On the Scalable Multi-Objective Multi-Agent Pathfinding Problem

Jens Weise
Computational Intelligence Group
Otto von Guericke University
Magdeburg, Germany
0000-0002-8828-8752

Sebastian Mai
Computational Intelligence Group
Otto von Guericke University
Magdeburg, Germany
0000-0002-2255-3277

Heiner Zille
Computational Intelligence Group
Otto von Guericke University
Magdeburg, Germany
0000-0002-7262-9487

Sanaz Mostaghim
Computational Intelligence Group
Otto von Guericke University
Magdeburg, Germany
0000-0002-9917-5227

***Abstract*—The Multi-Agent Pathfinding problem (MAPF) has several applications in industry and robotics. The aim of a MAPF-solver is to find a set of optimal and non-overlapping paths for a number of agents in a navigation scenario. Existing approaches are shown to successfully deal with MAPF, where either the makespan or flow-time is used as a single objective. In this article, we treat the MAPF as a multi-objective optimisation problem (MOMAPF). In this paper, we consider three different objective functions, called makespan, flow-time and path-overlaps which are to be optimised at the same time. The MOMAPF problem in this paper is designed to be a scalable test problem for multi-objective optimisation algorithms, where we can scale up the variable space to reflect different real-world scenarios. We propose a new problem formulation for MOMAPF optimisation algorithms and implement it into the NSGA-II and NSGA-III and provide an experimental evaluation of the optimisation results.**

***Index Terms*—MAPF, Multi-Objective, Meta-Heuristics, Optimisation, Evolutionary Algorithms, MOMAPF, Multi-Agent, Pathfinding**

I. Introduction

Finding an optimal path for a mobile agent (robot) to move from a starting point to a specific position in a given search space is a well-researched problem. However, it is still a challenge to find multiple non-conflicting paths (trajectories) for multiple agents. Potential applications for this type of pathfinding problem currently are in automated warehouses and factories, where an increasing number of robots are deployed. A generalised version of this problem from real multi-robot systems is the Multi-Agent Path Finding problem (MAPF), a known optimisation problem with a very large solution space. The size of the search space can exponentially increase with the number of agents. In addition, an optimal path for an agent in such multi-agent scenarios can differ from the single-agent case, for instance, the optimal path can contain cycles. Typically, the quality of the solutions in MAPF is measured using two objectives: flow-time (sum of cost) and makespan (maximum cost) [1]. These two objectives are conflicting [2], and the existing literature usually deals with one of them as a single-objective problem.

In this article, we aim to study MAPF as a multi-objective problem and introduce a new scalable benchmark for multi-objective optimisation algorithms called Multi-Objective MAPF (MOMAPF). In addition to the above two objectives, we introduce the number of collisions as the third objective. So far, this objective has usually been studied as a constraint in the literature of pathfinding in robotics. It has been shown, however, that even solutions without conflicts often fail in robotic applications due the inaccuracies in the plan-execution and end up with numerous collisions [3].

In order to deal with a MOMAPF problem, we propose a new solution representation and the corresponding operators into the well known NSGA-II [4] and NSGA-III [5]. These algorithms are tested on variants of the MOMAPF problem with scalable number of agents and waypoints on three different environments. The environment influences the fitness landscape for the three objectives.

Our experiments show that MOMAPF can be taken as a scalable test problem for algorithms with integer encoding. This three-objective problem contains a scalable decision variable space varying between one decision variable for one agent and one waypoint to K agents and W waypoints resulting in $K \times W$ as the size of the decision space.

The remainder of this paper is structured as follows: Section II informs about existing approaches to the MAPF and multi-objective pathfinding. In Section III the MOMAPF problem and objectives are defined whereas in Section IV we show how multi-objective optimisation methods can be applied to the problem. This section also covers the solution encoding and the operators used by the algorithm to solve it. Section V shows the performed experimental evaluation and Section VI concludes this article.

This work was partially funded by the German Federal Ministry of Education and Research through the MOSAIK project (grant no. 01IS18070B).

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

II. State of the Art

This section elucidates existing approaches related to the proposed MOMAPF problem. As the MOMAPF is a combination of the MAPF as well as multi-objective problem formulations, we first go into detail on related works on multi-agent pathfinding, and afterwards take a look at existing multi-objective pathfinding problems.

A. Multi-Agent Pathfinding Problem

In recent years several applications emerged that call for solving different versions of the MAPF. One of the most prominent applications is in pathfinding for mobile robotic systems in automated warehouses [6]. A general description of the MAPF problem and several benchmarks have been published by Stern et al. [6]. Fellner et al. [7] summarised existing approaches on optimal solvers for the MAPF[1]. In addition to search-based solvers, the MAPF problem can be converted to a satisfiability (SAT) problem and be solved by a SAT-solver [1]. Most search-based solvers optimise the flow-time metric, while the solvers based on problem transformation optimise the makespan metric [1]. Some approaches assumed that selfish agents can cooperate to solve the MAPF problem, henceforth cooperative coevolution may be an interesting technique for other metaheuristic implementations [8].

There are many different variants of the MAPF with various environments, agent-actions, types of collisions, and the evaluation criteria [6], [7]. Oliveira et al. [9] have shown that the execution of a valid plan often fails due to inexact timing. This motivates our approach to include the number of collisions as an objective function measuring robustness, rather then treating collisions as a constraint to the problem.

B. Multi-Objective Pathfinding

The research area of multi-objective optimisation deals with the simultaneous optimisation of several conflicting objective functions. Multi-objective optimisation have been studied in the context of multi-agent systems in the past. For instance, the movements of multi-objective particle swarm optimisation have been adjusted to physical limitations and properties of robotic agents [10], [11]. Other work has considered the paths of multiple agents in warehousing or street environments as multiple objective functions of an optimisation problem in static or dynamic scenarios [12], [13]. However, there is not much work about the actual pathfinding as the goal of the optimisation, nor have they considered conflicts in the agents' paths.

In the related approaches on single-agent multi-objective pathfinding, usually a single path is evaluated by several different objective functions and multi-objective optimisation techniques are used to find an optimal path. In contrast to that, in our approach each path is only evaluated by its length and two additional objectives are represented through the interaction of multiple paths for multiple agents. The following related works describe approaches to multi-objective pathfinding with only a single-agent.

As Ahmed et al. stated, pathfinding for autonomous agents involves not only finding collision free paths (with respect to fixed obstacles) but also optimising several objectives [14]. They investigate pathfinding on a grid by using the well-known NSGA-II [4]. Castillo et al. compared different genetic algorithms to solve the multi-objective pathfinding problem [15]. The article in [16] also investigates on a genetic algorithm on the multi-objective pathfinding problem with large-scale grids by using a variable length chromosome approach. They compared their results with a standard A*-algorithm. Another example for multi-objective pathfinding is presented by Tozer et al. [17]. A method to find wire routes in a vehicle geometry is presented in [18], where the goal was to connect two electrical objects to each other. A well-known multi-objective genetic algorithm was used to find a path through a discretised 3-D space in a vehicle geometry to examine possible ways for wire connection. This problem also highly relates to the pipe routing problem [19].

When designing the algorithm it is of great value to find a suitable decision space representation. The authors in [20] proposed a graph-based representation of free space in an object. Yu et al. also used a graph-based approach when solving a pathfinding problem in 3-D space with fixed obstacles with an ACO algorithm [21].

III. MOMAPF Problem Formulation

This section describes the proposed Multi-Objective Multi-Agent Pathfinding (MOMAPF) problem. As described in the previous section, pathfinding has been treated with single agents with multiple objectives, and has been also treated as a multi-agent single-objective problem (see Section II). This paper presents a novel approach of combining these two concepts. This in inspired by several real-world problems, e.g. autonomous vehicle routing is one of the applications where every vehicle is an agent and a route can be evaluated by several objectives [6]. In this paper, we consider three objectives which are described in the following.

The MOMAPF problem is defined as follows: A given scenario is represented by a grid-map containing free space (where an agent can move) and obstacles (that can not be passed). In such a 2-dimensional map, a set of K agents is placed in different starting locations n_{i0}, each with its own goal n_{ig} for agent i. All agents start at time-step 0 and move on the map between grid-cells. Conflicts occur in case two agents travel through the same grid-cell simultaneously. The goal of the MOMAPF problem is to find an optimal set of paths $P^* = \{p_1, \cdots p_K\}$ for all agents with respect to multiple objectives. Each path $p_i = (n_{i0}, n_{i1}, \cdots, n_{ig})$ is the sequence of nodes n with an unknown length that the agent i visits until it reaches its goal. n_{i0} and n_{ig} indicate the starting and goal locations for agent i. Each node n refers to a grid cell of the environment.

The problem is formulated in a time-discrete fashion. All agents move in the same speed, so that every agent can

[1]The website http://mapf.info provides an overview about the current state of research on the MAPF problems and solutions.

perform one step of movement per time-step, i.e. they can move from one grid-cell of the map to another one in the 8-neighbourhood (i.e. the eight surrounding grid-cells). The map is represented as an undirected graph with nodes corresponding to each grid-coordinate and edges that connect adjacent cells in the map. We assume the distances between all neighbouring nodes is 1.

To solve this problem we propose three different objective functions: (a) Flow-time, (b) Makespan and (c) Overlaps.

a) Objective 1 - Flow-time: The flow-time is the average length of the agents' paths. It can be computed by summing up all the path lengths of all agents divided by the number of agents:

$$f_1(P) = \frac{\sum_{i=1}^{K} |p_i|}{K} \tag{1}$$

When optimising for flow-time, it is important to minimise detours and waiting time for all agents.

b) Objective 2 - Makespan: For every solution of the MOMAPF problem, the makespan is the time needed until the last task is finished.

$$f_2(P) = \max_{i=1}^{K} |p_i| \tag{2}$$

Hence, when minimising the makespan objective it is important to give priority to the agent with the longest path. As a result, it is likely that the path-length for other agents increases and therefore flow-time increases while makespan is optimised [2].

c) Objective 3 - Overlaps: Our third objective measures the number of overlaps between the paths of agents, indicating the number of collisions. We take edge-conflicts as well as node-conflicts into account. An edge-conflict occurs when more than one agent is using an edge from one cell to another in the same time-step. Node-conflicts occur when more than one agent uses the same cell during the same time-step, i.e. occupies the same node of the graph.

$$f_3(P) = \sum_t |\bigcap_i^K \{n_{it}\}| + \sum_t |\bigcap_i^K \{\{n_{it}, n_{it+1}\}\}| \tag{3}$$

In classical MAPF, node- and edge-conflicts in a solution are considered to be infeasible. However, as mentioned above, research has shown that even conflict-free paths may fail in real-world applications in case the timing during plan execution is inexact [9]. Allowing the optimisation algorithms to consider solutions which contain collisions can be beneficial to the overall search process and exploration of the search space.

IV. Multi-Objective Optimisation for the MOMAPF Problem

In this section, we propose a new problem formulation for MOMAPF that can be used with exisiting multi-objective optimisation algorithms. We introduce a waypoint-based, integer-valued encoding of solution candidates and define corresponding genetic operators. These are implemented into the well-known NSGA-II [4] and NSGA-III [5] algorithms.

A. Decision Space and Encoding of the Solution Candidates

The MOMAPF problem is encoded using a graph structure. All grid cells (see Fig. 1) are represented by a node in an undirected graph. Edges connect neighbouring cells, where in this encoding movement is possible to all eight neighbouring nodes. Cells that are occupied by obstacles in the map are not part of the graph, i.e. the nodes next to obstacles have less neighbouring nodes. For simplicity we assume that all agents need to move at each time step, i.e. waiting on a node is not a possible action.

In order to encode one single path p_i for agent i through the graph, we use a sequence of a fixed number of W waypoints $\{w_{i1}, \cdots w_{iW}\}$. From these waypoints the path is generated as already defined $p_i = (n_{i0}, \cdots, n_{ig})$. That means, the path of a single agent is encoded by a list of nodes, the first and the last entries in that list being the starting point and the goal of that agent. All waypoints in between these two are subject to change by the evolutionary process. During the evaluation, a path through the graph is built from the waypoints by connecting them with a simple pathfinding algorithm. More precisely, Dijkstra's algorithm [22] is used to connect the starting point to the first waypoint, the first waypoint to the second, and so forth until the last part of the path is computed as the shortest path from the last waypoint to the goal node. Note that in this implementation, each agent is not aware of the other agents paths or positions, i.e. the shortest paths is computed while disregarding any possible conflicts with other agents.

As one whole solution to the MOMAPF problem contains the paths of all existing agents, one individual in the evolutionary algorithm is a collection of the K lists of waypoints, one for each agent in the problem. As mentioned, the first and the last waypoints are always fixed for each agent, hence we allow the genetic operators which are described below to perform only on the *inner* waypoints.

The number of inner waypoints in each agent's path list is fixed, but it can be seen that the more waypoints are used, the finer the adjustments are possible in order to navigate around obstacles and avoid conflicts. On the other hand, more waypoints also increase the size of the search space, and finding an optimal (short) path might be more difficult since all waypoints may initially be far away from each other.

B. Crossover

The crossover operator is designed to exchange two waypoints of two randomly selected agents with each other. Within the paths of those agents, a random inner waypoint is chosen respectively and these waypoints are exchanged between the solutions.

C. Mutation

Our implemented mutation operator chooses randomly one agent from a solution and one random inner waypoint from its path. This waypoint is then shifted in one of the eight possible neighbourhood directions. Due to our graph-based encoding of the decision space we ensure that this shift will

not move the waypoint into or through a wall / an obstacle. Fig. 1 illustrates an example. The second waypoint (blue), can be shifted according to the shown eight directions in blue.

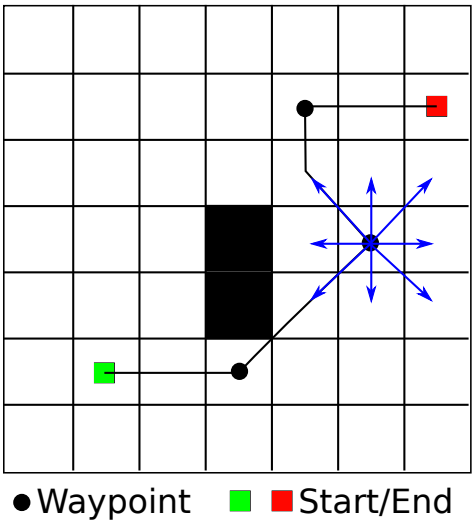


Fig. 1: Path of one agent. The first and last waypoints of the path are fixed, while the three inner waypoints are subject to change. The waypoints are connected with Dijkstra's algorithm. Black tiles show obstacles in the map. Possible mutations of the path are shown in blue arrows on the second waypoint.

D. Implementation

For the implementation of the proposed algorithm we use the *JGraphT* library for graph storage [23]. For optimisation, we use the well-known *jMetal* multi-objective optimisation framework, version 5.7, and the pre-implemented versions of NSGA-II [4] and NSGA-III [5] in order to extend their functionality to fit our encoding [24], [25].

To increase the algorithm's performance we decided to store calculated shortest paths in an efficient in-memory hash map, i.e. *Chronicle Map*[2]. Thus we assured that each two waypoints are always connected by the same shortest path, even in case several shortest paths exist.

V. Experiments

In this section we aim to evaluate our proposed approach on the MOMAPF problem using the encoding presented in the previous section.

A. Configuration and Parameter Settings

To show the performance of the optimisation algorithms in different settings of the problem, we use scenarios and maps from the Multi-Agent Pathfinding Benchmark [3] [6]. We limit the experiments to three different maps: An empty 32x32 map, the *room* map of size 64x64 and the *maze* map of size 128x128. The two latter, non-empty maps are shown in Fig. 2. The black areas are obstacles that can not be passed, while the grey cells are free. The benchmarks obtained from MAPF.info also provide several scenarios for each map, classified as even and random scenarios. With the assumption that randomly distributed agents are more likely to represent real-world applications like warehouses or traffic navigation, we select the scenarios from the random class for each map, where the start and end positions are randomly distributed. These random scenarios are used with 10 and 50 agents in each map respectively.

In addition to different map sizes and different numbers of agents, the size of a problem instance can further be adjusted by the amount of waypoints for each agent. A smaller number of inner waypoints reduces the size of the decision space for the algorithm, but allows less detailed adjustments in terms of navigating the agent around obstacles or to avoid conflicts with other agents. To account for different problem sizes and complexities, we use two different configurations with 4 and 10 waypoints, resulting in 2 and 8 inner waypoints respectively.

In total, our experiments contain three maps, two agent configurations (10 and 50) and two waypoint configurations (4 and 10), which results in a set of 12 unique problem instances. Two different algorithms are used to optimise the 3-objective problem, NSGA-II and NSGA-III. Each of those algorithms is used to optimise the 12 problem instances for 31 independent runs. The population sizes are set to 92 (due to the internal NSGA-III implementation of the jMetal framework). The number of generations per algorithm is set to 100, which results in a total of $9,200$ function evaluations. Due to the novelty of this approach of solving the MAPF problem in a multi-objective manner with a meta-heuristic algorithm, the true Pareto-fronts and corresponding Pareto-sets of the problem instances are unknown. To compare the performance of the two algorithms with each other, we therefore apply the normalised Hypervolume (HV) indicator [26], using the nadir point of the union of all obtained runs per problem instance as the reference point. The results of NSGA-II and NSGA-III are compared and tested for statistical significance using the two-sided pairwise Mann-Whitney-U Test, with the null hypothesis that the distributions of the two samples have equal medians. Statistical significance of the differences between the performance is assumed for a p-value smaller than 0.01.

B. Results and Analysis

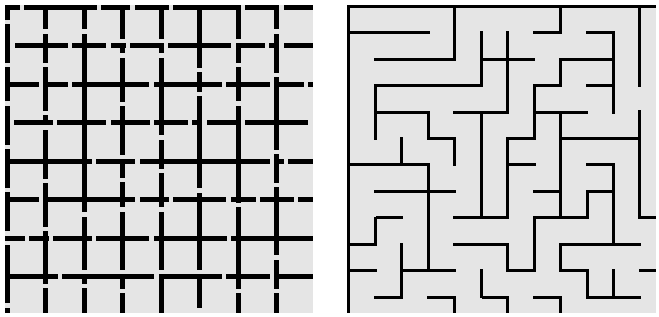
The median HV and interquartile range (IQR) values obtained by NSGA-II and NSGA-III are shown in Table I. Both NSGA-II and NSGA-III show similar performance with no significant differences in the HV indicator. The difference between both algorithms lies only in the concept for obtaining and maintaining diversity in the objective space. The non-significant differences in the results indicate that the complexity of the trade-off between the three objectives functions does not affect algorithm performance, i.e. diversity can be achieved by traditional diversity-preserving methods and the problem does not require special many-objective techniques.

[2]Chronicle-Map on GitHub https://github.com/OpenHFT/Chronicle-Map
[3]The benchmark maps can easily be obtained from www.mapf.info

TABLE I: Obtained median and IQR values of the Hypervolume indicator for the NSGA-II and NSGA-III algorithms. The results of both algorithms show no statistically significant differences to each other ($p < 0.01$).

Map Type	Map size	# Agents	# Waypoints	NSGA-II	NSGA-III
empty	32 x 32	10	4	0.59789 (1.26E-2)	0.59540 (9.29E-3)
empty	32 x 32	10	10	0.56771 (1.63E-2)	0.56609 (1.77E-2)
empty	32 x 32	50	4	0.51422 (1.63E-2)	0.51493 (1.37E-2)
empty	32 x 32	50	10	0.35800 (9.92E-3)	0.35790 (1.09E-2)
room	64 x 64	10	4	0.64414 (1.96E-2)	0.64108 (1.61E-2)
room	64 x 64	10	10	0.59322 (1.76E-2)	0.59381 (1.55E-2)
room	64 x 64	50	4	0.52532 (1.86E-2)	0.52233 (1.36E-2)
room	64 x 64	50	10	0.43734 (1.98E-2)	0.43752 (1.29E-2)
maze	128 x 128	10	4	0.67837 (2.02E-2)	0.67453 (1.83E-2)
maze	128 x 128	10	10	0.58794 (2.23E-2)	0.58468 (2.14E-2)
maze	128 x 128	50	4	0.47958 (1.04E-2)	0.47634 (2.05E-2)
maze	128 x 128	50	10	0.43796 (1.28E-2)	0.44031 (6.12E-3)

In the following, we perform a more in-depth analysis of the final populations of the NSGA-III algorithm.



(a) Room Map 64x64 (b) Maze Map 128x128

Fig. 2: Maps with obstacles [6]

Figure 3 shows pairwise 2D-projections of the objective values of the final populations. The figure shows all solutions from 31 runs of the 10 agent problem instance on the empty 32x32 map with 4 and 10 waypoints per agent. The dominated and non-dominated individuals of each problem instance are highlighted in different colours respectively. The objective values of the non-dominated individuals show that all three objectives are conflicting, as multiple solutions exist that represent different trade-offs between the objectives. In addition, the existence of solutions without any overlaps between paths show the applicability of the algorithm to find feasible solutions to the classical MAPF problem. We can further observe a trade-off between overlaps and makespan as well as overlaps and flow-time.

The projections also reveal that overall, using only four waypoints results in a much better performance of the solutions in all three objective functions. The authors presume that the reduced search-space helps to achieve convergence. While reducing the search-space might exclude optimal solutions, a smaller search space can on the other hand be explored and exploited further with the computational budget at hand. This shows that with the relatively low number of function evaluations, our approach is more suited to scenarios with only few conflicts. Another possible reason for the better performance in the 4-waypoint instances may be the proposed crossover operator, as only switching two waypoints between two solutions constitutes a larger amount of change (relative to the size of the individual). This reveals potential to discover better solutions in the future by designing more advanced operators.

Fig. 6 depicts a projection in the map's x-y-plane showing the resulting paths of one specific non-dominated solution on map *room* with the configuration of ten agents and four waypoints. The agents often take huge detours through the map, which is most likely not a satisfactory solution since there is room for improvement, i.e. aligning the waypoints.

Similar conclusions can be drawn from analysing the non-dominated solutions on the *room* map (Fig. 4). We used only four waypoints and want to compare two experiments with ten and 50 agents. With 50 agents the algorithm is not able to find conflict-free solutions any more, while it is able to find non-overlapping solution with ten agents. However, it is an open question whether the algorithm was not able to find conflict-free solutions with the given computational budget, or whether it is not possible to represent conflict-free solutions with only four waypoints when considering 50 agents.

In all three maps we can clearly observe a trade-off between the three different objective functions. On the *maze* map, which is the largest and most complex of the three, we now take a closer look at the obtained non-dominated solutions of the 4-waypoint instances. Fig. 5 shows the corresponding 2-D projections of the combined solutions found in our 31 independent runs. We can observe here that in the solution set there are two non-dominated solutions which contain overlaps that are in exchange superior in the other two objectives. This indicates that in real applications it might be beneficial to accept small amounts of conflicts to be resolved outside of the MOMPAF algorithm (e.g. by local-planning during the runtime of the system). In exchange, solutions with superior flow-time and makespan can be found in our multi-objective problem formulation, that may not be seen by the decision maker if zero overlaps were used as a hard constraint. The shape of the other non-dominated front in the solutions in the makespan/flow-time-plot shows that our algorithm found a Pareto-front with many different trade-offs between the makespan and flow-time objectives.

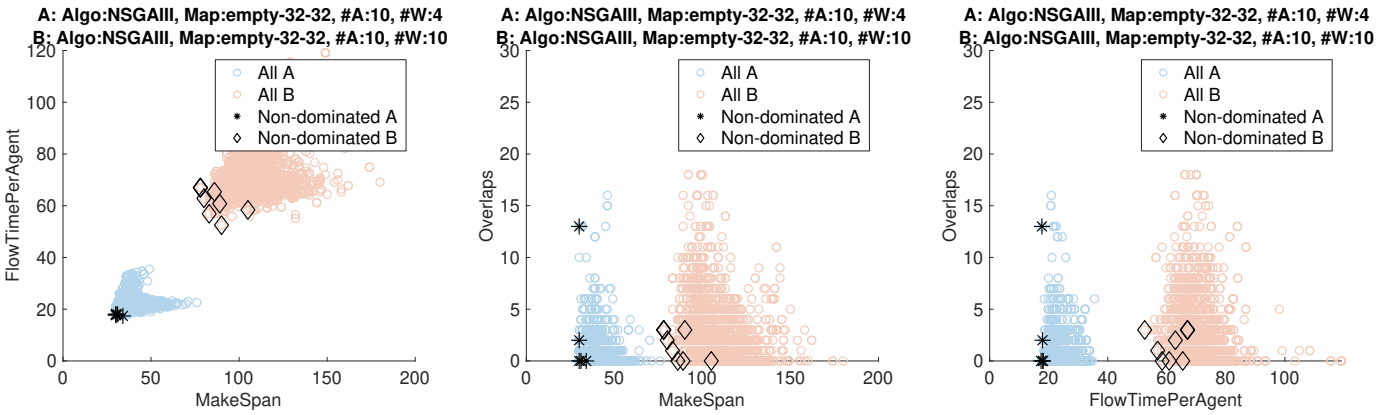


Fig. 3: Matrix-Scatter-Plot of the NSGA-III Experiment on Map *empty*, 10 Agents, 4 vs 10 Waypoints

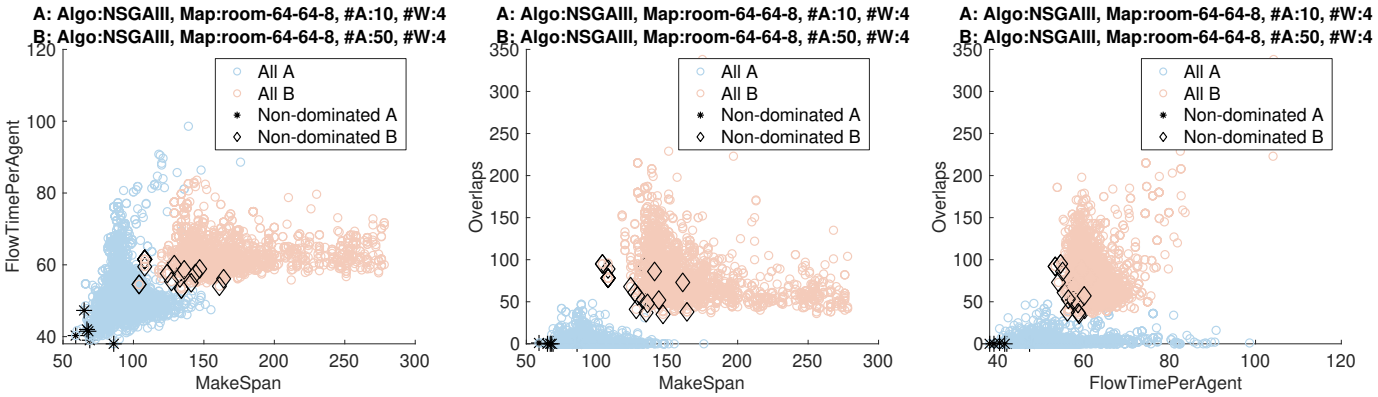


Fig. 4: Matrix-Scatter-Plot of the NSGA-III Experiment on Map *room*, 10 vs 50 Agents, 10 Waypoints

VI. Conclusion and Outlook

In this paper we present the Multi-Objective Multi-Agent Pathfinding problem (MOMAPF), as well as a simple algorithm that is capable of solving the problem for certain configurations, by using a list of agents and their waypoints as solution representation. The novelty of our approach lies in (1) solving the MAPF with a meta-heuristic algorithm and (2) treating the MAPF as a multi-objective problem. In our experiments the used algorithms perform with promising results, however the representation of the solutions we have chosen reduces the solution space to a fixed-size, integer-valued problem and excludes a huge number of valid (and possibly optimal) solutions. The reduction of the search space is critical, as the algorithm ceases to converge in case we change the number of waypoints from four to ten. The most evident application of our approach is in Multi-Robot Systems, however the (MO)MAPF is related to several other routing problems like the pipe-routing problem or other routing problems involving multiple paths (like finding the path to build multi-lane roads) [19].

This paper is only a first investigation on the topic of MOMAPF and we used a very elementary meta-heuristic approach to solve the problem. Based on the proposed benchmark in this article, multiple future research directions can be identified.

First, we aim to investigate into the MOMAPF problem as a more intrinsic problem, e.g. that every agent has a set of objectives. Furthermore, we assumed that agents are not able to wait but have to move to another position at every time step. An area of future research is to extend our approach and enable agents to wait at a node. One way to do so is to expand the graph in time, i.e. have a layer for every time step and edges going only forward in time. Using a time-expanded graph would have exceeded our computational budget, thus we did not implement it in the experiments in this article. An approach to simplify a large time-expanded graph is to insert loops on each node representing waiting when traversing this edge. This, however, requires a different mechanism to connecting the waypoints, than the simple path finding algorithm in our implementation which is not capable of appropriate loop handling. Furthermore, we believe that other genetic operators could help the algorithm to converge faster and discover more promising solution candidates.

We also plan to perform MOMAPF with real robots. In a robotics application a feasible metric for robustness needs to be chosen instead of our current conflict-based objective [3]. We believe that an improved robustness objective will also lead to an improved fitness landscape, as perturbing a robust solution is likely to lead to another feasible solution, while

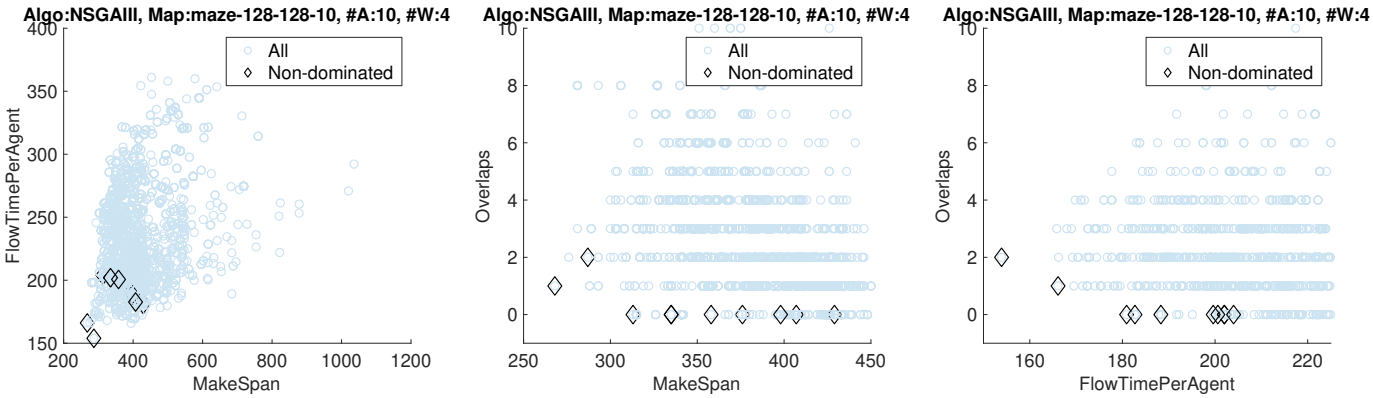


Fig. 5: Matrix-Scatter-Plot of the NSGA-III Experiment on Map *maze*, 10 Agents, 10 Waypoints

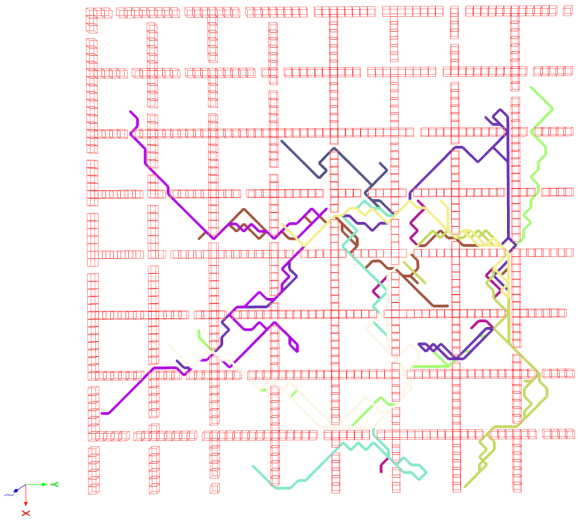


Fig. 6: Projected result paths of a non-dominated solution using the NSGA-III algorithm on the *room* map, 10 Agents, 4 Waypoints

even a small perturbation may lead to a severe change in the current metric. Such deviations are likely to occur frequently, since robots can never perfectly execute the planned movement. Furthermore, robot movement is always restricted by kinematic constraints and robots move in continuous space and need to deal with localisation errors [27]. Hence, the resulting movement is much more complex than the simple grid based movement model in our current article. An approach to make solutions more feasible for the actual use with robots is to implement a certain collision radius. If an agent enters another agent's radius it can be considered as a collision. An algorithm's solution can be perturbed up to a certain extend while remaining feasible. We propose that a more realistic benchmark could provide fitness values based on a path using a vehicle model (like the Dubins model [28]) and a robustness measure based on the timing of overlapping paths. Finally, a decision making strategy that selects one of the non-dominated solutions also needs to be developed. The concept of multi-modal optimisation [29]–[31] might support here in order to find several solutions with the same objective values but different configurations, i.e. agent paths. Decision makers might use this information to determine the best alternative.

Finally, we hope to see improved algorithms that can solve the MOMAPF problem in a more advanced fashion and can deal with a larger number of agents (for instance by applying specific large-scale optimisation methods [32]–[36]) as well as with a dynamic number of waypoints.

References

[1] P. Surynek, A. Felner, R. Stern, and E. Boyarski, "An empirical comparison of the hardness of multi-agent path finding under the makespan and the sum of costs objectives," *Proceedings of the 9th Annual Symposium on Combinatorial Search, SoCS 2016*, vol. 2016-Janua, no. SoCS, pp. 145–146, 2016.

[2] M. Bhuvaneswari, Ed., *Application of Evolutionary Algorithms for Multi-objective Optimization in VLSI and Embedded Systems*. New Delhi: Springer India, 2015. [Online]. Available: http://link.springer.com/10.1007/978-81-322-1958-3

[3] R. Barták, J. Švancara, V. Škopková, D. Nohejl, and I. Krasičenko, "Multi-agent path finding on real robots," *AI Communications*, vol. 32, no. 3, pp. 175–189, 2019.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[5] K. Deb and H. Jain, "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*, vol. 18, no. 4, pp. 577–601, 8 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6600851/

[6] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. S. Kumar, E. Boyarski, and R. Bartak, "Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks," *ArXiv Preprint*, 6 2019. [Online]. Available: https://arxiv.org/abs/1906.08291http://arxiv.org/abs/1906.08291

[7] A. Felner, R. Stern, S. E. Shimony, E. Boyarski, M. Goldenberg, G. Sharon, N. Sturtevant, G. Wagner, and P. Surynek, "Search-Based Optimal Solvers for the Multi-Agent Pathfinding Problem: Summary and Challenges," *Tenth Annual Symposium on Combinatorial Search*, no. SoCS, pp. 29–37, 2017. [Online]. Available: https://www.aaai.org/ocs/index.php/SOCS/SOCS17/paper/view/15781

[8] Z. Bnaya, R. Stern, A. Felner, R. Zivan, and S. Okamoto, "Multi-agent path finding for self interested agents," *Proceedings of the 6th Annual Symposium on Combinatorial Search, SoCS 2013*, pp. 38–46, 2013.

[9] G. M. B. Oliveira, R. G. O. Silva, G. B. S. Ferreira, M. S. Couceiro, L. R. do Amaral, P. A. Vargas, and L. G. A. Martins, "A Cellular Automata-Based Path-Planning for a Cooperative and Decentralized Team of

Robots," in *IEEE congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 739–746.

[10] S. Mai, H. Zille, C. Steup, and S. Mostaghim, "Multi-objective collective search and movement-based metrics in swarm robotics," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 387–388. [Online]. Available: https://doi.org/10.1145/3319619.3321967

[11] ——, "Online optimization of movement cost for robotic applications of pso," in *Progress in Artificial Intelligence*, P. Moura Oliveira, P. Novais, and L. P. Reis, Eds. Cham: Springer International Publishing, 2019, pp. 307–318.

[12] S. Mostaghim, C. Steup, and H. Zille, "Multi-objective distance minimization problems – applications in technical systems," *at-Automatisierungstechnik*, vol. 66, no. 11, pp. 964–974, 2018.

[13] H. Zille, A. Kottenhahn, and S. Mostaghim, "Dynamic distance minimization problems for dynamic multi-objective optimization," in *IEEE Congress on Evolutionary Computation (CEC)*, June 2017, pp. 952–959.

[14] F. Ahmed and K. Deb, "Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms," *Soft Computing*, vol. 17, no. 7, pp. 1283–1299, 7 2013. [Online]. Available: http://dx.doi.org/10.1007/s00500-012-0964-8http://link.springer.com/10.1007/s00500-012-0964-8

[15] O. Castillo, L. Trujillo, and P. Melin, "Multiple objective genetic algorithms for path-planning optimization in autonomous mobile robots," *Soft Computing*, vol. 11, no. 3, pp. 269–279, 2007.

[16] M. Alajlan, A. Koubaa, I. Chaari, H. Bennaceur, and A. Ammar, "Global path planning for mobile robots in large-scale grid environments using genetic algorithms," in *2013 International Conference on Individual and Collective Behaviors in Robotics (ICBR)*, no. 1. IEEE, 12 2013, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/document/6729271/

[17] B. Tozer, T. Mazzuchi, and S. Sarkani, "Many-objective stochastic path finding using reinforcement learning," *Expert Systems with Applications*, vol. 72, pp. 371–382, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2016.10.045

[18] J. Weise, S. Benkhardt, and S. Mostaghim, "Graph-based multi-objective generation of customised wiring harnesses," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '19*. New York, New York, USA: ACM Press, 2019, pp. 407–408. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3319619.3321908

[19] G. Belov, L. Cohen, M. G. de la Banda, D. Harabor, S. Koenig, and X. Wei, "Position Paper: From Multi-Agent Pathfinding to Pipe Routing," no. May, 5 2019. [Online]. Available: http://arxiv.org/abs/1905.08412

[20] J. Weise, S. Benkhardt, and S. Mostaghim, "A Survey on Graph-based Systems in Manufacturing Processes," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 11 2018, pp. 112–119. [Online]. Available: https://ieeexplore.ieee.org/document/8628683/

[21] X. Yu, W.-N. Chen, T. Gu, H. Yuan, H. Zhang, and J. Zhang, "ACO-A*: Ant Colony Optimization Plus A* for 3-D Traveling in Environments With Dense Obstacles," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 4, pp. 617–631, 8 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8510897/

[22] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 12 1959. [Online]. Available: http://link.springer.com/10.1007/BF01386390

[23] D. Michail, J. Kinable, B. Naveh, and J. V. Sichi, "JGraphT – A Java library for graph data structures and algorithms," 4 2019. [Online]. Available: http://arxiv.org/abs/1904.08355

[24] J. J. Durillo and A. J. Nebro, "JMetal: A Java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, no. 10, pp. 760–771, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.advengsoft.2011.05.014

[25] A. J. Nebro, J. J. Durillo, and M. Vergne, "Redesigning the jMetal multi-objective optimization framework," *GECCO 2015 - Companion Publication of the 2015 Genetic and Evolutionary Computation Conference*, pp. 1093–1100, 2015.

[26] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[27] S. Mai, C. Steup, and S. Mostaghim, "Simultaneous Localisation and Optimisation for Swarm Robotics," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, Bangalore, India, 2018, pp. 1998–2004. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8628767

[28] L. E. Dubins, "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[29] M. Javadi, H. Zille, and S. Mostaghim, "Modified crowding distance and mutation for multimodal multi-objective optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '19*. New York, New York, USA: ACM Press, 2019, pp. 211–212. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3319619.3321970

[30] ——, "The effects of crowding distance and mutation in multimodal and multi-objective optimization problems," in *Springer ECCOMAS book series on Computational Methods in Applied Sciences*. Guimarães, Portugal. In Press: ACM, 2020.

[31] M. Javadi, C. Ramirez-Atencia, and S. Mostaghim, "Combining manhattan and crowding distances in decision space for multimodal multi-objective optimization problems," in *Springer ECCOMAS book series on Computational Methods in Applied Sciences*. Guimarães, Portugal. In Press: ACM, 2020.

[32] H. Zille, "Large-scale multi-objective optimisation: new approaches and a classification of the state-of-the-art," Ph.D. dissertation, Otto-von-Guericke-Universitaet Magdeburg, Fakultaet für Informatik, 2019.

[33] H. Zille, H. Ishibuchi, S. Mostaghim, and Y. Nojima, "A framework for large-scale multi-objective optimization based on problem transformation," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 260–275, April 2018.

[34] H. Zille and S. Mostaghim, "Comparison study of large-scale optimisation techniques on the LSMOP benchmark functions," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, November 2017, pp. 1–8.

[35] H. Zille, H. Ishibuchi, S. Mostaghim, and Y. Nojima, "Mutation operators based on variable grouping for multi-objective large-scale optimization," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, Dec 2016, pp. 1–8.

[36] H. Zille and S. Mostaghim, "Linear search mechanism for multi- and many-objective optimisation," in *Evolutionary Multi-Criterion Optimization*, K. Deb, E. Goodman, C. A. Coello Coello, K. Klamroth, K. Miettinen, S. Mostaghim, and P. Reed, Eds. Cham: Springer International Publishing, 2019, pp. 399–410.