Hybrid Adaptive Greedy Algorithm Addressing the Multi-Robot Path Planning Problem

Anikó Kopacz, Enol García González, Camelia Chira, and José R. Villar

***Abstract*—In the past few years, path planning and scheduling became a high-impact research topic due to their real-world applications such as transportation, manufacturing and robotics. This paper focuses on the Multi-robot Path Planning (MPP) problem, which consists of planning the route for a set of robots in a given static environment. The main goal is to navigate the robots from a starting point to a destination point without colliding with other robots or static obstacles. We propose a hybrid method – H* – that combines adaptive route planning based on A* and local search algorithm to optimize routes in the context of the MPP problem. The A* algorithm finds the optimal solution for the route search problem and a heuristic approach is applied to scale up to the multi-agent scenario. The overall length of determined paths and the number of robot collisions are minimized during the evaluations in specific small-scale environments –a room with 10x30 cells, four 10x30 cells rooms interconected two-by-two, and a realistic warehouse. Computational experiments are conducted for multi-robot scenarios and the performance of H* is compared to several path-searching algorithms including A* variations extended for the multi-agent scenario and coevolutionary algorithms. Experimental results demonstrate that H* outperforms the A* based heuristic approaches in terms of path length. H* shows similar performance as the coevolutionary method and performs better on smaller-scale maps.**

Link to graphical and video abstracts, and to code: https://latamt.ieeer9.org/index.php/transactions/article/view/9386

***Index Terms*—Multi-robot path planning, multi-agent systems, route planning, local search, greedy optimization**

I. Introduction

MANUFACTURING and transportation play a crucial role in the current economy, and innovation that focuses on the efficiency of these systems is a prevalent research area. Advances in technology have simplified the administration of logistic centers through the integration of robots in transportation operations. The navigation and coordination of multiple robots in the same environment is crucial to optimize the manufacturing process. The task of planning optimal paths for multiple robots within a shared environment, ensuring collision-free movement without human intervention, is called Collision-Free Multi-Robot Path Planning (MPP). Several domains encounter the MPP problem, including intelligent laboratories [1], [2], space exploration [3], warehouse management [4], manufacturing [5], unmanned aerial vehicles [6], [7], [8], and others.

The real-world need to solve the MPP problem is reflected in the literature, as many contributions address it and try to solve it differently. The MPP problem is indisputably related to the Single Robot Path Planning problem, which is about calculating the optimal path for a robot to reach a goal. This problem has been solved by finding the optimal route with the minimum time complexity using heuristic techniques such as CA*[9], M*[10], or D*[11], which are variations of the well-known A* algorithm to generalize it to multiple robots.

Although these techniques scale well when the size of the graph increases, their scalability decreases as the number of robots increases. Researchers have been motivated to develop solutions for the MPP scenario due to the industry's requirement to efficiently plan routes for a larger number of robots within a limited time frame. Heuristics and metaheuristics successfully applied to address the MPP problem include Particle Swarm Optimization [12], [13] or genetic algorithms [14], [15]. Furthermore, studies employing reinforcement learning have showcased a great potential in addressing the MPP problem, e.g. see [16], [17], [18], [19]. Nevertheless, studies that discuss the current solutions emphasize that there is potential for improvement; [20] discusses common limitations including high time-complexity (genetic algorithms, particle swarm optimization), reliance on selecting correct parameters (reinforcement learning) identifying suboptimal solutions, designed for static environments (A*) or using unrealistic distances in the heuristic function (D*).

The present work focuses on the MPP problem and aims to address the limitations of existing methods by exploring the use of hybrid approaches to create robust solutions. We propose a novel hybrid method named H* to construct collision-free routes for robots operating in parallel in various predefined scenarios. Two heuristic approaches are combined to optimize route planning and navigation in a grid-world environment. First, potential routes are constructed greedily, minimizing the path length; a variant of the A* algorithm [21] is applied to identify a candidate path for each robot. Then, the proposed routes are refined using a local search

The associate editor coordinating the review of this manuscript and approving it for publication was Alexandre S. Brandão (*Corresponding author: Enol García González*).

The authors Anikó Kopacz and Camelia Chira acknowledge the research support from the project "Romanian Hub for Artificial Intelligence - HRIA", Smart Growth, Digitization and Financial Instruments Program, 2021-2027, MySMIS no. 334906. The authors Enol García González and José R. Villar have been funded by the Spanish Ministry of Economics and Industry –grant PID2020-112726RB-I00–, the Spanish Research Agency –grant PID2023-146257OB-I00–, by Principado de Asturias –grant IDE/2024/000734–, and by the Council of Gijón through the University Institute of Industrial Technology of Asturias –grant SV-25-GIJÓN-1-23–.

A. Kopacz, and C. Chira are with the Faculty of Mathematics and Computer Science, Babeș-Bolyai University, Romania (e-mails: aniko.kopacz@ubbcluj.ro, and camelia.chira@ubbcluj.ro).

Enol García González, and J. R. Villar are with the Computer Science Department at University of Oviedo, Spain (e-mails: garciaenol@uniovi.es, and villarjose@uniovi.es).

operator focused on collision avoidance. H* is evaluated with respect to the length of the robot paths and the computational cost of obtaining the paths. We analyze the performance achieved with our approach compared to the effectiveness of heuristics [9] and metaheuristics [22], [14] for the route planning problem. We assess the effectiveness of the solutions proposed by the aforementioned methods in three different enclosed environments of varying complexity, with a number of robots ranging from 6 to 30.

The structure of the paper is as follows. Section II describes the MPP problem in general and Section III provides a concise overview of current methods. Then, Section IV details the H* method and presents the general algorithm. Section V describes the experimental setup and the results of the experiments obtained with the H* method for various multi-robot scenarios. The performance of the H* algorithm is compared to heuristic and metaheuristic approaches from the literature that tackle the MPP problem. Finally, Section VI presents the conclusions drawn from this study and presents opportunities for further improvement of this contribution.

II. Problem Statement

Nowadays, autonomous robots and navigation systems with multiple actors play a key role in industrial applications, such as specialized warehouse transportation and manufacturing. Oftentimes, navigation system controllers and planning are designated to schedule routes of robots and coordinate the movement and direction of robots in order to avoid collisions and minimize system delays.

The multi-robot path planning problem [14], [20], [22], [23] consists of two computational tasks that need to synchronize:

1) determining the individual paths for each robot in a given environment and
2) coordinating the actions of the robots to minimize collisions or delays.

The path-searching subtask considering multiple infrastructures can be represented using graphs or networks; thus, well-known graph-searching methods are suitable for solving the path searching problem, such as A^* [21] and Dijkstra [24]. The coordination and scheduling of robot actions can be assessed with operational research methods, e.g., dynamic programming [25]. Dynamic programming provides the optimal solution for coordination tasks; however, in case of real-world problems, heuristic approaches are often favored due to the complexity and computational demands of the exact solution. Several heuristic methods were proven suitable to assess the multi-agent nature of the path-planning problem – among other solutions for MPP, we discuss heuristic solvers in Section III.

Robots are placed in a grid-world environment, where the movement of the robots is restricted by walls and other robots. Agents are required to avoid colliding with permanent and temporary obstacles. Permanent obstacles – also referred to as walls of the maze – are known and specific to a given environment setup. In this context, temporary obstacles constitute of other robots. Each robot is able to move from one cell to an adjacent one in four directions (left, up, right, down) or stay on the cell. The robots operate in a parallel manner, and robots execute one of the five possible actions in every time-step of the simulations. Fig. 1 shows a two-dimensional map configuration proposed in [15] and possible paths for three robots that navigate the maze. While the routes are overlapping and robots traverse the same cells, collisions will not occur, because the robots arrive at the respective cells in different time-steps.

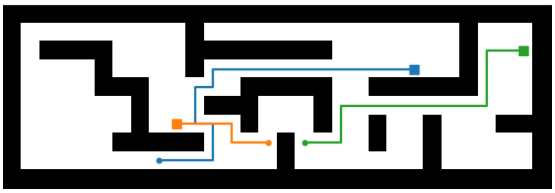


Fig. 1. Example grid-world environment with 3 robots and their routes. Initial positions are marked with circles, target positions are squares.

Given the map of the environment, the initial and target positions of the robots, multiple simulations (episodes) can be performed to analyze planning algorithms and strategies. The performance of methods targeting path planning can be compared prioritizing various objectives. In order to analyze the effectiveness of an algorithm to avoid collisions, the number of robots that did not reach their target can be used as a metric. Evaluation metrics – that aggregate the effectiveness of robots in the context of reaching their destination – synthesize the goal of minimizing resources consumed to traverse a given route, and the goal of avoiding collisions include, e.g. the average length of robot paths, the longest robot path.

In many industrial applications, navigation and planning systems are designated to identify routes and coordinate actions to minimize system delays. Although resource allocations for operating robots is best to keep low, additional resources required for detours to avoid collisions are justifiable. Robot collision can be avoided by halting movement or re-routing, which strategy is preferred depends on the application.

III. Related Work

The problem of multi-robot path planning (MPP) involves finding paths for multiple robots. Although the single robot path planning problem is relatively easy to solve using algorithms such as Dijkstra [24], Floyd-Warshall [26], and A* [21], the MPP problem is much more complex due to the presence of multiple agents. The increase in complexity requires the development of new algorithms and approaches to address the problem efficiently.

The initial studies focus on variations of methods used to address the single-robot path planning problem, as it bears close resemblance. Many of the existing proposals in this category are variations of the A* algorithm. For example, Cooperative A* [9] and its variants, Cooperative A* (CA*), Hierarchical Cooperative A* (HCA*), and Windowed Hierarchical Cooperative A* (WHCA*), use the A* algorithm to compute individual robot paths and apply a heuristic strategy to resolve potential collisions. In case of CA*, a reservation table marks all planned cells as occupied by one robot, forcing all other robots that want to traverse the same cells to stand

idle and wait until the cells are released. HCA* proposes to calculate robot routes in reverse order, from destination to start, to reduce the number of waiting times that occur using CA*. Both CA* and HCA* have the problem of prioritizing one robot over the rest, resulting in one robot having no waiting time and another having many waiting movements. WHCA* modifies A* so that in each iteration not only is one node expanded but a time window is evaluated, and waiting is incorporated into the robot. WHCA* introduces a new parameter, specifically, the inclusion of the window size requires consideration. Obtaining better routes is possible with larger window sizes, but this comes at the cost of increased algorithm complexity. On the other hand, when the window size is set to 1, the algorithm's behavior is identical to that of HCA*.

Another promising method that extends the A* approach is M* [10]. With M*, the authors propose a modification on A* in which all routes are calculated in parallel using A*. To solve the problem of collisions between robots, expanding nodes that cause collisions is not allowed. As a consequence, when a collision is detected in the calculation of the routes, the agent is forced to retrace its steps through the network and try another path. A caveat of the M* approach is that, in complex environments, going back multiple times and re-planning routes can cause the execution time to grow rapidly. Furthermore, because the re-planning step is performed on all the robots that collide, a collision-free solution is not guaranteed, because it is possible that colliding robots choose to traverse the same positions again.

Another similar proposal to M* is the D* family of algorithms: D* [11], LiteD* [27], FieldD* [28], and FocusedD* [29]. The overall algorithm of this category closely resembles M*, as it starts by concurrently expanding all the paths of robots within the environment. The key difference compared to M* lies in the collision resolution strategy. In the case of M*, collisions were resolved by blocking the path and step back to find an alternative route. D* suggests replacing blocking with a penalty. This involves modifying the cost of the area where the collision occurs and propagating the penalty backward until the conflict is resolved. This modification allows robots involved in the collision to avoid the need for re-planning trajectories and instead proceed along the same path. Robots that find an alternative route will change the course of their path to avoid going in the direction of the conflicting positions due to the significant associated cost.

One major drawback that has been identified in the existing literature is the significant computational time required [30], [31]. For WHCA*, achieving high-quality solutions requires setting a large window size. On the other hand, M* is similar to brute-force search due to the significant slowdown caused by backtracking, resulting in a substantial decrease in node expansion.

In the context of industrial applications such as transportation, planning systems are crucial to effectively handle an increasing number of robots. This capability is necessary to minimize the expenses associated with scaling and upgrading the system. In order to provide viable solutions within a reduced timeframe, many authors have opted to use metaheuristics to try to solve the MPP problem.

Most of the published proposals for solving the MPP problem using metaheuristics are papers that present hybridizations of Particle Swarm Optimization (PSO) with other metaheuristics. PSO is a metaheuristic that mimics the natural behavior of many species of animals that demonstrate collective intelligence. The application of collective intelligence involves updating the solutions in a population based on the best solution discovered so far and the current best solution among the other individuals in the population. Several suggestions in the literature have explored the combination of PSO with other optimization algorithms such as Grey Wolf Optimizer [32], Differential Perturbed Velocity [13], or Gravitational Search [12]. In these hybrid approaches, one of the metaheuristics is employed to generate feasible routes, while the other is utilized to optimize these routes and resolve any collisions that may arise.

It is also common practice to employ genetic algorithms to solve the MPP problem. For example, [15], [14] apply genetic algorithms hybridized with other metaheuristics, namely Artificial Potential Fields [33], and with heuristics, namely A*. Similarly as with PSO, upon hybridizing genetic algorithms, the complementary technique generates feasible routes, then the genetic algorithm is applied to optimize the routes and eliminate collisions.

In recent years, reinforcement learning has been proven to be suitable for addressing a wide range of problems related to navigation and route planning. In [34], a reinforcement learning setup is applied to solve the single robot path-finding problem. In the study conducted by [35], robots were taught how to navigate different types of maze using a deep reinforcement learning technique. Multi-robot cooperative deep reinforcement learning [36] using robot-centered images as input shows promising results in navigation and collision avoidance tasks. Deep Q-learning in combination with convolutional neural networks was applied in [16] to the MPP problem. To facilitate a faster convergence of the deep Q-learning method for MPP, in [17] a path is calculated for a single agent, and the path obtained corresponds to prior knowledge for the neural network. Another reinforcement learning method, proximal policy optimization, has shown an effective application in dynamic environments, and transfer learning techniques are employed to facilitate adaptation to novel environments [19].

IV. Proposed Method

This section presents the H* algorithm that addresses the MPP problem by identifying collision-free routes for multi-robot settings. Fig. 2 illustrates the general process of the proposed H* method. We define the following steps for optimizing route planning in a multi-agent scenario:

1) Greedy path search and
2) Local search operator.

It is important noting that both steps of the proposed algorithm have polynomial complexity. The greedy path search is implemented using a heuristic approach to generate routes for robots given a maze, knowing the initial and target positions of robots. As detailed in Subsection IV-A, routes are determined

following a greedy approach for each robot, minimizing the length of the paths and the number of collisions along the selected paths. The greedy algorithm is designed to create the initial routes, minimizing the two aforementioned objectives. The second stage applies local search to enforce the optimization considering the two objectives as well; this stage is detailed in Subsection IV-B.

A. Greedy Path Search

Graph search algorithms, such as A^*, find the shortest path for setups with a single robot. A straightforward approach to applying well-known graph searching for the multi-robot path searching problem in a grid world is considering the maze an undirected graph, where a node represents each accessible position on the map, and edges denote the adjacency of cells. However, routes obtained by graph-searching algorithms on the map do not take into account to avoid other agents and may even block some of them. One of the heuristic approaches included in the path search process that facilitates considering other robot actions is excluding cells occupied by other robots from the graph-search.

After analyzing the shortest routes identified by A^* for different layouts of the map, several cell categories can be distinguished based on the circumstances in which the cell is added to the shortest route. Let c_i denote a position in a grid world environment that can be traversed by a robot from the MPP setting, where $i \in \{d, c, j\}$ marks the type of cell (d is shorthand for dead-end, c is for corridor and j stands for junction). In this article, cells that are adjacent by having a common edge with the c cell are considered neighbors; the neighborhood of cell c_i is denoted by n_k, where $1 \leq k \leq N$, N being the number of neighbors of c_i.

Any c_d position marking a cell with exactly one neighbor (also referred to as *dead-ends*, see Fig. 3a) appear in the shortest routes constructed by A^* only if c_d is either

1) the robot's initial position or
2) its target.

Dead-ends are avoided in the remaining cases due to the fact that traversing the cell would add a detour to the optimal paths. Moreover, any concave obstacle that does not contain starting positions or destinations can be reduced to a dead-end region by growing cell dimensions within it.

A position c_c that has exactly two adjacent cells may appear in shortest paths constructed by A^*, however, it is crucial to note that the shortest path contains the sequence $n_1 - c_c - n_2$ or $n_2 - c_c - n_1$, with neighbors n_1 and n_2 of c_c. Thus, robots choose to visit the cell c_c only if they also visit its neighbors. Configurations where a sequence of adjacent cells have exactly 2 neighbors, are also referred to as corridors; Fig. 3b features an example of such a configuration. The disposition c_c acts as an intermediate position between n_1 and n_2, therefore, from the perspective of route planning and action selection, one could consider c_c as information assigned to an edge between the n_1 and n_2 nodes. This property holds for a sequence of $c_c^1, c_c^2, ..., c_c^m$, where every $c_c^i, 1 \leq i \leq m$ has two neighbors, and every c_c^i cell is connected to $c_c^{i+1},\ 1 \leq i < m$. Lastly, cells with at least three distinct neighbors act as junctions, and their neighbors must be evaluated to select them to the shortest path (see Fig. 3c).

We constructed a simplified graph representation of the grid-world environment based on the merging of cells that have exactly two neighbors. A sequence of grid positions, where every element of the sequence has exactly 2 adjacent cells, will be merged into a single graph node. In the representation graph, the original order of the merged dispositions is also saved in order to ease planning to traverse the merged cells. Positions that do not meet this criterion are added as separate nodes in the graph constructed from the map. If c is an intermediate position in the sequence $n_1 - c - n_2$, but only c has 2 neighbors, and both n_1 and n_2 have 1 or at least 3 adjacent cells, then c will be a distinct node in our representation graph. In this case, the c cell is included as a separate node, so the robots are not forced to select the $c - n_2$ route when stepping in the n_1 position and may continue their path by selecting another neighbor of n_1.

Given the simplified graph representation of the grid, A^* is suitable to determine the shortest paths for each robot, without considering the multi-agent component of the problem. In order to minimize the likelihood of collisions or blocking other robots, the current positions of the robots are marked as occupied nodes in the representation graph, and A^* is modified to avoid positions or merged positions marked as occupied. The current position of a robot is also deemed occupied; as a consequence, we eliminate routes that circle back to the current position of the robot. Robot paths are constructed independently, meaning that robots do not have information about future actions and dispositions of other robots. Thus, the graph search takes into consideration the current positions of other agents, and future positions of robots in subsequent time-steps are not estimated.

Algorithm 1 summarizes the selection of movements for a given robot by constructing routes to the robot's destination. All n_k cells adjacent to the robot's current positions are selected as the next positions (see line 9), then A^* is applied to determine the shortest route from n_k to the destination. A greedy selection is applied to determine the next movement from the constructed routes: the robot is trying to minimize the length of the route and chooses the movement that is associated with the least number of steps that lead to the target (see lines 11-13). The robot paths consist of the total sequence of movements and are refined by local search.

B. Local Search Operator

As mentioned above, the greedy algorithm described in Section IV-A generates candidate routes that may contain collisions. The local search operator optimizes the paths generated by the greedy path search and eliminate existing collisions by using stop operations. The local search operator consists of two components:

1) loop elimination and
2) collision resolution.

Algorithm 2 eliminates loops from a robot path. The list of coordinates is checked for repeating entries (lines 6-8). If

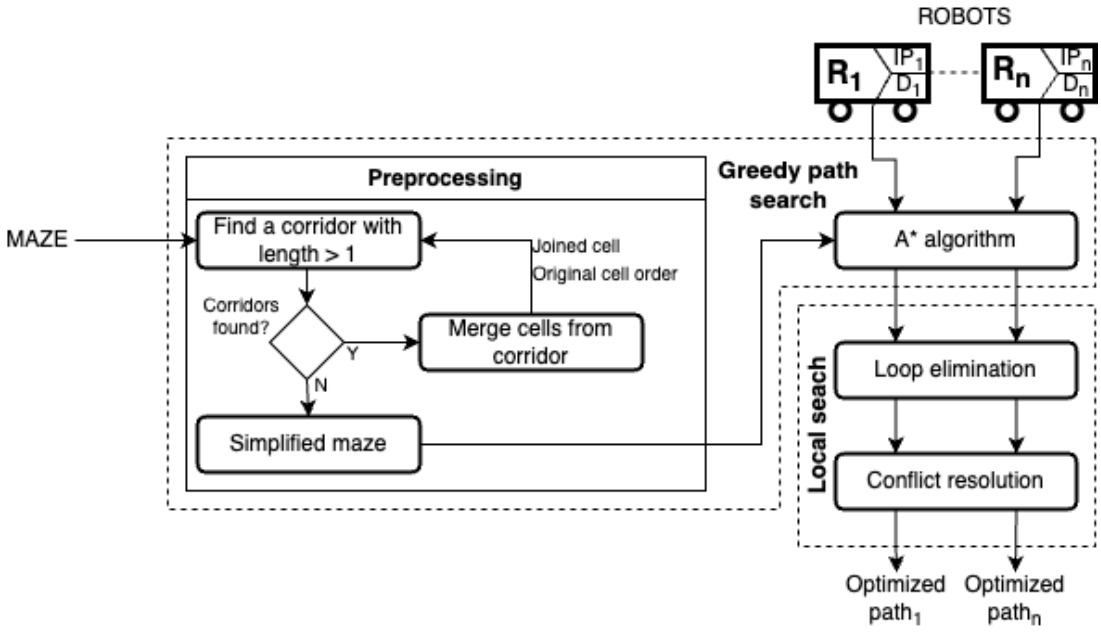


Fig. 2. General overview of H*: the greedy path search determines possible routes for the robots; then the local search operator is applied to optimize the obtained routes via loop elimination and conflict resolution. IP_k and D_k are the initial and final position for robot k.

Algorithm 1 Greedy path search

Input
  r ▷ current robot
  graph ▷ a graph representation of the grid-world
Output
  best_path ▷ route selected for the r robot
$c_r \leftarrow$ current position of the r
$best\_path \leftarrow \emptyset$
for $k := 1$ to N **do** ▷ select all neighbors of c_r
  $path \leftarrow [c_r, n_k]$
  $path \leftarrow$ complete $path$ to $target_r$ with A^*
  if $path$ is shorter than $best\_path$ or $best\_path = \emptyset$ **then**
    $best\_path \leftarrow path$
  end if
end for

Algorithm 2 Loop elimination

Input
  path ▷ a single path to optimize that may contain loops.
Output
  path ▷ Optimized path that no longer contains loops
n $\leftarrow$ len(path)
for $i := n$ to 1 step -1 **do**
  for $j := 1$ to i step 1 **do**
    if $path_i = path_j$ **then**
      Remove all positions between j and i
    end if
  end for
end for

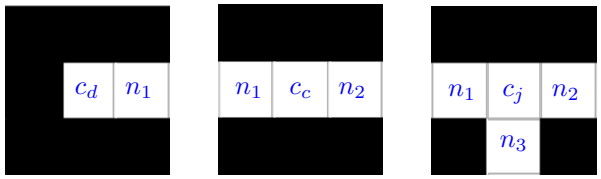


Fig. 3. Cell layouts and the relationship their neighbours encountered during shortest path search in the grid-world environment (c_i denotes the characterized cell for the three distinct configurations and n_k denotes its' neighbors): a) A dead-end (c_d) has 1 neighbor (shown as n_1), b) A corridor (c_c) has 2 neighbors (n_1 and n_2), and c) A junction (c_j) has 3 neighbors (n_1, n_2 and n_3).

a coordinate appears twice along the robot's path, a loop is detected and any actions taken between can be eliminated.

After eliminating any loops from the paths, collisions are identified and addressed by adding halt operations, which stops the robot's movement for one time-step. To detect collisions, the positions occupied by robots are calculated for each instance of time. By expressing the robot positions as a function of time, in the following two situations, a collision occurs:

- Two robots have the same position at the same time-step,
- Two robots exchange their position at two consecutive time-steps.

Algorithm 3 eliminates collisions iteratively from the list of robot paths given as input until no collisions are left. If any two paths have collisions and the first collision is selected and resolved by Algorithm 4 (see line 8).

Algorithm 4 receives two robot paths that have a collision, the time-step t when the collision occurs and tries to eliminate it by introducing waits. An iterative process is used that begins

at the t time-step and moves away by one unit of distance with each iteration (see line 8). Each iteration will attempt to resolve the collision by inserting waiting operations equal to the distance to the collision (lines 9-10), Ivi.e. in the first iteration 1 halt operation is inserted, in the second 2 halt operations are inserted, and so on. In each iteration, it is verified that the collision disappears by adding the wait to the first robot (lines 11-15). If that does not work, it will be removed from the first robot, and an attempt will be made to resolve it by adding it to the second robot (lines 16-20).

Algorithm 3 General algorithm for collision elimination

1: **Input**
2: paths ▷ Optimized paths without loops
3: **Output**
4: paths ▷ Paths without collisions
5: $collisions \leftarrow$ Detect collisions on $paths$
6: **while** $collisions \neq \emptyset$ **do**
7: $paths \leftarrow$ Solve first collision on $collisions$ using Alg. 4
8: $collisions \leftarrow$ Detect collisions on $paths$
9: **end while**

Algorithm 4 Collision elimination for a single collision

1: **Input**
2: path1 ▷ One of the paths that causes the collision
3: path2 ▷ The other path that causes the collision
4: t ▷ Index of the position where the collision happens
5: **Output**
6: path1 ▷ New path for path1
7: path2 ▷ New path for path2
8: **for** $i := 1$ to t step 1 **do**
9: $new\_path1 \leftarrow$ Insert i number of halt operation on $path1$ at position $t - i$
10: $new\_path2 \leftarrow$ Insert i number of halt operation on $path2$ at position $t - i$
11: $n\_collisions \leftarrow collissions(new\_path1, path2)$
12: **if** $n\_collisions = 0$ **then**
13: $path1 \leftarrow new\_path1$
14: Terminate algorithm
15: **end if**
16: $n\_collisions \leftarrow collissions(path1, new\_path2)$
17: **if** $n\_collisions = 0$ **then**
18: $path2 \leftarrow new\_path2$
19: Terminate algorithm
20: **end if**
21: **end for**

V. Experiments and Results

Computational experiments are performed for several scenarios that consider different characteristics and impose various levels of difficulty for planning methods. We selected three distinct multi-robot map layouts to assess the effectiveness of the H* hybrid method in the context of the MPP problem.

TABLE I
Relation of the Number of Robots in the Instances of each Scenario for the Experimentation

Scenario	Initial number of robots	Maximum number of robots	Step size
1	6	15	1
2	12 (3 per room)	24 (6 per room)	4 (1 per room)
3	10	30	5

A. Evaluated Environments

Scenario 1 represents the toy problem shown on Fig. 4a was proposed in [15] to evaluate the performance of MPP methods. Obstacles are placed that form loosely defined rooms of various sizes and small corridors. In the experiments shown in the current paper, this scenario represents a low-complexity setup given the size of the map and the semi-structured nature of obstacle placement. Smaller rooms and the presence of several corridors are favorable for exhaustive searches, because the number of possible solutions is confined. We report the performance of route-searching methods to validate the accuracy of the constructed robot paths.

Scenario 2 depicted in Fig. 4b was proposed in [14] and investigates whether optimization policies resolve conflicts caused by multiple robots traversing the same corridors in real time. The maze comprises four open rooms without obstacles connected by narrow corridors. The initial and ending positions of the robots are distributed among the different rooms, forcing all robots to cross the corridors, increasing the number of possible collisions.

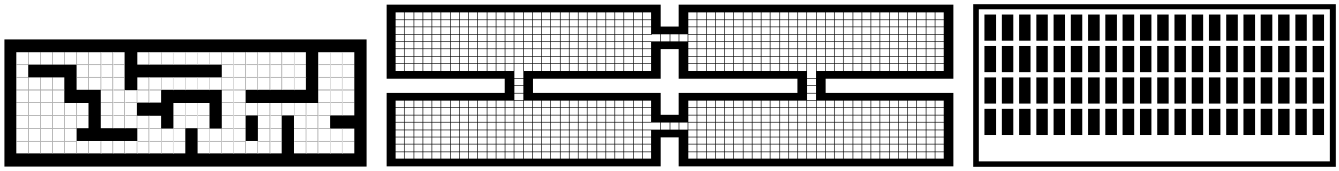
Scenario 3 represents a real-world application of the MPP problem [37], simulating an industrial warehouse where robots must go to a shelf to retrieve an object (see Fig. 4c). There are two different zones: a storage zone and a work zone. The storage area consists of shelves considered obstacles from the robots' point of view, forming a network of long, narrow corridors. The work zone is an open, unobstructed area. In this scenario, every robot starts the simulations from a disposition in the dedicated work zone and heads toward a designated point in the storage area.

B. Experimental Setup and Evaluation Criteria

Each scenario will be evaluated with a varying number of robots. Table I presents the details of the cases examined for each scenario, which includes the minimum number of robots, i.e. the most sparse test case, the maximum number of robots, i.e. the most dense test case, and the incremental increase in the number of robots until reaching the maximum.

For each scenario and the number of robots, we evaluate an instance of the MPP problem characterized by the starting points and destinations of each robot. During the experiments, each configuration is evaluated 10 times using the H* method, then the results are averaged by scenarios and number of robots, and the following metrics are reported:

- The length of the longest path for each configuration.
- The aggregated sum of the path lengths of all robots.



(a) (b) (c)

Fig. 4. Layouts in this experimentation. a) Scenario 1, evaluated for $6, 7, ..., 15$ robots; b) Scenario 2, evaluated for $3, 4, 5, 6$ robots per room; and c) Scenario 3, evaluated for $10, 15, 20, 25, 30$ robots.

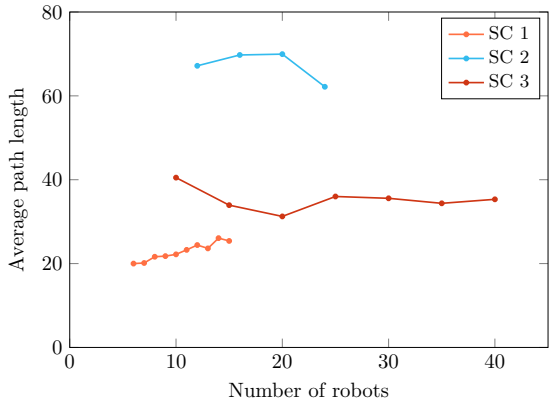


Fig. 5. Average path length per configuration compared to the number of robots from Scenario (SC) 1, 2 and 3.

C. Results

Fig. 5 shows the average path length in relation to the number of robots evaluated for all configurations from Scenario 1, 2 and 3. The H* method scales well with regards to the number of robots in all three scenarios. In the case of Scenario 1, a gradual increase can be seen in the average path length. Nevertheless, for Scenario 2 and 3 the average path length does not follow the increase in the number of robots. The H* method scales well when increasing the number of robots in Scenario 1, 2 and 3 having a minimal impact on the average route length.

D. Comparison and Discussion

We report the performance of the H* method presented in this article and highlight the effectiveness of the approach compared to three different methods from the literature that tackle the MPP problem. The WHCA [9] algorithm extends the A* algorithm to the multi-robot setting using a heuristic approach. Enol2023 [14] is a hybrid method that combines A* with a co-evolutionary algorithm for route planning. Lastly, we include NSGA-III [22] for comparison. NSGA-III starts from a set of pregenerated randomized routes and applies multiobjective evolutionary optimization to identify a combination of routes that minimize the path length and number of collisions, the same criteria that have been used for our H* proposal. The best solutions reported by NSGA-III [22] are selected from a set of non-dominated feasible solutions which have no collisions and therefore address the second objective related to the number of collisions to optimality.

TABLE II
THE MAXIMUM AND AGGREGATED SUM OF THE LENGTHS OF ROBOT PATHS DETERMINED BY CO-EVOLUTIONARY, HEURISTIC AND THE H* METHOD. BLANK CELLS CORRESPOND TO THE CASES WHERE THE DETERMINED ROBOT PATHS HAVE COLLISIONS. SC STANDS FOR THE SCENARIO ID, CW MEANS CURRENT WORK. BEST SOLUTIONS IN BOLD

	Enol2023	**WHCA**	**NSGA-III**	**H* cw**
SC 1	Max / Sum	Max / Sum	Max / Sum	Max / Sum
6	55.0 / 363.9	**32.0** / **116.0**	**32.0** / 120.0	33.0 / 120.0
7	56.4 / 377.1	**32.0** / 141.0	**32.0** / 146.2	33.0 / **141.0**
8	57.2 / 438.5	**32.0** / **161.0**	**32.0** / 174.6	39.0 / 173.0
9	60.5 / 389.1	**32.0** / **178.0**	36.0 / 195.2	38.0 / 196.0
10	59.2 / 444.5	36.0 / **214.0**	50.2 / 248.3	**34.0** / 222.0
11	68.9 / 414.7	**41.0** / 260.0	50.6 / 300.7	46.0 / **256.0**
12	63.1 / 426.1	/	52.1 / 338.8	**40.0** / **293.0**
13	79.0 / 437.8	/	54.3 / 387.6	**40.0** / **307.0**
14	78.4 / 469.8	/	62.0 / 412.2	**46.0** / **365.0**
15	90.8 / 486.0	/	70.1 / 489.0	**46.0** / **381.0**
SC 2	Max / Sum	Max / Sum	Max / Sum	Max / Sum
3×4	77.0 / **736.0**	/	**76.0** / 788.3	108.0 / 806.0
4×4	**79.0** / **951.0**	/	82.2 / 920.2	111.0 / 1116.0
5×4	**79.0** / **1206.0**	/	88.0 / 1336.9	111.0 / 1399.0
6×4	**79.0** / **1458.0**	/	/	118.0 / 1492.0
SC 3	Max / Sum	Max / Sum	Max / Sum	Max / Sum
10	**62.0** / 399.0	**62.0** / **390.0**	70.0 / 415.1	64.0 / 405.0
15	**62.0** / 536.0	62.0 / 541.0	140.2 / 740.9	69.0 / **509.0**
20	**62.0** / 639.0	62.0 / 651.6	236.7 / 1339.5	73.0 / **625.0**
25	**62.0** / **808.0**	/	248.6 / 1662.3	85.0 / 900.0
30	**62.0** / **946.0**	/	253.3 / 1810.8	85.0 / 1067.0

Table II summarizes the results of the experiments for the proposed H* method compared to other experiments for the heuristic and metaheuritic approaches, as reported in [22]. The average length of the longest route (Max) and the average sum of all robot paths (Sum) metrics are calculated for the 10 runs of the robot configurations; these two metrics were chosen because , on the one hand, they provide a clear picture of the quality of the solutions; on the other hand, all methods avoid collisions except the WHCA, so the number of collisions becomes 0 in almost all of the cases. In Scenario 1, H* finds routes shorter than Enol2023 with respect to the longest robot path and the aggregated sum of all paths. H* is surpassed by the WHCA* and NSGA-III heuristics for instances with 6 to 9 robots for both Max and Sum robot paths. However, as the number of robots increases, H* continues to find collision-free routes, while WHCA does not determine the collision-free path for configurations with 12 robots or more for Scenario 1 and Scenario 2. In case of Scenario 2, Enol2023 has the shortest Max path, WHCA is the second best ahead of H* with regards

to the Max path lengths. H* outperforms NSGA-III, but does not surpass Enol2023 for Scenario 3 for path length metrics. WHCA finds routes shorter than H * up to configurations with 20 robots; for configurations with 25 or 30 robots, WHCA does not obtain a collision-free solution and H* becomes the method that obtains the second best results.

The experimental results confirm that – similarly to Enol2023 – H* is robust with regard to an increase in the number of robots within scenarios; these are the only two methods out of the four analyzed that are capable of addressing all instances of the problem. The proposal in this work, H*, has proven to be the best for solving problems such as those from Scenario 1, surpassing Enol2023. Although in Scenario 3, Enol2023 continues to be the best solution, H* is the second-best option by improving the results of WHCA* in both time and path length.

During the experimentation, a hypothesis test was performed to validate that the analysis presented in the discussion in the previous paragraphs was significant. This hypothesis test was done using a Mann-Whitney U-Test to evaluate the significance of the observed differences. All the hypothesis test confirmed that the values obtained show statistically significant differences, which supports the comparability of the results.

VI. Conclusion

Multi-robot systems have a wide range of real-life applications where the resources required to adapt to changes in circumstances are limited, for instance, transportation and manufacturing. Heuristic approaches can be applied to decrease the computational demand for finding solutions that adapt to settings with increasing robot densities. This paper introduces the H* algorithm, a hybrid adaptive method that addresses the Multi-robot Path Planning (MPP) problem. Experimental results are promising and show that H* outperforms heuristic A* based methods from the literature.

In future work, we focus on evaluating the impact of the local search algorithm on improving the solutions determined by various metaheuristics. Part of our future efforts will be directed toward investigating the impact of local search on enhancing co-evolutionary methods from the literature, such as M* [38] or WHCA [9]. We also intend to improve the H* algorithm to increase its effectiveness while maintaining the low resource requirements of the approach and apply machine learning methods to further enhance the proposed robot paths, e.g. reinforcement learning. Comparing the performance of this proposal against using the number of collisions as a constraint will be addressed.

Acknowledgments

The authors Anikó Kopacz and Camelia Chira acknowledge the research support from the project "Romanian Hub for Artificial Intelligence - HRIA", Smart Growth, Digitization and Financial Instruments Program, 2021-2027, MySMIS no. 334906. The authors Enol García González and José R. Villar have been funded by the Spanish Ministry of Economics and Industry –grant PID2020-112726RB-I00–, the Spanish Research Agency –grant PID2023-146257OB-I00–, by Principado de Asturias –grant IDE/2024/000734–, and by the Council of Gijón through the University Institute of Industrial Technology of Asturias –grant SV-25-GIJÓN-1-23–.

References

[1] Q. Tan, M. Denojean-Mairet, H. Wang, X. Zhang, F. C. Pivot, and R. Treu, "Toward a telepresence robot empowered smart lab," *Smart Learning Environments*, vol. 6, no. 1, p. 5, May 2019. [Online]. Available: https://doi.org/10.1186/s40561-019-0084-3

[2] S. Solak, Ö. Yakut, and E. Dogru Bolat, "Design and implementation of web-based virtual mobile robot laboratory for engineering education," *Symmetry*, vol. 12, no. 6, 2020. [Online]. Available: https://doi.org/10.3390/sym12060906

[3] D. Huang, H. Jiang, Z. Yu, C. Kang, and C. Hu, "Leader-following cluster consensus in multi-agent systems with intermittence," *International Journal of Control, Automation and Systems*, vol. 16, no. 2, pp. 437–451, Apr 2018. [Online]. Available: https://doi.org/10.1007/s12555-017-0345-2

[4] N. V. Kumar and C. S. Kumar, "Development of collision free path planning algorithm for warehouse mobile robot," *Procedia Computer Science*, vol. 133, pp. 456–463, 2018, international Conference on Robotics and Smart Manufacturing (RoSMa2018). [Online]. Available: https://doi.org/10.1016/j.procs.2018.07.056

[5] H. Shen, L. Pan, and J. Qian, "Research on large-scale additive manufacturing based on multi-robot collaboration technology," *Additive Manufacturing*, vol. 30, p. 100906, 2019. [Online]. Available: https://doi.org/10.1016/j.addma.2019.100906

[6] E. Stump and N. Michael, "Multi-robot persistent surveillance planning as a vehicle routing problem," in *2011 IEEE International Conference on Automation Science and Engineering*, 2011, pp. 569–575. [Online]. Available: https://10.1109/CASE.2011.6042503

[7] X. Chen and B. Zhou, "A heuristics pulse algorithm with relaxation pruning strategy for resources re-initialized uav path planing," *Journal of Intelligent & Fuzzy Systems*, vol. 41, pp. 3541–3553, 2021. [Online]. Available: https://doi.org/10.3233/JIFS-210901

[8] R. Liu, J. Liang, and M. Alkhambashi, "Research on breakthrough and innovation of uav mission planning method based on cloud computing-based reinforcement learning algorithm," *Journal of Intelligent & Fuzzy Systems*, vol. 37, pp. 3285–3292, 2019. [Online]. Available: https://doi.org/10.3233/JIFS-179130

[9] Z. Bnaya and A. Felner, "Conflict-oriented windowed hierarchical cooperative a," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3743–3748. [Online]. Available: https://doi.org/10.1109/ICRA.2014.6907401

[10] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 3260–3267. [Online]. Available: https://doi.org/10.1109/IROS.2011.6095022

[11] A. Stentz and I. C. Mellon, "Optimal and efficient path planning for unknown and dynamic environments," *International Journal of Robotics and Automation*, vol. 10, no. 3, pp. 89–100, 1995. [Online]. Available: https://doi.org/10.1007/978-1-4615-6325-9_11

[12] P. Das, H. Behera, and B. Panigrahi, "A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning," *Swarm and Evolutionary Computation*, vol. 28, pp. 14–28, 2016. [Online]. Available: https://doi.org/10.1016/j.swevo.2015.10.011

[13] P. Das, H. Behera, S. Das, H. Tripathy, B. Panigrahi, and S. Pradhan, "A hybrid improved pso-dv algorithm for multi-robot path planning in a clutter environment," *Neurocomputing*, vol. 207, pp. 735–753, 2016. [Online]. Available: https://doi.org/10.1016/j.neucom.2016.05.057

[14] E. García, J. R. Villar, Q. Tan, J. Sedano, and C. Chira, "An efficient multi-robot path planning solution using a* and coevolutionary algorithms," *Integrated Computer-Aided Engineering*, vol. 30, pp. 41–52, 2023. [Online]. Available: https://doi.org/10.3233/ICA-220695

[15] M. Kiadi, E. García, J. R. Villar, and Q. Tan, "A*-based co-evolutionary approach for multi-robot path planning with collision avoidance," *Cybernetics and Systems*, vol. 0, no. 0, pp. 1–16, 2022. [Online]. Available: https://doi.org/10.1080/01969722.2022.2030009

[16] H. Bae, G. Kim, J. Kim, D. Qian, and S. Lee, "Multi-robot path planning method using reinforcement learning," *Applied Sciences*, vol. 9, no. 15, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/15/3057

[17] Y. Yang, L. Juntao, and P. Lingling, "Multi-robot path planning based on a deep reinforcement learning DQN algorithm," *CAAI Transactions on Intelligence Technology*, vol. 5, no. 3, pp. 177–183, 2020. [Online]. Available: https://doi.org/10.1049/trit.2020.0024

[18] Y. Wei and R. Zheng, "Multi-robot path planning for mobile sensing through deep reinforcement learning," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10. [Online]. Available: https://doi.org/10.1109/INFOCOM42981.2021.9488669

[19] S. Wen, Z. Wen, D. Zhang, H. Zhang, and T. Wang, "A multi-robot path-planning algorithm for autonomous navigation using meta-reinforcement learning based on transfer learning," *Applied Soft Computing*, vol. 110, p. 107605, 2021. [Online]. Available: https://doi.org/10.1016/j.asoc.2021.107605

[20] Ángel Madridano, A. Al-Kaff, D. Martín, and A. de la Escalera, "Trajectory planning for multi-robot systems: Methods and applications," *Expert Systems with Applications*, vol. 173, p. 114660, 2021. [Online]. Available: https://doi.org/10.1016/j.eswa.2021.114660

[21] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. [Online]. Available: https://doi.org/10.1109/TSSC.1968.300136

[22] E. G. González, J. R. Villar, C. Chira, E. A. de la Cal, L. Sánchez, and J. Sedano, "Multi-objective optimization for multi-robot path planning on warehouse environments," in *18th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2023)*, ser. Lecture Notes in Networks and Systems, vol. 750, 2023, pp. 279–289. [Online]. Available: https://doi.org/10.1007/978-3-031-42536-3_27

[23] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar, and E. Boyarski, "Multi-agent pathfinding: Definitions, variants, and benchmarks," *Proceedings of the International Symposium on Combinatorial Search*, vol. 10, pp. 151–158, 09 2021. [Online]. Available: https://doi.org/10.1609/socs.v10i1.18510

[24] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959. [Online]. Available: https://doi.org/10.1007/BF01386390

[25] R. Bellman, "The theory of dynamic programming," *Bulletin of the American Mathematical Society*, vol. 60, pp. 503–515, 1954. [Online]. Available: https://doi.org/10.1090/S0002-9904-1954-09848-8

[26] R. W. Floyd, "Algorithm 97: Shortest path," *Commun. ACM*, vol. 5, no. 6, p. 345, jun 1962. [Online]. Available: https://doi.org/10.1145/367766.368168

[27] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005. [Online]. Available: https://doi.org/10.1109/TRO.2004.838026

[28] D. Ferguson and A. Stentz, "Using interpolation to improve path planning: The field d* algorithm," *Journal of Field Robotics*, vol. 23, no. 2, pp. 79–101, 2006. [Online]. Available: https://doi.org/10.1002/rob.20109

[29] A. Stentz, "The focussed d* algorithm for real-time replanning," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, p. 1652–1659. [Online]. Available: https://doi.org/10.5555/1643031.1643113

[30] S. Lin, A. Liu, J. Wang, and X. Kong, "A review of path-planning approaches for multiple mobile robots," *Machines*, vol. 10, no. 9, 2022. [Online]. Available: https://doi.org/10.3390/machines10090773

[31] Y. D. Setiawan, P. S. Pratama, S. K. Jeong, V. H. Duy, and S. B. Kim, "Experimental comparison of a* and d* lite path planning algorithms for differential drive automated guided vehicle," in *AETA 2013: Recent Advances in Electrical Engineering and Related Sciences*, I. Zelinka, V. H. Duy, and J. Cha, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 555–564. [Online]. Available: https://doi.org/10.1007/978-3-642-41968-3_55

[32] F. Gul, W. Rahiman, S. S. N. Alhady, A. Ali, I. Mir, and A. Jalil, "Meta-heuristic approach for solving multi-objective path planning for autonomous guided robot using pso–gwo optimization algorithm with evolutionary programming," *Journal of Ambient Intelligence and Humanize Computing*, vol. 12, p. 7873–7890, 2021. [Online]. Available: https://doi.org/10.1007/s12652-020-02514-w

[33] M. Nazarahari, E. Khanmirza, and S. Doostie, "Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm," *Expert Systems with Applications*, vol. 115, pp. 106–120, 2019. [Online]. Available: https://doi.org/10.1016/j.eswa.2018.08.008

[34] M. Kiadi, Q. Tan, and J. R. Villar, "Optimized path planning in reinforcement learning by backtracking," *Current Trends in Computer Sciences & Applications*, vol. 1, no. 4, pp. 80–90, 2019. [Online]. Available: https://doi.org/10.32474/CTCSA.2019.01.000116

[35] P. W. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, "Learning to navigate in complex environments," *ArXiv*, vol. abs/1611.03673, 2016. [Online]. Available: https://doi.org/10.48550/arXiv.1611.03673

[36] D. Wang, H. Deng, and Z. Pan, "Mrcdrl: Multi-robot coordination with deep reinforcement learning," *Neurocomputing*, vol. 406, pp. 68–76, 2020. [Online]. Available: https://doi.org/10.1016/j.neucom.2020.04.028

[37] Y. Mei, S. Li, C. Chen, and A. Han, "A multi-robot task allocation and path planning method for warehouse system," in *2021 40th Chinese Control Conference (CCC)*, 2021, pp. 1911–1916. [Online]. Available: https://doi.org/10.23919/CCC52363.2021.9549796

[38] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1–24, 2015. [Online]. Available: https://doi.org/10.1016/j.artint.2014.11.001

Anikó Kopacz is currently a PhD student at the Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Romania, under the supervision of Camelia Chira. Anikó is a member of the Metaheuristics for Complex Systems research group. Her current research interests include multi-agent systems, network analysis, machine learning, and reinforcement learning.

Enol García González received the BSc in Software Engineering by the University of Oviedo and MSc in Artificial Intelligence by the Technical University of Madrid. PhD in Computer Science by the University of Oviedo. My main research interests are metaheuristics and optimization algorithms, specially applied to real-world environments. Teaching at the University of Oviedo since 2021.

Camelia Chira is professor in computer science at the Faculty of Mathematics and Computer Science, Babeş-Bolyai University (Romania) and senior researcher in Artificial Intelligence within the Research Institute on Artificial Intelligence, Virtual Reality and Robotics. Current main research interests include evolutionary algorithms, nature-inspired computing, complex networks, machine learning, computer vision and bioinformatics. Her research work has generated important scientific contributions in the field of AI and its applications to complex search and optimization problems. She participated in several international research projects with results disseminated in more than 100 publications.

José R. Villar is an Industrial Engineer in Electronics and Control (Universidad de Oviedo). Ph. D. in Computer Science (University of León). Currently, he is an Full Professor with the Department of Computer Science at University of Oviedo. His research lines include i) AI applied to Bio-medical Engineering, ii) Metaheuristics and its applications to real world problems. Main figures in the last 5 years: i) JCR indexed: 22, ii) in international scientific conferences: 30, iii) research projects with public funding: 3, iv) I+D+I projects with private funding: 3. Dr. Villar is a member of the Advisory Committee for the Integrated Computer-Aided Engineering.