High-Level Mission Planning for Multi-Agent Indoor System

1st Rostislav Karásek
Institute of Flight Guidance
German Aerospace Center
38108 Braunschweig, Germany
Rostislav.Karasek@dlr.de

2nd Christian Kallies
Institute of Flight Guidance
German Aerospace Center
38108 Braunschweig, Germany
Christian.Kallies@dlr.de

***Abstract*—The manuscript presents a high-level mission planning for multi-agent indoor systems. The high-level mission planning separates the mission goals between the agents, plans the order of the mission goals, and provides corridors serving as constraints for a real-time controller of the multi-agent system in which the real-time controller searches for optimal paths while resolving conflicts between the agents. The proposed algorithm uses a highly optimized tree data structure to represent a 3D indoor environment. Then the set of adjacent tree nodes defines the shortest possible corridor to fulfill the mission goals while avoiding obstacles in the indoor environment. Planning the mission goals order and assignment to agents is an NP-hard problem that we solve using heuristic algorithms to find a viable solution before the mission starts. This work implements a multi-objective optimization algorithm combining a genetic algorithm and simulated annealing to find a viable solution for the mission as a composition of the unobstructed corridors between the individual mission goals found by the A* path planning algorithm. The evaluation of the proposed high-level mission planning in a typical indoor environment finds a viable solution in time, even for a large number of mission goals. Also, the behavior of the multi-agent system is easily altered to prefer solutions minimizing the total traveled distance or distributing the workload evenly between the agents based on the mission character.**

***Index Terms*—A* algorithm, Christofides Algorithm, Genetic Algorithm, Multi-Agent System, Multiple Traveling Salesman Problem, Path Planning, Simulated Annealing, Unmanned Aerial System**

I. Introduction

Controlling the multi-agent system indoors while avoiding obstacles and other agents is challenging. Moreover, if the multi-agent system aims to fulfill a mission, e.g., finding sources of dangerous gas leakage, the task becomes quickly impossible in real-time. That is especially true when the multi-agent system is assigned many waypoints where the gas concentration measurements should provide the best information for localizing gas leakage sources [1]. An approach to decrease the calculational complexity is to separate the task into high-level mission planning that assigns the waypoints to agents in proper order and provides 3D corridors for the individual agents. Then the low-level control algorithm provides control signals to guide agents through corridors toward waypoints while avoiding collisions with other agents. The collision with the environment is avoided by keeping the agents inside the corridors.

The task of the multi-agent system is to split the assigned waypoints between individual agents and plan in which order to visit the assigned waypoints. We aim that the total traveled distance of all agents is minimized and simultaneously, the overall mission time is kept short. We can formally describe the problem as follows:

- Each waypoint has to be assigned to an agent exactly once. Return to starting position is not assumed since a new waypoint may emerge.
- Each agent has to visit all assigned waypoints in an order that minimizes its traveled distance.
- The waypoints have to be assigned such that the objective function is minimized.

This problem definition is similar to the Multiple Traveling Salesman Problem (MTSP) with multiple depots [2]. However, the MTSP is defined as a single objective optimization problem that aims to minimize the total traveled distance.

The exact search for an optimal solution is an NP-hard problem with exponential complexity that is prohibitively slow even for medium size problems of tens of waypoints. On the other hand, the heuristic algorithms aim to provide a solution using a polynomial complexity algorithm, but the solution can be suboptimal. Notable heuristics approaches are genetic algorithm [3] and simulated annealing [4]. Refer to [2], [5] for a detailed overlook of the possible algorithms.

The genetic algorithm is a heuristic algorithm that consists of a population of possible solution candidates. The best-performing candidates are selected for the crossover. After the crossover, some randomly selected candidates are randomly changed (mutated). The set of new candidates yields a population for the next generation.

The walls separating rooms and many small to medium size objects with possibly complicated shapes are obstacles between which the multi-agent system must navigate without collision. The environment has to be represented by a convenient data structure for the multi-agent system to use for navigation. The key feature of the environment representation is to model the actual environment precisely and efficiently. If the representation precision is high, the representation is more complicated, slows navigation, and consumes too much memory. If the representation is coarse, larger space than necessary might represent obstacles, leading to longer paths or even rendering some areas unreachable.

A natural approach to environment representation is to divide it into cells small enough that each cell is assumed fully occupied or free. A naive query for a random cell in a memory yields a linear complexity in the number of cells. However, storing the cells as nodes in a tree structure decreases the query complexity to logarithmic in the number of nodes. The tree structure remains vital for creating and updating the environment representation, e.g., from a point cloud [6]. Arbitrary node address can be obtained using the spacial localization code based on Morton's code [7]. The vital property for fast navigation is quickly finding a specific node in the memory.

Path planning is an integral part of the navigation process that aims to find the shortest collision-free path between the agent's location and the goal location. There are many approaches to path planning. One possible dichotomy is stochastic and deterministic methods.

A typical stochastic path planning method is Rapidly-exploring Random Trees (RRT) [8], which grows a tree based on the kinetic model of an agent until the goal state is reached. Another approach to stochastic path planning is a Probabilistic RoadMap (PRM) [9], which uses a local planner to connect random agent configurations in a free space. The path is obtained using a graph-based minimum distance algorithm when the space is connected enough.

The deterministic methods represent the free space as a graph and use graph-search algorithms to obtain the desired path. Most graph-based algorithms are based on Dijkstra's minimum distance algorithm [10]. Adding an appropriate heuristic function to distance measure can speed up Dijkstra's algorithm as in A* algorithm [11] or improve the performance in unknown terrain when replanning is required as in D* Lite [12].

The presented method does not aim to follow the standard approach of searching for a complete collision-free path that the agent should follow as closely as possible. We aim for high-level mission planning that constrains the free space between the agents and goals, providing well-defined constraints for a path planning algorithm that considers the dynamics of the agents and, therefore, provides more detailed and physically feasible paths. A possible algorithm based on Model Predictive Control (MPC) [13] has been established in [14]–[16]. It uses a linearized kinematic model of the agents together with a set of constraints representing obstacles or possibly the free space to optimize controls for the whole multi-agent system while covering a set of given waypoints. The calculational load of MPC can be prohibitive, especially for multi-agent systems where constraints encapsulate all obstacles in the environment. Our solution aims to provide corridors defined as a series of free nodes that must be transversed by an agent to achieve the goal. This aim is central to the proposed high-level mission planning.

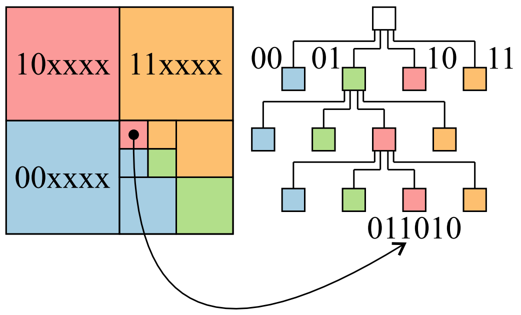

Fig. 1. Visualization of a single obstacle (black dot) in a 2D environment by a quadtree of depth three. The color code visualizes which quadrant of a node corresponds to a tree node at each level of depth.

II. High-Level Mission Planning

A. Environment Characterization

We represent the d-dimensional environment by a tree structure where each node has 2^d equisized children nodes [17]. The tree structure allows the clustering of areas of identical properties using variable depth of tree branches. Therefore, it is unnecessary to transverse the tree from the root to the leaf of the maximum specified depth but stop at the node with all children with the same properties. The variable branch depth saves memory and speeds up the query. We store only nodes representing occupied space, further decreasing memory consumption.

Fig. 1 shows an example of a 2D environment representation where the black dot visualizes a single obstacle with the corresponding location code 011010. The corresponding quadtree on the left shows that only branches with at least one obstacle are stored in memory. There is one bit of location code per dimension per tree depth. Hence, the 2D environment of depth three requires a location code 2x3 bits long to address all environment nodes.

This work assumes 3D environment representation. In 3D, the environment is split into eight octants and each octant into another eight child octants until the maximum tree depth is approached. Hence, we refer to the used tree structure as an octree.

B. Collision-Free Path Planning

Since our environmental representation is already a graph, i.e., octree, we use the A* algorithm [11] to find the shortest path through the graph. The A* algorithm works as follows:

1) The start node is inserted into the priority queue implemented as a binary heap [18], where the priority queue key is the cost to get to the start node, which is zero, plus the heuristics cost. The distance from the current node to the goal node is the heuristics cost used in the A* algorithm.
2) A node with the lowest key is popped from the priority queue and used as the current node. Each popped node is marked as finished.

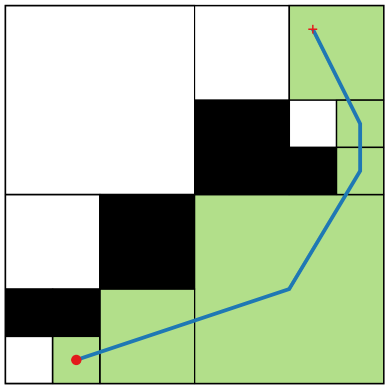
Fig. 2. Example path (blue line) in a 2D environment; start point (red dot), end point (red cross), occupied area (black boxes), safe and shorted corridor obtained by the A* algorithm (green boxes)

3) All nodes adjacent to the current node not yet marked as finished are inserted into the priority queue with the priority queue key corresponding to the cost to transverse from the previous node to the current node plus the cost to get to the previous node from the start node, plus the heuristics cost of the current node.
4) Steps 2 and 3 are repeated until the popped node is the goal node.

The path is recovered by following the pointers to previous nodes starting at the current (goal) node. The total cost to get to the goal from the start node is the cost of the goal node.

The ancestors closest to the tree root, which are entirely free of obstacles, serve as graph nodes for the A* algorithm. This significantly decreases the number of nodes to visit before finding the shortest path since we do not need to use only the leaves of the octree.

The A* graph grows as it visits neighboring nodes and uses nodes not marked as occupied, allowing only collision-free paths. Since the A* does not necessarily visit all the free-space nodes, building the whole free-space graph is unnecessary.

We use an efficient approach to find the current node's neighbors based on [19]. If the neighbor is free, the algorithm checks the ancestors progressively until it founds the largest free ancestor. On the other hand, if the neighbor is partly occupied, the neighbor's children, who also neighbor with the current node, are checked until all free smaller-size neighbors are found, if any. All largest free neighbors not yet marked as finished in A* are added to the A* graph and inserted into the priority queue.

Fig. 2 shows an example environment in 2D where occupied nodes are black. The path planning algorithm finds an obstacle-free path between the star position marked by a red circle and the end position marked by a red cross. The shortest obstacle-free path is shown by a blue line connecting the centers of the free nodes. The sum of distances connecting the neighboring nodes serves as the distance for the path planning algorithm. The path planning connects the previous node's center and the waypoint position or two waypoints directly if located in the same node. Fig. 2 shows this for the end position in the upper right corner. The output of the path-planning algorithm is the environment node series (corridor) marked green, serving as constraints for the controller, e.g., MPC.

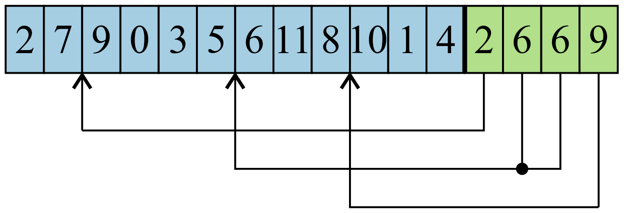

Fig. 3. The two-part chromosome visualization for solving MTSP using genetic algorithm.

C. Mission Assignment and Path Planning Coupling

Minimizing the total traveled distance while keeping the traveled distance of all agents similar leads to a multi-objective optimization problem, where simultaneously minimizing all objectives might not be possible. The formal definition of the optimization problem is

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \epsilon \boldsymbol{X}} f\left(\boldsymbol{x}\right), \tag{1}$$

with the multi-objective function

$$f\left(\boldsymbol{x}\right) = \alpha f_{\mathrm{d}}\left(\boldsymbol{x}\right) + \left(1-\alpha\right) f_{\sigma}\left(\boldsymbol{x}\right), \tag{2}$$

where $\hat{\boldsymbol{x}}$ is the optimal solution of the multi-objective optimization problem. We define two objective functions as

$$f_{\mathrm{d}}\left(\boldsymbol{x}\right) = \sum_{n=0}^{N-1} d\left(\boldsymbol{v}_{\mathrm{a},n}, \boldsymbol{V}_{\mathrm{w}}, \boldsymbol{x}\right), \tag{3}$$

$$f_{\sigma}\left(\boldsymbol{x}\right) = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}\left(d\left(\boldsymbol{v}_{\mathrm{a},n}, \boldsymbol{V}_{\mathrm{w}}, \boldsymbol{x}\right) - \frac{1}{N} f_{\mathrm{d}}\left(\boldsymbol{x}\right)\right)^2}, \tag{4}$$

where the distance function $d\left(\boldsymbol{v}_{\mathrm{a},n}, \boldsymbol{V}_{\mathrm{w}}, \boldsymbol{x}\right)$ returns the sub-route distance of the n-th agent, starting at position $\boldsymbol{v}_{\mathrm{a},n}$, given the whole set of waypoint positions $\boldsymbol{V}_{\mathrm{w}}$ and the chromosome vector $\boldsymbol{x}$.

The weighting of the objective functions (3) and (4) is given by $\alpha \,\epsilon\, (0, 1)$ that can be selected based on the desired behavior of the multi-agent system. Selecting α close to one prioritizes minimizing the total travel distance while decreasing α gives more weight to minimizing the standard deviation of distances traveled by individual agents.

The chromosome vector $\boldsymbol{x}$ uniquely defines which waypoints are assigned to the agents in which order. The distance between any two points in the environment is obtained using the A* algorithm.

The proposed method combines the genetic algorithm [3] and simulated annealing [4] to solve (1). We use the two-part chromosome representation based on [20], where the first part represents the order of waypoints and the second represents cut positions. The cuts separate the waypoints between agents, a minor modification of the approach in [20], where the second part represents the number of waypoints assigned to agents. Representing the number of cuts rather than the number of assigned waypoints shortens the chromosome's second part by one bin. The visualization of the two-part chromosome is given in Fig. 3. The first part of the chromosome represents the order of 12 numbered waypoints that should be assigned between five agents based on the second part of the chromosome. The second part of the chromosome represents cuts that cut the first part into subroutes assigned to individual agents. The first agent is assigned waypoints 2 and 7 in this order. The second agent is assigned waypoints 9, 0, 3, and 5. The third agent is not assigned any waypoint since the cut position is the same as the previous one. The fourth agent is assigned waypoints 6, 11, and 8. And finally, the fifth agent is assigned the remaining waypoints, 10, 1, and 4.

Often, the first generation is randomly generated in genetic algorithm. However, this approach creates a population where the order of the waypoints is too far from optimal, causing slow convergence of the genetic algorithm. Our method generates the cuts randomly from a uniform distribution, and the order of waypoints is a random permutation of waypoints. Then waypoints associated with each agent are reordered using the Christofides algorithm [21].

We use the two-part chromosome crossover (TCX) based on [22], which significantly decreases the search space compared to the crossover approach in [20]. The parents for crossover are sampled from the population based on systematic resampling [23], where the resampling probability w^i of the i-th population member is the reciprocal of the normalized member's objective (2) defined as

$$w^i = \frac{1}{\frac{1}{\sum_{j=0}^{J} f\left(\boldsymbol{x}^j\right)} f\left(\boldsymbol{x}^i\right)}, \tag{5}$$

where the superscripts i and j are indexing members of the genetic algorithm population with the population size J. The systematic resampling provides the first parent for the crossover. The second parent is obtained using systematic resampling while assuring that a parent is not crossed-over with itself. The parents are crossed-over with the crossover probability p_x, creating a candidate for the next generation. If the parents are not crossed-over, the first parent becomes the candidate for the next generation.

Each part of the new candidate's chromosome mutates with the mutation probability p_m. The mutation uses the swap method [24] that selects two waypoints randomly and swaps their order in the first part of the chromosome. The second part of the chromosome selects one cut randomly and exchanges it with a new random cut. Then, the second part of the chromosome is sorted in ascending order so the cuts are in nondecreasing order. If the second part of the chromosome is mutated, one or more waypoints are associated with a different agent. Adding some waypoints to the beginning or end of the agent's path can significantly deteriorate the candidate. Hence, if the second part of the chromosome is mutated, we reorder

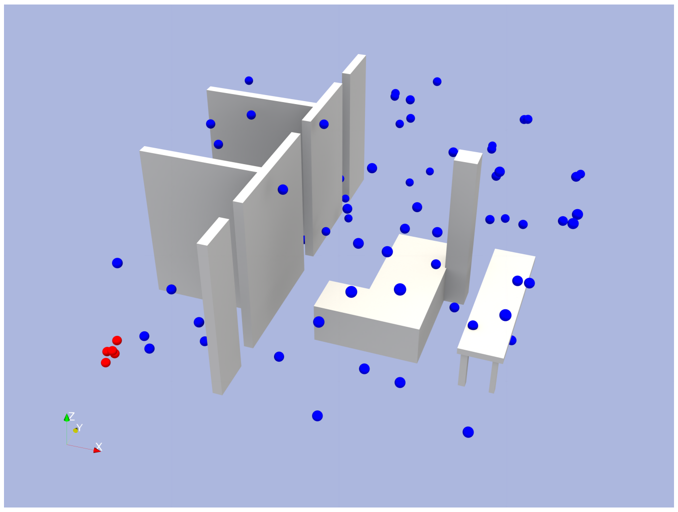

Fig. 4. Visualization of the experimental environment representing a typical indoor environment with separated rooms and medium-sized obstacles (couch, table, supporting column), agents' starting positions (red dots), and 80 waypoints to be visited (blue dots).

the candidate's waypoints using the Christofides algorithm to improve the candidate.

The objective is calculated for the generated candidate. The candidate i in generation k is accepted with the probability

$$p = \begin{cases} 1; & \text{if } f^k\left(\boldsymbol{x}^i\right) < f^{k-1}\left(\boldsymbol{x}^i\right) \\ e^{-\left(f^k\left(\boldsymbol{x}^i\right) - f^{k-1}\left(\boldsymbol{x}^i\right)\right)\frac{1+\log(1+k)}{T_0}}; & \text{otherwise} \end{cases}, \tag{6}$$

where T_0 is the initial temperature constant and $f^k\left(\boldsymbol{x}^i\right)$ is the objective function at the k-th generation for i-th population member. The exponentially decreasing acceptance probability is based on the simulated annealing acceptance rate with logarithmic cooling schedule.

A new candidate is generated according to the method above if the simulated annealing acceptance process rejects the candidate. This loop continues until the candidate is accepted. The candidate generation method is repeated until a whole new population is generated for the next generation.

Finally, the whole process is repeated for the required number of generations.

III. Experimental Results

A virtual environment visualized in Fig. 4 serves to study the performance of the proposed high-level mission planning method. The assumed environment characterizes a simplified four-room apartment with a size of 15x15 meters. For better visualization, the floor, ceiling, and surrounding walls are not shown but are assumed during the path planning. The environment is converted to the octree with depth eight, yielding a spatial resolution of about 6 cm.

The experiments assume five agents and a varying number of waypoints. The agents' initial position is chosen randomly inside a 75 cm large cube in the bottom left corner of the environment. The waypoints are uniformly sampled inside the environment but ensured to be outside of the occupied

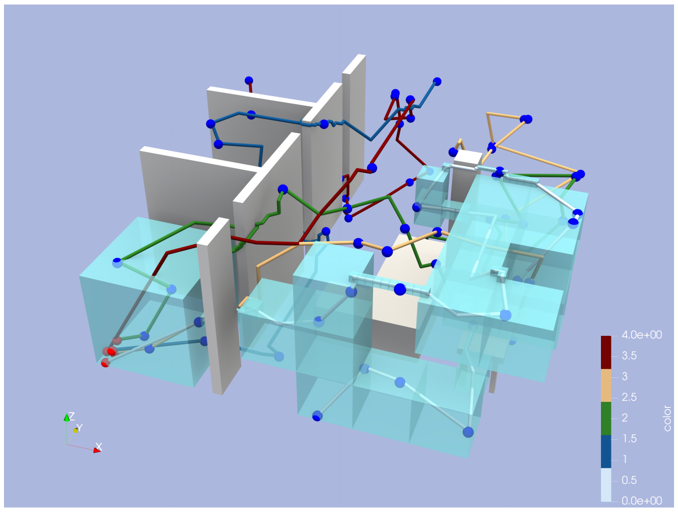

Fig. 5. Visualization of the high-level mission planning output for five agents, 80 waypoints, and $\alpha = 0.25$. The color code corresponds to the agent's number as given by the color bar. Additionally, the corridor serving as constraints for the control algorithm for agent 0 is visualized.

space. A seeded random number generator generates both agents and waypoints. The seed ensures the repeatability of the experiments. Moreover, it ensures the scenario is identical while solving for different α.

In total, we evaluated 20 scenarios where the number of waypoints was 10, 20, 40, 80, and 160, and the α was 0.25, 0.50, 0.75, and 1.00. An example of a scenario for 80 waypoints is shown in Fig. 4, where the red points mark the agents' initial positions, and the blue points mark the waypoints. A workstation with Intel i7-12700 was used to evaluate all the scenarios using Python programming language.

The calculation of A*-based paths between all required pairs of points is parallelizable, where each pair of points is either pair of waypoints or agent-waypoint pair. Then all possible paths through the waypoints comprise the precalculated paths between the individual pairs of points. We measure the duration of the optimization routine without the A* algorithm finding the shortest path between pairs of points since it better captures the complexity of the proposed algorithm because all A*-based paths can be obtained in parallel if deployed to a ground station with a sufficient number of CPU cores.

Fig. 5 shows an example of the high-level mission planning for 80 waypoints and $\alpha = 0.25$. The result shows that all agents are assigned a route since the low value of α prioritizes even workload distribution between agents. Moreover, the agents' routes have similar distances with a standard deviation of only 1.4 meters. The total traveled distance was 249.3 meters. The main output of the proposed method are the corridors that serve as constraints for a control algorithm generating controls for the whole multi-agent system. The light blue cubes show an example of the corridor in Fig. 5 for agent 0 that traveled 49.6 meters.

The total traveled distances and routes standard deviations for the different waypoint counts and configurations of α are

TABLE I
THE TOTAL TRAVELED DISTANCE IN METERS $f_{\mathrm{d}}(\boldsymbol{x})$ FOR DIFFERENT NUMBER OF WAYPOINTS (WP) AND α.

α \ wp	10	20	40	80	160
0.25	64.2	137.4	211.4	249.3	425.4
0.50	51.9	99.6	164.1	233.6	365.0
0.75	51.9	98.4	138.0	213.0	313.5
1.00	51.1	94.6	139.5	212.1	320.6

TABLE II
THE STANDARD DEVIATION OF TRAVELED DISTANCE IN METERS $f_{\sigma}(\boldsymbol{x})$ FOR DIFFERENT NUMBER OF WAYPOINTS (WP) AND α.

α \ wp	10	20	40	80	160
0.25	15.7	2.3	2.0	1.4	2.6
0.50	20.8	18.2	19.6	9.5	9.7
0.75	20.8	21.1	55.2	37.0	70.9
1.00	20.4	37.8	55.8	52.9	92.6

captured in Table I and Table II, respectively. Noticeably, for increasing α, the total travel distance decreases while the route standard deviation increases. An important outcome of the experiments is that it is possible to control the behavior of the multi-agent system by weighting the proposed objective functions (3) and (4). Then based on the mission-specific criteria, we can decide if we aim to minimize the mission duration or the total traveled distance.

The presented results were obtained after 10 000 generations. However, the results are not significantly changing after 3 000 generations, as shown in Fig. 6, where the distances of individual agents (left axis) and the total traveled distance (right axis) are plotted as a function of the number of generations k. With increasing α, the total traveled distance corresponding to $f_{\mathrm{d}}(\boldsymbol{x})$ decreases. However, the workload is

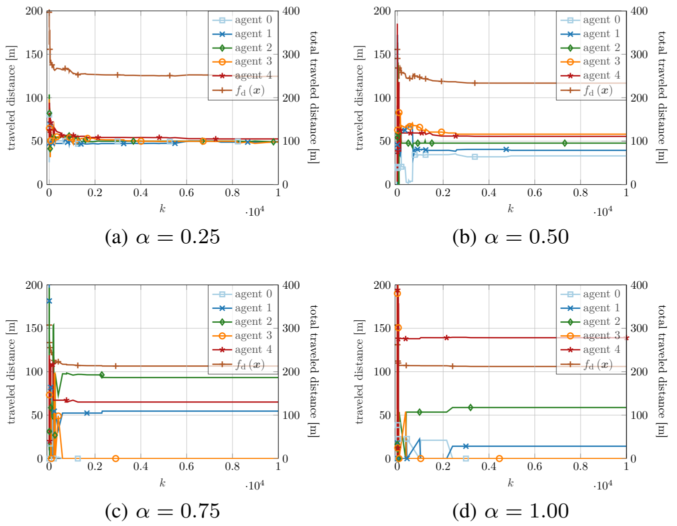

Fig. 6. Plot of the traveled distance of individual agents (left) and the total traveled distance $f_{\mathrm{d}}(\boldsymbol{x})$ (right) for different α at k-th optimization iteration with 80 waypoints.

TABLE III
DURATION OF 3 000 GENERATIONS FOR DIFFERENT NUMBER OF WAYPOINTS ($\alpha = 0.50$)

wp	10	20	40	80	160
t [s]	549.3	839.7	1375.2	2979.1	11544.1

unevenly distributed between the agents since increasing α decreases the weight of $f_\sigma(\boldsymbol{x})$. Interestingly, the total traveled distances for $\alpha = 0.75$ and $\alpha = 1.00$ difference is marginal, while the influence on the even distribution of workload is significant. In the case of 40 and 160 waypoints, the total traveled distance was even worse when the standard deviation was ignored by $\alpha = 1.00$ than when $\alpha = 0.75$ was used.

The duration of the optimization algorithm for 3 000 generations is in Table III. The results show that the time complexity is more than linear in the number of waypoints. If we assume polynomial complexity, we can estimate that the time complexity of the proposed algorithm is $O\left(m^{2.12}\right)$. The highest complexity method is the Christofides algorithm with a known $O\left(n^3\right)$. However, the Christofides algorithm is only called when the second part of the chromosome is mutated. Since the mutation probability is usually kept small (0.05 in our case), the influence on the overall time complexity is also small. The time complexity of the TCX is not studied in [22], but it should be less than $O\left(m \log\left(m\right)\right)$ since only parts of parent chromosomes are sorted during the crossover. Therefore, we believe that complexity mostly comes from the simulated annealing-based acceptance and rejection of candidates. As the optimization progress, the rejection rate increases, and an increasing number of candidates must be generated before it is accepted. We believe the performance could be improved by tuning the annealing schedule, e.g., using adaptive simulated annealing and early stopping when the optimization converges.

IV. CONCLUSION

We proposed a novel approach to multi-agent high-level mission planning for path planning in a cluttered indoor environment. The proposed algorithm solves the problem of assigning waypoints to multiple agents and providing the order in which the agents visit the waypoints. Moreover, it provides unobstructed corridors serving as constraints for a controller planning the optimal controls for steering the agent through the assigned waypoints.

The studied problem of assigning waypoints to agents is similar to an NP-hard MTSP which can be solved exactly with exponential complexity. Our approach achieves polynomial complexity by combining genetic algorithm and simulated annealing heuristics for solving MTSP. We proposed a novel approach to generating the initial generation of candidates based on a graph-based Christofides' approximation to the traveling salesman problem that significantly speeds up the optimization convergence.

Future research topics should aim to expand the proposed method for a 4D environment for collision-free path planning between moving obstacles. Also, the proposed high-level mission planning should be integrated with MPC controller to not only to pre-plan the mission but also execute it.

ACKNOWLEDGMENT

This work has been performed in the framework of the DLR projects *STARE* and *INTAS*.

REFERENCES

[1] T. Wiedemann, D. Shutin, and A. J. Lilienthal, "Model-based gas source localization strategy for a cooperative multi-robot system—a probabilistic approach and experimental validation incorporating physical knowledge and model uncertainties," *Robotics and Autonomous Systems*, vol. 118, pp. 66–79, Aug. 2019. [Online]. Available: https://doi.org/10.1016/j.robot.2019.03.014

[2] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, Jun. 2006. [Online]. Available: https://doi.org/10.1016/j.omega.2004.10.004

[3] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Cambridge, MA: MIT Press, 1975.

[4] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, Jun. 1953. [Online]. Available: https://doi.org/10.1063/1.1699114

[5] O. Cheikhrouhou and I. Khoufi, "A comprehensive survey on the multiple traveling salesman problem: Applications, approaches and taxonomy," *Computer Science Review*, vol. 40, p. 100369, May 2021. [Online]. Available: https://doi.org/10.1016/j.cosrev.2021.100369

[6] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, Feb. 2013. [Online]. Available: https://doi.org/10.1007/s10514-012-9321-0

[7] P. van Oosterom, "The spatial location code," in *Proceedings of the 7th international symposium on spatial data handling: Advances in GIS research II*, 2010.

[8] S. M. LaValle, "Rapidly-exploring random trees : a new tool for path planning," *The annual research report*, 1998.

[9] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996. [Online]. Available: https://doi.org/10.1109/70.508439

[10] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, no. 1, pp. 269—-271, 1959.

[11] P. Hart, N. 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

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

[13] J. B. Rawlings, "Tutorial overview of model predictive control," *IEEE control systems magazine*, vol. 20, no. 3, pp. 38–52, 2000.

[14] M. Ibrahim, C. Kallies, and R. Findeisen, "Learning-supported approximated optimal control for autonomous vehicles in the presence of state dependent uncertainties," in *2020 European Control Conference (ECC)*. IEEE, 2020, pp. 338–343.

[15] M. Ibrahim, M. Kögel, C. Kallies, and R. Findeisen, "Contract-based hierarchical model predictive control and planning for autonomous vehicle," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15 758–15 764, 2020.

[16] M. Kögel, M. Ibrahim, C. Kallies, and R. Findeisen, "Safe hierarchical model predictive control and planning for autonomous systems," *International Journal of Robust and Nonlinear Control*, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rnc.6808

[17] A. Kuenz, *High Performance Conflict Detection and Resolution for Multi-Dimensional Objects*. Braunschweig: Forschungsbericht Deutsches Zentrum für Luft- und Raumfahrt, 2015.

[18] J. W. J. Williams, "Algorithm 232 - Heapsort," *Communications of the ACM*, vol. 7, no. 6, pp. 347–348, Jun. 1964. [Online]. Available: https://doi.org/10.1145/512274.512284

[19] J. Vörös, “A strategy for repetitive neighbor finding in octree representations,” *Image and Vision Computing*, vol. 18, no. 14, pp. 1085–1091, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0262885600000494

[20] A. E. Carter and C. T. Ragsdale, “A new approach to solving the multiple traveling salesperson problem using genetic algorithms,” *European Journal of Operational Research*, vol. 175, no. 1, pp. 246–257, Nov. 2006. [Online]. Available: https://doi.org/10.1016/j.ejor.2005.04.027

[21] N. Christofides, “Worst-case analysis of a new heuristic for the travelling salesman problem,” *Operations Research Forum*, vol. 3, 1976.

[22] S. Yuan, B. Skinner, S. Huang, and D. Liu, “A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms,” *European Journal of Operational Research*, vol. 228, no. 1, pp. 72–82, Jul. 2013. [Online]. Available: https://doi.org/10.1016/j.ejor.2013.01.043

[23] P. Kozierski, M. Lis, and J. Zietkiewicz, “Resampling in particle filtering - comparison,” *Studia z Automatyki i Informatyki*, vol. 38, pp. 35–64, 1 2013.

[24] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA: Addison-Wesley Longman Publishing Co., Inc., 1989.