* 16, 1 (1958), 87–90. https://doi.org/10.1090/qam/102435

[3] Nicola Beume, Boris Naujoks, and Michael Emmerich. 2007. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research* 181, 3 (Sept. 2007), 1653–1669. https://doi.org/10.1016/j.ejor.2006.08.008

[4] Julian Blank and Kalyanmoy Deb. 2020. Pymoo: Multi-Objective Optimization in Python. *IEEE Access* 8 (2020), 89497–89509. https://doi.org/10.1109/ACCESS.

2020.2990567

[5] Lukas Bostelmann-Arp, Christoph Steup, and Sanaz Mostaghim. 2023. Multi-Objective Seed Curve Optimization for Coverage Path Planning in Precision Farming. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, Lisbon Portugal, 1312–1320. https://doi.org/10.1145/3583131.3590490

[6] Laurene Claussmann, Marc Revilloud, Dominique Gruyer, and Sebastien Glaser. 2020. A Review of Motion Planning for Highway Autonomous Driving. *IEEE Transactions on Intelligent Transportation Systems* 21, 5 (May 2020), 1826–1848. https://doi.org/10.1109/TITS.2019.2913998

[7] Indraneel Das and J. E. Dennis. 1998. Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems. *SIAM Journal on Optimization* 8, 3 (1998), 631–657. https://doi.org/10.1137/S1052623496307510 _eprint: https://doi.org/10.1137/S1052623496307510.

[8] Kalyan Deb. 2001. *Multiobjective Optimization Using Evolutionary Algorithms. Wiley, New York.*

[9] Kalyanmoy Deb and Himanshu Jain. 2014. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation* 18, 4 (Aug. 2014), 577–601. https://doi.org/10.1109/TEVC.2013.2281535

[10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (April 2002), 182–197. https://doi.org/10.1109/4235.996017

[11] Kalyanmoy Deb, Udaya Bhaskara Rao N., and S. Karthik. 2007. Dynamic Multi-objective Optimization and Decision-Making Using Modified NSGA-II: A Case Study on Hydro-thermal Power Scheduling. In *Evolutionary Multi-Criterion Optimization*, Shigeru Obayashi, Kalyanmoy Deb, Carlo Poloni, Tomoyuki Hiroyasu, and Tadahiko Murata (Eds.). Vol. 4403. Springer Berlin Heidelberg, Berlin, Heidelberg, 803–817. https://doi.org/10.1007/978-3-540-70928-2_60 Series Title: Lecture Notes in Computer Science.

[12] Kalyanmoy Deb and J. Sundar. 2006. Reference point based multi-objective optimization using evolutionary algorithms. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM, Seattle Washington USA, 635–642. https://doi.org/10.1145/1143997.1144112

[13] E. W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numer. Math.* 1, 1 (Dec. 1959), 269–271. https://doi.org/10.1007/BF01386390

[14] Jinshuai Dong, Hongguo Wang, and Quan-Ke Pan. 2022. An Improved Multi-objective Evolutionary Algorithm for A Robot Detect Path Planning Problem. In *2022 41st Chinese Control Conference (CCC)*. 2071–2076. https://doi.org/10.23919/CCC55666.2022.9902876

[15] M. Dorigo, V. Maniezzo, and A. Colorni. 1996. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26, 1 (Feb. 1996), 29–41. https://doi.org/10.1109/3477.484436

[16] K.O. Ellefsen, H.A. Lepikson, and J.C. Albiez. 2017. Multiobjective coverage path planning: Enabling automated inspection of complex, real-world structures. *Applied Soft Computing* 61 (2017), 264–282. https://doi.org/10.1016/j.asoc.2017.07.051

[17] Jianqi Gao, Yanjie Li, Xinyi Li, Kejian Yan, Ke Lin, and Xinyu Wu. 2024. A review of graph-based multi-agent pathfinding solvers: From classical to beyond classical. *Knowledge-Based Systems* 283 (Jan. 2024), 111121. https://doi.org/10.1016/j.knosys.2023.111121

[18] Peter Hart, Nils Nilsson, and Bertram Raphael. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 4, 2 (1968), 100–107. https://doi.org/10.1109/TSSC.1968.300136

[19] Julia Heise, Jens Weise, and Sanaz Mostaghim. 2023. Towards Benchmarking of Pathfinding Algorithms in Path-Influenced Environments. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. ACM, Lisbon Portugal, 69–70. https://doi.org/10.1145/3583133.3596434

[20] Hisao Ishibuchi, Ryo Imada, Yu Setoguchi, and Yusuke Nojima. 2018. How to Specify a Reference Point in Hypervolume Calculation for Fair Performance Comparison. *Evolutionary Computation* 26, 3 (Sept. 2018), 411–440. https://doi.org/10.1162/evco_a_00226

[21] Hisao Ishibuchi, Hiroyuki Masuda, Yuki Tanigaki, and Yusuke Nojima. 2015. Modified Distance Calculation in Generational Distance and Inverted Generational Distance. In *Evolutionary Multi-Criterion Optimization*, António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos Coello Coello (Eds.). Vol. 9019. Springer International Publishing, Cham, 110–125. https://doi.org/10.1007/978-3-319-15892-1_8 Series Title: Lecture Notes in Computer Science.

[22] Wei Liu, Rui Wang, Tao Zhang, Kaiwen Li, Wenhua Li, Hisao Ishibuchi, and Xiangke Liao. 2023. Hybridization of Evolutionary Algorithm and Deep Reinforcement Learning for Multiobjective Orienteering Optimization. *IEEE Transactions on Evolutionary Computation* 27, 5 (2023), 1260–1274. https://doi.org/10.1109/TEVC.2022.3199045

[23] R. Timothy Marler and Jasbir S. Arora. 2010. The weighted sum method for multi-objective optimization: new insights. *Structural and Multidisciplinary Optimization* 41, 6 (June 2010), 853–862. https://doi.org/10.1007/s00158-009-0460-7

[24] Annibale Panichella. 2019. An adaptive evolutionary algorithm based on non-euclidean geometry for many-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, Prague Czech Republic, 595–603. https://doi.org/10.1145/3321707.3321839

[25] Braxton Rolle, Ravi Kiran, and Jeremy Straub. 2022. A Pathfinding Algorithm for Lowering Infection Exposure of Healthcare Personnel Working in Makeshift Hospitals. *Healthcare* 10, 2 (Feb. 2022), 344. https://doi.org/10.3390/healthcare10020344

[26] V. Sathiya and M. Chinnadurai. 2019. Evolutionary Algorithms-Based Multi-Objective Optimal Mobile Robot Trajectory Planning. *Robotica* 37, 8 (2019), 1363–1382. https://doi.org/10.1017/S026357471800156X

[27] Alexey Skrynnik, Anton Andreychuk, Konstantin Yakovlev, and Aleksandr Panov. 2022. Pathfinding in stochastic environments: learning *vs* planning. *PeerJ Computer Science* 8 (Aug. 2022), e1056. https://doi.org/10.7717/peerj-cs.1056

[28] Roni Stern, Nathan Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Kumar, Roman Barták, and Eli Boyarski. 2021. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. *Proceedings of the International Symposium on Combinatorial Search* 10, 1 (Sept. 2021), 151–158. https://doi.org/10.1609/socs.v10i1.18510

[29] N. R. Sturtevant. 2012. Benchmarks for Grid-Based Pathfinding. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 2 (June 2012), 144–148. https://doi.org/10.1109/TCIAIG.2012.2197681

[30] Kaiwen Wang. 2024. Obstacle avoidance and path finding optimization model for wheeled logistics robots based on A* path finding algorithm. Shanghai, China, 030003. https://doi.org/10.1063/5.0215244

[31] Jens Weise. 2023. Evolutionary many-objective optimisation for pathfinding problems. http://dx.doi.org/10.25673/101389

[32] Jens Weise, Sebastian Mai, Heiner Zille, and Sanaz Mostaghim. 2020. On the Scalable Multi-Objective Multi-Agent Pathfinding Problem. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, Glasgow, United Kingdom, 1–8. https://doi.org/10.1109/CEC48606.2020.9185585

[33] Sandra Zajac and Sandra Huber. 2021. Objectives and methods in multi-objective routing problems: a survey and classification scheme. *European Journal of Operational Research* 290, 1 (April 2021), 1–25. https://doi.org/10.1016/j.ejor.2020.07.005

[34] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. 2001. *SPEA2: Improving the strength pareto evolutionary algorithm.* Technical Report. ETH Zurich. 21 p. pages. https://doi.org/10.3929/ETHZ-A-004284029 Artwork Size: 21 p. Medium: application/pdf.

[35] E. Zitzler and L. Thiele. 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation* 3, 4 (Nov. 1999), 257–271. https://doi.org/10.1109/4235.797969