Constrained Multi-agent Path Planning Problem

Ali Maktabifard[1](✉), Dávid Földes[1], and Bendegúz Dezső Bak[2]

[1] Faculty of Transportation Engineering and Vehicle Engineering, Department of Transport Technology and Economics, Budapest University of Technology and Economics, Budapest, Hungary
a.maktabifard@edu.bme.hu, foldes.david@kjk.bme.hu

[2] Faculty of Mechanical Engineering, Department of Fluid Mechanics, Budapest University of Technology and Economics, Budapest, Hungary
bak.bendeguz@gpk.bme.hu

Abstract. Planning the most efficient routes in a cooperative manner is a challenge for many mobility and logistics service providers. In this paper, a new methodological approach is presented based on the Multi-Agent Path Planning (MAPP) problem which is a variant of the classical Multiple Traveling Salesmen Problem (MTSP). Given a team of m agents that must visit n targets, the optimal paths plan (a set of m paths) should be determined such that each target is visited only once. Minimizing the time of this cooperative operation is the optimization goal in this study. Thus, the plan is optimal, if the longest path in the plan is the shortest possible (Min-Max problem). In order to deal with more practical situations, two additional constraints are applied: the maximum number of targets each agent is allowed to visit, and the maximum range for each agent. Accordingly, a so-called Constrained Multi-Agent Path Planning (CMAPP) problem is elaborated in this paper. An easy to apply Genetic Algorithm (GA) is presented, which improves the paths using genetic-like operators and a heuristic method. The applicability of the approximate solution was tested in four random scenarios where it showed a decent performance. The developed method can be used for route planning of mobility and logistics services in which several destinations must be reached by a fleet of vehicles (e.g., group ride-sharing, last-mile delivery).

Keywords: Multi-Agent Planning Problem · Genetic Algorithm · Routing Problem · Optimization · Metaheuristic · MAPF · MTSP

1 Introduction

The importance of cooperative path planning for vehicle fleets has been demonstrated by various applications such as logistics and delivery services [19,26], ride-sharing services [13,25], and mission planning for autonomous/unmanned vehicles [4,28]. In general, a multi-agent planner is required for cooperative path planning of vehicle fleets. This planning problem can be defined as finding a set of paths that allows a fleet of agents to reach a specified number of targets in the minimum amount of time. This problem is similar to the classical Multiple Traveling Salesmen Problem (MTSP) [1], and its variants such as the Multi-Agent Path Planning (MAPP) problem [16]. The classical MTSP is generally defined as follows. Given n cities (targets) and m salesmen (agents), the aim is to find m tours (closed paths) starting and ending at a depot (initial position of the agents) such that each target is visited only once and the total cost of visiting all targets is minimized. The cost metric can be expressed in terms of distance, time, etc., [1]. The MAPP problem is similar to the classical MTSP with two differences:

- Subtours (open paths) are considered in the MAPP problem, so each agent starts and ends its path at two distinct points.
- Agents can have different initial positions.

More practical situations can be modeled by applying additional restrictions to these combinatorial optimization problems. These restrictions affect the agents mostly by restricting their path length, workload, or operational range [13,19,25,26]. As a novelty, in this study the cooperative path planning for vehicle fleets is modeled as a so-called Constrained Multi-Agent Path Planning (CMAPP) problem. This problem is formed by applying two additional constraints to the MAPP problem: the maximum number of targets each agent is allowed to visit, and the maximum range for each agent. This paper presents a two-step solution method composed of an initial solution and a complex solution for this problem. The initial solution benefits from a new targets assignment algorithm. The complex solution uses a Genetic Algorithm (GA) to improve the result of the initial solution. The additional constraints are applied in both initial and complex solutions. The aim of this study is to show the applicability of the developed method.

The remainder of the paper is organized as follows. Section 2 reviews solution methods for similar problems. The CMAPP problem is described in detail in Sect. 3. The methodology of the proposed solution is described in Sect. 4; then results are presented and discussed in Sect. 5. Finally, conclusions are drawn in Sect. 6.

2 Review of Solution Methods

Solving the MTSP and the MAPP problem is difficult because of their complex combinatorial character (NP-hardness). In general, two types of approaches are used to tackle the MTSP [12]: *exact* and *heuristic-based* approaches.

The exact approaches are based on either the transformation of the MTSP to an equivalent Traveling Salesman Problem (TSP), or relaxing some constraints of the problem [1,12]. The problem is solved by applying exact methods such as Branch-and-Bound [6], Cutting Planes [20], and Integer Linear Programming

Table 1. Literature instances on the MTSP having similarities to the CMAPP problem considered in our study

Literature	Solution method	Multi-depot MTSP	Min-Max MTSP	Min. and Max. No. of targets for each agent
[17]	Integer linear programming formulations + new bounding and Subtour Elimination Constraints (SECs)	×		×
[18]	A heuristic approach based on an Evolution Strategy (ES)	×	×	
[8]	An ACO algorithm	×		×
[30]	Two variants of Parthenogenetic Algorithm (PGA)	×		Only Min.
[15]	An Ant Colony-Parthenogenetic Algorithm (AC-PGA)	×		×
[14]	A heuristic approach based on a graph simplification method and the 2-OPT algorithm	×		

Formulations [17]. The exact approaches are restricted to the MTSPs with reasonable sizes (Euclidean and Non-Euclidean problems up to 100 and 500 cities (targets), respectively [6]) because their performance is highly dependent on the size of the problem. Accordingly, the solution runtime encounters an exponential rise with increasing the problem size [1,12]. Additionally, transforming the MTSP to an equivalent TSP might result in an even more difficult problem to solve, especially using the exact approaches [6,20].

The heuristic-based approaches solve the problem by applying approximate heuristic methods such as Ant Colony Optimization (ACO) [8,12,15], Genetic Algorithm (GA) [12,15,29,30], Simulated Annealing [24], Neural Networks [23], and Tabu Search [22]. The heuristic-based approaches can achieve near-optimal solutions in a reasonable amount of time even for larger problems [12].

The problems tackled in several previous papers on the MTSP have some similarities to the CMAPP problem considered in our study (see Table 1). These similarities include considering multiple depots for the MTSP, minimizing the length (cost) of the longest tour (Min-Max MTSP), and applying additional constraints.

There are only a few studies on the variants of the MAPP problem. In [9], a GA was introduced to solve a so-called Subtour problem which is similar to the MAPP problem with one difference: there is only one agent. This method was developed for solving the MAPP problem in [10], and a similar method was proposed in [16] to address the MAPP problem. In [27], a heuristic approach based on a graph simplification method was presented to solve a multi-depot open tours MTSP which is basically the MAPP problem with Min-Sum optimization objective.

Concluding the literature review, the MTSP is a well-studied problem, while there are only a few studies on the MAPP problem. Furthermore, previous studies have not solved the MAPP problem with our additional constraints.

3 Constrained Multi-agent Path Planning Problem: General Terms and Notation

In this section, the required foundation from Graph theory, and the notations of the classical MTSP and the MAPP problem are presented. In addition, our additional constraints for the MAPP problem are defined.

According to Graph theory, a *graph* is an ordered pair $G = (V, E)$ consisting of: $V = \{v_1, ..., v_m\}$, a set of m *vertices* (nodes), and $E = \{(v_i, v_j) \mid v_i, v_j \in V,\ \ i \neq j\}$, a set of *edges* (links) connecting vertices v_i and v_j. This type of graph may be precisely referred to as *simple graph* which means multiple edges connecting the same two vertices are not allowed. In the context of the classical MTSP and the MAPP problem, the targets and the points where the agents begin their journey, are considered as vertices. In this study, only undirected graphs are taken into consideration. In an *undirected graph*, edges are comprised of unordered pairs of vertices where $(v_i, v_j) = (v_j, v_i)$. Moreover, if in a graph all vertices of V are connected to each other, the graph is a *complete graph* and it is designated by $K_m(V)$ where m is the number of vertices constituting the vertex set V.

A *path/cycle* is a sequence of edges connecting a sequence of vertices. If the sequence of vertices is composed of distinct vertices, a *simple path/cycle* is given with the exception that for simple cycle the starting and ending vertices are repeated. Here we only consider simple paths/cycles, so henceforth paths/cycles simply refer to simple paths/cycles. $P = (V_1, E_1)$ is a path in $G = (V, E)$ if: $V_1 = \{v_1, ..., v_k\} \subset V$, and $E_1 = \{(v_1, v_2), (v_2, v_3), ..., (v_{k-1}, v_k)\} \subset E$. This means a path consisting of k vertices is a sequence of $k - 1$ edges connecting these vertices in which each two consecutive edges share a vertex in common. Likewise, $C = (V_2, E_2)$ is a cycle in $G = (V, E)$ if: $V_2 = \{v_1, ..., v_k\} \subset V$, and $E_2 = \{(v_1, v_2), ..., (v_{k-1}, v_k), (v_k, v_1)\} \subset E$. In other words, a cycle comprising k vertices is a sequence of k edges connecting these vertices in which each two consecutive edges as well as the first and the last edge share a vertex in common. Hence, a cycle starts and ends at the same vertex (tour), while a path starts and ends at different vertices (subtour).

The number of edges in a path or cycle is called the length of that path or cycle. For $G = (V, E)$, the set of all paths and cycles with length k are designated by $\mathbb{P}_k(G)$ and $\mathbb{C}_k(G)$, respectively. A *weight* (cost) $w(v_i, v_j)$ can be assigned to an edge. If every edge of a graph has a weight, the graph is called a *weighted graph*. In a weighted graph if $w(v_i, v_j) = w(v_j, v_i)$ is valid for every edge, the weighted graph is called *symmetric*. In this study, for simplicity, the Euclidean distance between two vertices of each edge is assigned as the weight of that edge

$w(v_i, v_j) = w(v_j, v_i) = |\vec{r}(v_i) - \vec{r}(v_j)|$; note that the problems discussed here are not restricted to Euclidean distance, so any desired measure (e.g., rectilinear distance, actual driving distance) can be assigned as the weight of edges. For a path $P \in \mathbb{P}_k(G)$, the sum of its edges' weights is called the total cost of the path:

$$c(P) = \sum_{i=1}^{k} w(v_i, v_{i+1}) \tag{1}$$

Similarly, the total cost of a cycle $C \in \mathbb{C}_k(G)$ is:

$$c(C) = \sum_{i=1}^{k-1} w(v_i, v_{i+1}) + w(v_k, v_1) \tag{2}$$

If there is no associated weight to each edge, the total cost of a path or cycle is simply equal to the length of the path or cycle.

The classical MTSP and the MAPP problem can be formulated on the basis of the foundation from Graph theory described above. Consider $A = \{a_1, ..., a_m\}$ and $T = \{t_1, ..., t_n\}$ as the set of m agents and n targets, respectively. The agents and targets are located in Euclidean space. The location of the i^{th} agent and the j^{th} target is therefore determined by $\vec{r}(a_i)$ and $\vec{r}(t_j)$, respectively. The classical MTSP is formulated as follows. Consider a as the depot for all agents, so $\forall\, a_i = a$ and $\forall\, \vec{r}(a_i) = \vec{r}(a)$. The configuration space of the problem is the complete graph $K_{n+1}(V)$ with the vertex set $V = T \cup a$. Consider C_i as a cycle with length k_i that starts and ends at vertex a (the depot). Let $\mathbb{C} = \{C_1, ..., C_m\}$ be the set of m cycles C_i with length k_i. The aim is to determine $\mathbb{C}$ such that each target is visited only once (visitation only by one agent), and the total cost of $\mathbb{C}$ (Eq. (3)) is minimized (Min-Sum problem).

$$c(\mathbb{C}) = \sum_{i=1}^{m} c(C_i) \tag{3}$$

The total cost of $\mathbb{C}$ is the sum of the lengths (costs) of all m cycles C_i constituting $\mathbb{C}$ (see Eq. (3)).

Likewise, the MAPP problem is formulated as follows. For each agent, the configuration space of the problem is the complete graph $K_{n+1}(V_i)$ with the vertex set $V_i = T \cup a_i$. The Euclidean distance between two vertices of each edge (e.g., v_x and v_y) is assigned as the weight of that edge; $w(v_x, v_y) = w(v_y, v_x) = |\vec{r}(v_x) - \vec{r}(v_y)|$ where $v_x, v_y \in V_i$. As a result, $K_{n+1}(V_i)$ is a symmetric weighted graph. Consider P_i as a path with length k_i starting at vertex a_i. Let $\mathbb{P} = \{P_1, ..., P_m\}$ be the set of m paths P_i with length k_i that each pair of them do not have any vertex in common (except possible same position for starting points). The aim is to determine $\mathbb{P}$ such that each target is visited only once (visitation only by one agent), and the cost of the path with the largest cost in $\mathbb{P}$ (Eq. (4)) is minimized.

$$c_m(\mathbb{P}) = \max_{i=1}^{m} c(P_i) \tag{4}$$

Therefore, the MAPP problem is a Min-Max problem in which a team of m agents must visit n targets in the minimum amount of time. The whole team of agents should be used, so each agent must visit at least one target ($k_i \geq 1$). However, the number of targets visited by each agent can be different.

In this study, our additional constraints for the MAPP problem are applied independently of each other. These constraints are defined as follows. In the MAPP problem, if the maximum number of targets that can be assigned to each agent is limited, the solution would be affected since the maximum length of each agent's path is restricted. Accordingly, we introduced the maximum number of targets each agent is allowed to visit as the first additional constraint. Let $Q = \{q_1, ..., q_m\}$ be the set of the maximum number of targets for each agent, so the maximum number of targets that can be assigned to the i^{th} agent is q_i. As a result, the length of each agent's path is limited to: $1 \leq k_i \leq q_i$. In the case of applying this additional constraint, if the sum of all elements of Q is less than the number of targets ($\sum_{i=1}^{m} q_i < n$), then the fleet of m agents cannot visit all n targets. Hereafter, this situation is referred to as the *lack-of-capacity issue*. Additionally, if the maximum number of targets for each agent is the same for all agents ($\forall\, q_i = q$), the fleet of agents is called a homogeneous capacity fleet. In this study, for solving the problem with the first additional constraint, a homogeneous capacity fleet was assumed.

Furthermore, we defined the maximum range of the i^{th} agent as the maximum Euclidean distance from the position of a_i that can be accessed by the i^{th} agent. This range is unlimited without applying any constraint on it, so each agent can access all targets regardless of their Euclidean distance from its position. However, if this range is limited, the only accessible targets for each agent are located within its maximum range. This not only impacts the assignment of targets to agents, but also can influence the total cost of each agent's path. Accordingly, we introduced the maximum range for each agent as the second additional constraint. Let $R = \{r_1, ..., r_m\}$ be the set of the maximum range for each agent, so the maximum range of the i^{th} agent is r_i. Consequently, the j^{th} target can be accessible for the i^{th} agent only if: $|\vec{r}(a_i) - \vec{r}(t_j)| \leq r_i$. In the case of applying this additional constraint, if there would be at least one target that is not in the maximum range of any agent ($|\forall\, \vec{r}(a_i) - \vec{r}(t_j)| > r_i$), then the fleet of m agents cannot visit all n targets. Henceforth, this situation is referred to as the *out-of-range issue*. Moreover, if the maximum range for each agent is the same for all agents ($\forall\, r_i = r$), the fleet of agents is called a homogeneous range fleet. In this study, for solving the problem with the second additional constraint, a homogeneous range fleet was assumed.

4 Methodology of the Approximate Solution

In this study, a two-step solution based on the method developed in [16] is presented for the CMAPP problem. The solution method is composed of an initial solution and a complex solution. The complex solution improves the result of the initial solution using a GA that applies genetic-like operators and a heuristic method. A solution is a set of paths that is called *Plan* ($\mathbb{P}$) in this context. The Plan is optimal if the longest path in the Plan is the shortest possible (Min-Max problem). Thus, the solution algorithm is as follows:

– Step 1: a Plan $\mathbb{P}$ is generated by the initial solution.
– Step 2: the complex solution is applied to the Plan $\mathbb{P}$ to minimize the cost $c_m(\mathbb{P})$ (see Eq. (4)) and eventually obtain the near-optimal Plan.

4.1 Initial Solution

The initial solution generates the Initial Plan (a starting set of paths). Consider $A = \{a_1, ..., a_m\}$ and $T = \{t_1, ..., t_n\}$ as the set of m agents and n targets, respectively. A Plan is viable if each pair of paths do not have any vertex in common (except possible same starting points). For the order of planning, we assume that the targets are assigned to the agents in the following order: $a_1 \rightarrow a_2 \rightarrow \cdots \rightarrow a_m$. The targets assignment algorithm is as follows:

– Step 1: a target $t_{i_1} \in T_1 = T$ is selected and assigned to the first agent.
– Step 2: a target $t_{i_2} \in T_2 = T_1 - t_{i_1}$ is selected and assigned to the second agent.
  $\vdots$
– Step m: a target $t_{i_m} \in T_m = T_{m-1} - t_{i_{m-1}}$ is selected and assigned to the m^{th} agent.

A viable Plan $\mathbb{P} = \{P_1, ..., P_m\}$ is generated by iterating this algorithm until all targets would be assigned to the agents.

However, a target selection approach is also required for selecting a target in every step of the targets assignment algorithm. Therefore, different initialization methods can be employed by adopting various target selection approaches. In this study, two initialization methods were employed:

1. *Greedy* initialization method:
   It selects targets on the basis of the nearest neighbor heuristic and assigns them to the agents. For each agent, the nearest unassigned target to the previously assigned target to that agent is selected in every iteration of the targets assignment algorithm. In order to select the first target for each agent, the nearest unassigned target to the position of that agent (a_i) is selected.
2. *Random* initialization method:
   It selects targets randomly and assigns them to the agents. For each agent, a random unassigned target is selected in every iteration of the targets assignment algorithm.

The solution for the non-constrained MAPP problem was also considered in order to enable us to evaluate the effect of applying our two additional constraints that were applied independently of each other. Hence, six different solution modes were analyzed in this study (see Table 2).

Table 2. Analyzed solution modes

Solution mode	Initialization method		Additional constraint	
	Greedy	*Random*	*Max. number of targets for each agent*	*Max. range for each agent*
1	×			
2	×		×	
3	×			×
4		×		
5		×	×	
6		×		×

4.2 Complex Solution

In order to evolve a Plan $\mathbb{P}$ towards a near-optimal Multi-Agent Plan, the complex solution is applied to the paths $P_i \in \mathbb{P}$. The complex solution uses a GA applying three genetic-like operators and another operator employing a heuristic method to minimize the cost $c_m(\mathbb{P})$ (see Eq. (4)). This reduces the length of the longest path in the Plan, and eventually the near-optimal Plan is obtained. The algorithm of the complex solution is the same as a classical GA [11] with one exception: there is only one Plan $\mathbb{P}$ generated by the initial solution to be evolved (the population size is 1).

In every evolutionary iteration, the developed operators are applied sequentially to a Plan $\mathbb{P}$ and generate a new Plan $\mathbb{P}^{'}$, then:

- If $c_m(\mathbb{P}^{'}) < c_m(\mathbb{P})$, Plan $\mathbb{P}^{'}$ is kept for the next evolutionary iteration and Plan $\mathbb{P}$ is discarded.
- If $c_m(\mathbb{P}^{'}) > c_m(\mathbb{P})$, Plan $\mathbb{P}$ is kept for the next evolutionary iteration and Plan $\mathbb{P}^{'}$ is discarded.

In every evolutionary iteration, the following operators are applied in the same order:

1. *Crossover* operator:
   It is stochastically applied with the probability of $\mathcal{P}_{crossover}$. It selects two paths stochastically either by the best-worst selection with the probability of $\mathcal{P}_{best-worst}$ or by random selection with the probability of $1 - \mathcal{P}_{best-worst}$, then each of them is randomly cleaved (cleaving position can be different for each of them) into two parts and the parts are swapped.
2. *Mutation* operator:
   It is stochastically applied with the probability of $\mathcal{P}_{mutation}$. It randomly selects two paths and swaps two randomly selected targets between them.
3. *Migration* operator:
   It is stochastically applied with the probability of $\mathcal{P}_{migration}$. It randomly selects two paths P_i and P_j, then it removes a randomly selected target from

P_i and adds it to P_j. Obviously, in this procedure the length of P_i reduces, while the length of P_j enlarges.

4. *Boosting* operator:
It is stochastically applied with the probability of $\mathcal{P}_{boost}$. It enhances the quality of paths by applying the 2-OPT algorithm [2]. This algorithm examines whether the inequality (5) between each four targets v_i, v_{i+1}, v_j, v_{j+1} of a path is valid or not. If it is valid, edges (v_i, v_{i+1}) and (v_j, v_{j+1}) are substituted with edges (v_i, v_j) and (v_{i+1}, v_{j+1}), respectively. Applying this algorithm generates a shorter path.

$$c(v_i, v_{i+1}) + c(v_j, v_{j+1}) > c(v_i, v_j) + c(v_{i+1}, v_{j+1}) \tag{5}$$

5 Results and Discussion

In order to demonstrate the applicability of the developed solution, four *random scenarios* with 100 test cases for each one were considered (see Table 3). In all test cases, the agents and targets were spread out randomly in a dimensionless domain (*domain area* $= 1 \times 1$). The cost reduction Δc_m was defined as the cost reduction of the Final Plan compared to the Initial Plan (see Eq. (6)). The following evaluation parameters were considered: $\overline{\Delta c_m}$ is the average Δc_m for 100 test cases; $\overline{c_m}(\mathbb{P}_{final})$ is the average final cost for 100 test cases; q_i is the maximum number of targets for each agent; $\overline{r_i}$ is the average maximum range for each agent for 100 test cases; and $\mathcal{T}_{100}$ is the runtime for 100 test cases.

$$\Delta c_m = \frac{c_m(\mathbb{P}_{Initial}) - c_m(\mathbb{P}_{Final})}{c_m(\mathbb{P}_{Initial})} \times 100\% \tag{6}$$

Table 3. Test scenarios

Scenario	Number of agents (m)	Number of targets (n)
1	10	100
2	9	100
3	20	200
4	10	200

In terms of the first additional constraint, in each test case the maximum number of targets for each agent was determined by $q_i = \lceil \frac{n}{m} \rceil$ to ensure that the lack-of-capacity issue would not occur. Regarding the second additional constraint, in each test case the developed solution numerically calculates the minimum value for the maximum range for each agent such that the out-of-range issue would not occur. As a result, here the maximum range for each agent is a case-sensitive variable.

The operators of the complex solution were applied with the probabilities of: $\mathcal{P}_{crossover} = 0.7$, $\mathcal{P}_{best-worst} = 0.5$, $\mathcal{P}_{mutation} = 0.4$, $\mathcal{P}_{migration} = 0.6$, and

$\mathcal{P}_{boost} = 0.3$. Simulations were run for 150000 evolutionary iterations for each test case by means of MATLAB R2021b. The hardware configuration of the PC used was as follows. CPU: AMD Ryzen$^{\text{TM}}$ 5-5500U, RAM: 8 GB.

The computational results are shown in Table 4. Accordingly, if the maximum number of targets for each agent was restricted:

– in the case of employing Greedy initialization, the average final cost (except in scenario 1) and the runtime decreased, while the average cost reduction (except in scenario 1) increased compared to the non-constrained solution with Greedy initialization.
– in the case of employing Random initialization, the runtime (except in scenario 1) and the average cost reduction decreased, whereas the average final cost increased compared to the non-constrained solution with Random initialization.

Moreover, if the maximum range for each agent was limited:

– in the case of employing Greedy initialization, the average cost reduction and the average final cost decreased compared to the non-constrained solution with Greedy initialization. A general trend was not observed regarding the runtime.
– in the case of employing Random initialization, the average final cost in the scenarios with more targets (200) and the average cost reduction decreased, while the average final cost in the scenarios with fewer targets (100) increased compared to the non-constrained solution with Random initialization. A general trend was not observed regarding the runtime.

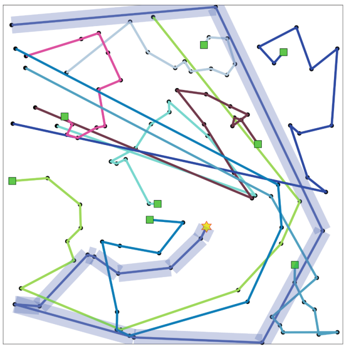
Additionally, in the absence of any additional constraints, Random initialization led to a smaller average final cost compared to Greedy initialization in scenarios with fewer targets (100). However, in the presence of additional constraints, Random initialization always led to a larger average final cost compared to Greedy initialization. An interesting result was that $\overline{r_i}$ was always less than 34% of the domain diameter ($\sqrt{1+1}$). Furthermore, if the number of agents and targets increased while $\frac{n}{m}$ remained constant, the average final cost decreased in the case of applying the additional constraints along with Greedy initialization. For 10 agents, when the number of targets increased by 100%, the average final cost experienced an increase averaging 47.4% and 59.6% across solution modes 1 to 3, and solution modes 4 to 6, respectively. For 100 targets, when the number of agents decreased by 10%, the average final cost underwent an increase averaging 5.4% across all solution modes. For 200 targets, when the number of agents decreased by 50%, the average final cost experienced an increase averaging 61% and 45.2% across solution modes 1 to 3, and solution modes 4 to 6, respectively.

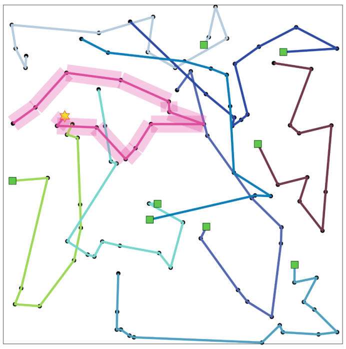
In order to visualize some instances of the evolution accomplished by the approximate solution, Fig. 1 and Fig. 2 illustrate the Initial Plans and the Final Plans generated by three solution modes for two different test cases. In these figures, the yellow star is the initial position of the agent with the longest path; the green squares are the initial positions of other agents; the paths are shown

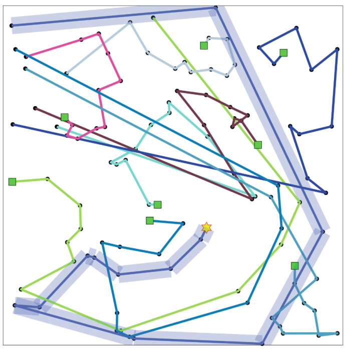
Table 4. Computational results

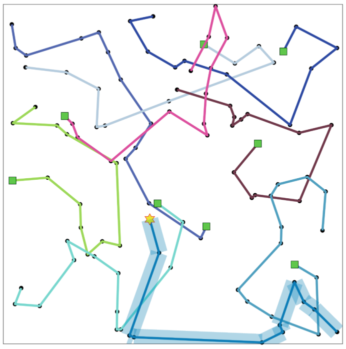
Scenario	Solution mode	$\overline{\Delta c_m}$ [%]	$\overline{c_m}$ ($\mathbb{P}_{final}$)	q_i	$\overline{r_i}$	$\mathcal{T}_{100}$ [s]
1	1	42.2161	1.1562			4275.8578
	2	39.5385	1.2115	10		3994.7323
	3	32.4922	1.1229		0.4466	4401.3488
	4	82.4312	1.1477			3702.8841
	5	80.3621	1.2843	10		4007.1264
	6	74.9173	1.1801		0.4466	4243.5765
2	1	41.7696	1.2622			4547.9586
	2	43.6597	1.2183	12		4446.8247
	3	32.6158	1.2054		0.4748	4146.6348
	4	83.0654	1.2370			4387.3790
	5	82.8877	1.2500	12		4157.2272
	6	76.1382	1.3021		0.4748	4302.6578
3	1	36.8756	1.1599			6314.7105
	2	38.3951	1.1295	10		6094.6654
	3	31.0760	0.9202		0.3416	6569.2436
	4	79.5629	1.3924			6533.8671
	5	77.7243	1.5373	10		6235.4490
	6	73.1099	1.0898		0.3416	6850.0604
4	1	31.1765	1.8080			7881.1427
	2	32.6695	1.7667	20		7332.4831
	3	28.0342	1.5705		0.4537	7552.6412
	4	85.0111	1.8575			7835.9091
	5	82.7509	2.1332	20		7334.5086
	6	80.2113	1.7799		0.4537	7824.6231

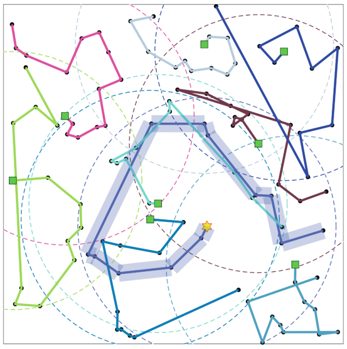
by solid lines and the longest path is highlighted in all illustrated Plans; and dashed-line circles depict the maximum range of the agents. Figure 1 shows the Plans obtained for a test case with 9 agents and 100 targets. In this test case the solution achieved the largest Δc_m (see Eq. (6)) among 100 test cases of scenario 2 in the case of applying Greedy initialization along with the constraint on the maximum number of targets for each agent. In the absence of any additional constraints, the cost $c_m(\mathbb{P})$ (see Eq. (4)) decreased from 3.16 in the Initial Plan (Fig. 1(a)) to 1.24 in the Final Plan (Fig. 1(b)). If the maximum number of targets for each agent was restricted to $q_i = 12$, the cost $c_m(\mathbb{P})$ reduced from 3.16 in the Initial Plan (Fig. 1(c)) to 1.17 in the Final Plan (Fig. 1(d)). Lastly, if the maximum range for each agent was limited to $r_i = 0.38$, the cost $c_m(\mathbb{P})$ dropped from 1.59 in the Initial Plan (Fig. 1(e)) to 1.17 in the Final Plan (Fig. 1(f)). Furthermore, the Plans generated for a test case with 10 agents and 200 targets


(a) Initial Plan, solution mode 1, scenario 2


(b) Final Plan, solution mode 1, scenario 2


(c) Initial Plan, solution mode 2, scenario 2


(d) Final Plan, solution mode 2, scenario 2


(e) Initial Plan, solution mode 3, scenario 2

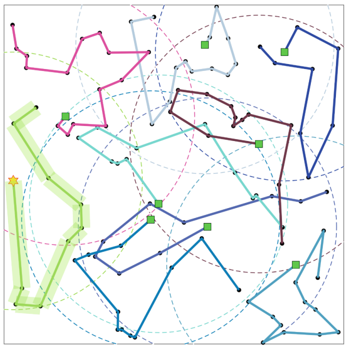
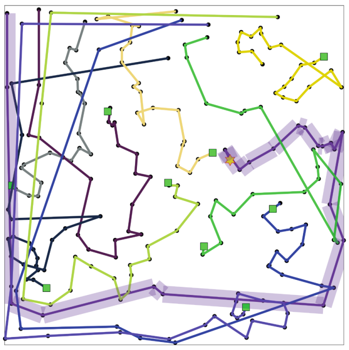
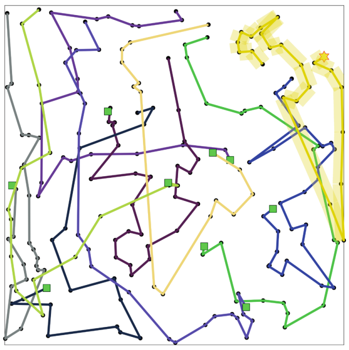
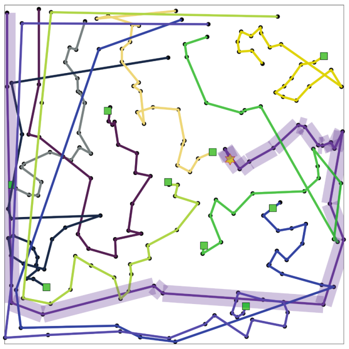
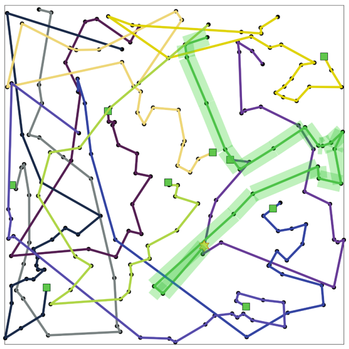

(f) Final Plan, solution mode 3, scenario 2

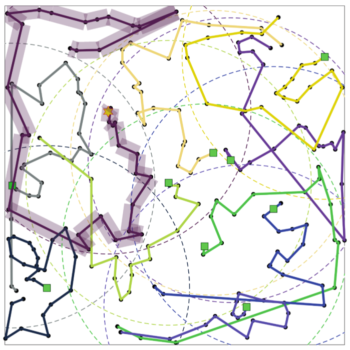
Fig. 1. Initial Plans (left) vs. Final Plans (right) generated by solution modes 1, 2 and 3; scenario 2; the test case in which the solution achieved the largest Δc_m in the case of applying Greedy initialization along with the constraint on the maximum number of targets for each agent. (Color figure online)


(a) Initial Plan, solution mode 1, scenario 4


(b) Final Plan, solution mode 1, scenario 4


(c) Initial Plan, solution mode 2, scenario 4


(d) Final Plan, solution mode 2, scenario 4


(e) Initial Plan, solution mode 3, scenario 4

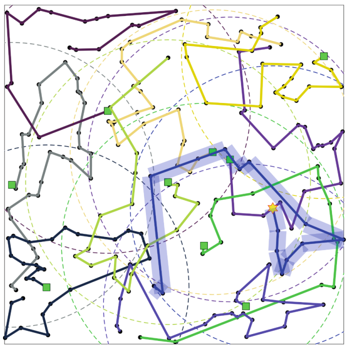

(f) Final Plan, solution mode 3, scenario 4

Fig. 2. Initial Plans (left) vs. Final Plans (right) generated by solution modes 1, 2 and 3; scenario 4; the test case in which the solution achieved the largest Δc_m in the case of applying Greedy initialization along with the constraint on the maximum range for each agent. (Color figure online)

are illustrated in Fig. 2. In this test case the solution attained the highest Δc_m among 100 test cases of scenario 4 if Greedy initialization was applied together with the constraint on the maximum range for each agent. In the absence of any additional constraints, the cost $c_m(\mathbb{P})$ reduced from 2.86 in the Initial Plan (Fig. 2(a)) to 1.74 in the Final Plan (Fig. 2(b)). If the maximum number of targets for each agent was limited to $q_i = 20$, the cost $c_m(\mathbb{P})$ dropped from 2.86 in the Initial Plan (Fig. 2(c)) to 1.95 in the Final Plan (Fig. 2(d)). Finally, if the maximum range for each agent was restricted to $r_i = 0.42$, the cost $c_m(\mathbb{P})$ decreased from 2.60 in the Initial Plan (Fig. 2(e)) to 1.52 in the Final Plan (Fig. 2(f)).

Comparing our developed method with previously proposed methods using a similar GA for the MAPP problem, our method is novel because despite the similarity in utilizing the mutation operator and the 2-OPT algorithm in [10], their initialization phase (initial solution) and crossover operator function differently. Moreover, our additional constraints were not applied in [16] and their initialization phase uses a different targets assignment algorithm. The presented method can be extended by applying other practical constraints realizing time windows and priorities [5,13,19,25], conflict-free paths [3,7], and multi-modal itinerary planning [21].

6 Conclusions

This paper presents a cooperative path planner based on the Multi-Agent Path Planning (MAPP) problem with additional constraints to model more practical situations. The proposed solution uses a Genetic Algorithm (GA) applying genetic-like operators and a heuristic method. It evolves an Initial Plan towards a near-optimal Multi-Agent Plan by minimizing the length of the longest path.

The applicability of the approximate solution was tested in four random scenarios. The computational results show that restricting the maximum number of targets for each agent can mostly improve the performance of the solution if the Initial Plan is generated by Greedy initialization. Furthermore, on average the cost of the Final Plan can be decreased by limiting the maximum range for each agent if the Initial Plan is generated by Greedy initialization. This trend is observed in the scenarios with more targets (200 in this study) even if the Initial Plan is generated by Random initialization. These results show that using a simple yet efficient initial solution, the developed method can potentially enhance the efficiency of route planning for a fleet of vehicles when the aim is to optimize open paths (subtours) in a cooperative manner, and the vehicles of the fleet have a limited operational range or a balance between their workload is desired. This situation can be observed in a variety of mobility and logistics services such as crowd-sourced delivery (e.g., TOURMIX), group ride-sharing, general home delivery, and airport shuttles.

The ultimate goal of this work is to simulate the challenges observed in cooperative path planning for vehicle fleets. Thus, it can be extended by applying other practical constraints like time windows and priorities. Our future work will focus on developing a Multi-Agent route planner for shared mobility services using the method presented here but by calculating the length of paths based on map data (i.e., actual driving distance).

References

1. Bektas, T.: The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega **34**(3), 209–219 (2006). https://doi.org/10.1016/j.omega.2004.10.004
2. Bentley, J.L.: Experiments on traveling salesman heuristics. In: Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1990), USA, pp. 91–99. SIAM (1990)
3. Bose, J., Reiners, T., Steenken, D., Voss, S.: Vehicle dispatching at seaport container terminals using evolutionary algorithms. In: Proceedings of the 33rd Annual Hawaii International Conference on System Sciences (HICSS), USA, vol. 2, pp. 1–10. IEEE (2000). https://doi.org/10.1109/HICSS.2000.926669
4. Chen, J., Ling, F., Zhang, Y., You, T., Liu, Y., Du, X.: Coverage path planning of heterogeneous unmanned aerial vehicles based on ant colony system. Swarm Evol. Comput. **69**, 101005, 1–11 (2022). https://doi.org/10.1016/j.swevo.2021.101005
5. Fournier, S.M.R., Hülse, E.O., Pinheiro, É.V.: A*-guided heuristic for a multi-objective bus passenger Trip Planning Problem. Public Transp. **13**, 557–578 (2021). https://doi.org/10.1007/s12469-019-00204-1
6. Gavish, B., Srikanth, K.: An optimal solution method for large-scale multiple traveling salesmen problems. Oper. Res. **34**(5), 698–717 (1986). https://doi.org/10.1287/opre.34.5.698
7. Gawrilow, E., Köhler, E., Möhring, R.H., Stenzel, B.: Dynamic routing of automated guided vehicles in real-time. In: Krebs, H.-J., Jäger, W. (eds.) Mathematics - Key Technology for the Future, pp. 165–177. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-77203-3_12
8. Ghafurian, S., Javadian, N.: An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesmen problems. Appl. Soft Comput. **11**(1), 1256–1262 (2011). https://doi.org/10.1016/j.asoc.2010.03.002
9. Giardini, G., Kalmár-Nagy, T.: Genetic algorithm for combinatorial path planning: the subtour problem. Math. Probl. Eng. **2011**, 483643, 1–31 (2011). https://doi.org/10.1155/2011/483643
10. Giardini, G., Kalmar-Nagy, T.: Genetic algorithm for multi-agent space exploration. In: Proceedings of AIAA Infotech@Aerospace 2007 Conference and Exhibit 2824, USA, pp. 1–15. AIAA (2007). https://doi.org/10.2514/6.2007-2824
11. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company Inc., Boston (1989)
12. Harrath, Y., Salman, A.F., Alqaddoumi, A., Hasan, H., Radhi, A.: A novel hybrid approach for solving the multiple traveling salesmen problem. Arab J. Basic Appl. Sci. **26**(1), 103–112 (2019). https://doi.org/10.1080/25765299.2019.1565193
13. Herbawi, W., Weber, M.: The ridematching problem with time windows in dynamic ridesharing: a model and a genetic algorithm. In: Proceedings of 2012 IEEE Congress on Evolutionary Computation, USA, pp. 1–8. IEEE (2012). https://doi.org/10.1109/CEC.2012.6253001

14. Hou, M., Liu, D.: A novel method for solving the multiple traveling salesmen problem with multiple depots. Chin. Sci. Bull. **57**(15), 1886–1892 (2012). https://doi.org/10.1007/s11434-012-5162-7
15. Jiang, C., Wan, Z., Peng, Z.: A new efficient hybrid algorithm for large scale multiple traveling salesman problems. Expert Syst. Appl. **139**, 112867, 1–11 (2020). https://doi.org/10.1016/j.eswa.2019.112867
16. Kalmár-Nagy, T., Giardini, G., Bak, B.D.: The multiagent planning problem. Complexity **2017**, 3813912, 1–12 (2017). https://doi.org/10.1155/2017/3813912
17. Kara, I., Bektas, T.: Integer linear programming formulations of multiple salesman problems and its variations. Eur. J. Oper. Res. **174**(3), 1449–1458 (2006). https://doi.org/10.1016/j.ejor.2005.03.008
18. Karabulut, K., Öztop, H., Kandiller, L., Tasgetiren, M.F.: Modeling and optimization of multiple traveling salesmen problems: an evolution strategy approach. Comput. Oper. Res. **129**, 105192, 1–19 (2021). https://doi.org/10.1016/j.cor.2020.105192
19. Király, A., Abonyi, J.: Redesign of the supply of mobile mechanics based on a novel genetic optimization algorithm using Google Maps API. Eng. Appl. Artif. Intell. **38**, 122–130 (2015). https://doi.org/10.1016/j.engappai.2014.10.015
20. Laporte, G., Nobert, Y.: A cutting planes algorithm for the *m*-salesmen problem. J. Oper. Res. Soc. **31**(11), 1017–1023 (1980). https://doi.org/10.1057/jors.1980.188
21. Redmond, M., Campbell, A.M., Ehmke, J.F.: Data-driven planning of reliable itineraries in multi-modal transit networks. Public Transp. **12**, 171–205 (2020). https://doi.org/10.1007/s12469-019-00221-0
22. Ryan, J.L., Bailey, T.G., Moore, J.T., Carlton, W.B.: Reactive Tabu Search in unmanned aerial reconnaissance simulations. In: Proceedings of 1998 Winter Simulation Conference (Cat. No. 98CH36274), USA, vol. 1, pp. 873–879. IEEE (1998). https://doi.org/10.1109/WSC.1998.745084
23. Somhom, S., Modares, A., Enkawa, T.: Competition-based neural network for the multiple travelling salesmen problem with minmax objective. Comput. Oper. Res. **26**(4), 395–407 (1999). https://doi.org/10.1016/S0305-0548(98)00069-0
24. Song, C.-H., Lee, K., Lee, W.D.: Extended simulated annealing for augmented TSP and multi-salesmen TSP. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN), USA, vol. 3, pp. 2340–2343. IEEE (2003). https://doi.org/10.1109/IJCNN.2003.1223777
25. Tong, Y., Zeng, Y., Zhou, Z., Chen, L., Ye, J., Xu, K.: A unified approach to route planning for shared mobility. Proc. VLDB Endow. **11**(11), 1633–1646 (2018). https://doi.org/10.14778/3236187.3236211
26. Torabbeigi, M., Lim, G.J., Kim, S.J.: Drone delivery scheduling optimization considering payload-induced battery consumption rates. J. Intell. Robot. Syst. **97**(3), 471–487 (2020). https://doi.org/10.1007/s10846-019-01034-w
27. Wang, X., Liu, D., Hou, M.: A novel method for multiple depot and open paths, multiple traveling salesmen problem. In: Proceedings of 2013 IEEE 11th International Symposium on Applied Machine Intelligence and Informatics (SAMI), USA, pp. 187–192. IEEE (2013). https://doi.org/10.1109/SAMI.2013.6480972
28. Yu, H., Meier, K., Argyle, M., Beard, R.W.: Cooperative path planning for target tracking in urban environments using unmanned air and ground vehicles. IEEE/ASME Trans. Mechatron. **20**(2), 541–552 (2015). https://doi.org/10.1109/TMECH.2014.2301459

29. Yuan, S., Skinner, B., Huang, S., Liu, D.: A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. Eur. J. Oper. Res. **228**(1), 72–82 (2013). https://doi.org/10.1016/j.ejor.2013.01.043
30. Zhou, H., Song, M., Pedrycz, W.: A comparative study of improved GA and PSO in solving multiple traveling salesmen problem. Appl. Soft Comput. **64**, 564–580 (2018). https://doi.org/10.1016/j.asoc.2017.12.031