Applied Soft Computing Journal 185 (2025) 113959

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Cooperative driving route planning based on asymmetric multi-agent path planning problem with limited service area constraints

Ali Maktabifard*, Dávid Földes

Budapest University of Technology and Economics, Faculty of Transportation Engineering and Vehicle Engineering, Department of Transport Technology and Economics, Budapest, Hungary

HIGHLIGHTS

- A cooperative driving route planning method for vehicle fleets is presented.
- Asymmetric Multi-Agent Path Planning problem is used to model the routing problem.
- Limited service area constraints are applied to each agent (vehicle).
- A Genetic Algorithm with Prioritized and Sorted Greedy initialization is applied.
- Limited service area constraints improve the performance of the developed solution.

ARTICLE INFO

Keywords:
Multi-agent planning problem
Combinatorial optimization
Genetic algorithm
Metaheuristic
Routing problem
Asymmetric MTSP
Limited service area constraints

ABSTRACT

Mobility and logistics service providers operating a vehicle fleet and serving several targets with each vehicle are faced with the challenge of planning efficient and cooperative driving routes while considering all the vehicles. To address this challenge, we present a novel driving route planning method based on an Asymmetric Multi-Agent Path Planning (AMAPP) problem, a variation of the classical Multiple Traveling Salesmen Problem (MTSP). Given a team of m agents (vehicles) that must visit n targets located in a real road network, a set of m optimized open paths (the *Plan*) must be found such that each target is visited exactly once. The optimization objective is to minimize the driving distance of the path with the longest driving distance in the Plan (a Min-Max problem) so that this cooperative operation can be completed in the least amount of time. In several cases, the service area for each agent is limited (e.g., certain districts in a city). To simulate this real-world condition, two additional constraints are applied: the maximum geodetic range for each agent, and the maximum spatial range of targets for each agent. An easy-to-apply Genetic Algorithm (GA) with two novel initialization methods is presented to solve this route planning problem. In order to validate the developed route planning method and demonstrate its applicability, the method was tested in a real-world test case where it showed a decent performance. The results show that the applied limited service area constraints not only decrease the average driving distance of the longest route in the route plan, but also reduce the average runtime of the developed solution. Additionally, the performance of the proposed GA was benchmarked using 26 problems based on the TSPLIB instances, where the proposed GA achieved new Best Known Solution (BKS) values for 19 benchmark problems and a result equal to the BKS value for another two problems, demonstrating its superiority and robustness. The developed method can be used for route planning of the mobility and logistics services requiring multiple destinations to be reached by a fleet of vehicles, such as group ride-sharing and last-mile delivery.

1. Introduction

Efficient and cooperative path planning for vehicle fleets has shown its importance for a variety of applications such as logistics and delivery services [1,2], ride-sharing services [3,4], bus fleets [5,6], and autonomous/unmanned vehicle fleets [7,8]. An efficient multi-agent planner is essential for cooperative path planning of vehicle fleets. This planning problem can be defined as finding a set of optimized paths that

* Corresponding author.
Email addresses: a.maktabifard@edu.bme.hu (A. Maktabifard), foldes.david@kjk.bme.hu (D. Földes).

https://doi.org/10.1016/j.asoc.2025.113959
Received 15 May 2024; Received in revised form 18 August 2025; Accepted 17 September 2025
Available online 19 September 2025

allows a team of agents to visit a set of given targets. The optimization objective of the problem can vary depending on which parameter of the corresponding system is aimed to be optimized. For instance, minimizing the total traveling distance/time/cost of all paths (Min-Sum), maximizing the total benefit of all paths (Max-Sum), or minimizing the traveling distance/time/cost of the longest path (Min-Max). This problem is similar to the classical Multiple Traveling Salesmen Problem (MTSP) [9], and its variants such as the Multi-Agent Path Planning (MAPP) problem [10]. The general definition of the classical MTSP is as follows. Given n targets (e.g., cities) and a team of m agents (e.g., salesmen) initially located at a single depot, find m tours (closed paths) for the m agents starting and ending at the depot such that each target is visited exactly once and the total cost of visiting all targets is minimized (a Min-Sum problem). Accordingly, the problem includes $n+1$ nodes (n target nodes and one depot node). The cost metric can be defined in terms of any metric of distance, travel time, transportation cost, etc., [11]. A $k \times k$ cost matrix ($k = n + 1$) is associated with the problem in which element ij is the cost of traversing from node i to node j in terms of the defined cost metric. This matrix is symmetric if the symmetry relation element ij = element ji holds true for each pair of nodes i and j; otherwise, the cost matrix is asymmetric, and the problem is referred to as an Asymmetric MTSP (AMTSP).

The Asymmetric MAPP (AMAPP) problem is simply an MTSP with the following main features: asymmetric cost matrix, multiple depots, Min-Max optimization objective, and open paths. This implies that in the AMAPP problem:

- Each agent starts and ends its path at two distinct positions (open paths).
- The optimization objective is to minimize the cost of the longest path (a Min-Max problem).
- Agents can have different initial positions (multiple depots).
- The problem includes $m + n$ nodes (n target nodes and m nodes representing the initial positions of m agents).
- The cost of traversing between two nodes is not necessarily the same in both directions (asymmetric cost matrix).

More practical situations can be modeled by applying additional restrictions to these combinatorial optimization problems. These restrictions affect the agents mostly by limiting their path length, workload, or operational range [2,7,12]. As a novelty, in this study the cooperative path planning for vehicle fleets is modeled as a so-called Multi-Agent Driving Route Planning (MADRP) problem with limited service area constraints. The MADRP problem is essentially an AMAPP problem with the following terms:

- The agents and targets are located in the real road network, with their locations specified by geographical coordinates.
- The cost metric of the problem is driving distance.

In this study, we calculate the driving distance using the OpenRouteService API (Application Programming Interface). To simulate the situations where the service area for each agent is limited, we apply two additional constraints on the MADRP problem: (1) The maximum geodetic range (GeoRange) for each agent, and (2) The maximum spatial range of targets for each agent. In the MADRP problem, m driving routes must be found for m agents to perform the task of visiting n targets cooperatively. The time required to complete this task by the agents is determined by the travel time of the agent having the longest route to drive because here it is assumed that the travel time of a driving route only depends on its distance (as a limitation, traffic and other factors impacting the travel time are not considered in this study). Thus, the completion time for this task is minimized if the distance of the longest driving route is minimized. Based on these assumptions, we can simply say the MADRP problem is a Min-Max problem in which a team of m agents must visit n targets in the minimum amount of time, and every target is visited exactly once.

This paper presents a solution method that utilizes a Genetic Algorithm (GA) to solve the MADRP problem. Similar to a classical GA, the GA presented here consists of two phases: initialization and evolution. An initial solution is generated in the initialization phase and then the evolution phase evolves the initial solution towards a near-optimal solution. This GA benefits from two novel initialization methods which enhance its performance. Note that the additional constraints are applied in both the initialization phase and the evolution phase.

The aim of this research is to develop a practical route planning method that is able to:

- Enhance the efficiency of route planning for a fleet of vehicles when the aim is to optimize open paths in a cooperative manner (most of the related literature instances focus on closed paths). This is useful for mobility and logistics services operating a vehicle fleet where the vehicles do not return to their initial locations after they complete their tasks, or when the aim is to optimize the routes only until all customers are served and the return of the vehicles to their initial locations does not affect the customers.
- Consider the asymmetry of driving routes in real road networks.
- Minimize the driving distance of the longest route in the route plan provided for a fleet of vehicles so that the fleet can visit a given number of target locations in the least amount of time (assuming the time that the fleet needs to complete this cooperative operation is directly proportional to the driving distance of the longest route in the route plan).
- Simulate the situations where the service area for each agent (vehicle) is limited due to various factors such as planning strategy, pricing strategy, or the preference of each agent itself. These situations have not received enough attention in the literature.

The remainder of the paper is organized as follows. Section 2 reviews the related literature. The AMAPP and MADRP problems are described in detail in Section 3. The methodology of the proposed solution is described in Section 4; then, the results are presented and discussed in Section 5. Finally, the conclusions are drawn in Section 6.

2. Literature review

Regardless of being symmetric or asymmetric, the MTSP and the MAPP problems are notoriously difficult to solve because of their complex combinatorial nature (NP-hardness). Most of the approaches used in the literature to tackle the MTSP can be categorized into two general types: *exact* and *heuristic-based* approaches.

The exact approaches solve the problem to optimality. However, in general, they are highly time-consuming owing to the complexity of their calculations [9]. The exact approaches apply deterministic methods, but these methods usually can be applied after either the transformation of the MTSP to an equivalent Traveling Salesman Problem (TSP), or relaxing some constraints of the problem [9,11]. The deterministic methods such as Integer Programming (IP) [13–16], Branch-and-Bound (BB) [17], Branch-and-Cut (BC) [18], and Cutting Planes (CP) [19] have been proposed in the literature to tackle the MTSP. The exact approaches are limited to the MTSPs with reasonable sizes because for large-scale problems, not only their results can be degenerate [11], but also their solution runtime increases significantly [9,20]. In general, the largest MTSPs solved by adopting exact approaches have up to 100 and 500 cities (targets) for the symmetric and asymmetric problems, respectively (assuming a limited number of salesmen (agents)) [11]. It should be also noted that transforming the MTSP to an equivalent TSP might result in an even more difficult problem to solve, especially using the exact approaches [15,19].

The heuristic-based approaches solve the problem by applying approximate heuristic methods such as Genetic Algorithm (GA) [2,5, 21–26], Ant Colony Optimization (ACO) [20,27–29], Artificial Bee Colony (ABC) [30,31], Particle Swarm Optimization (PSO) [21,32], Neural Network (NN) [33,34], Invasive Weed Optimization (IWO)

[30], Tabu Search (TS) [7,33,35], Grey Wolf Optimization (GWO) [36], Simulated Annealing (SA) [8,35,37], Evolution Strategy (ES) [38], Relative Distances Approach (RDA) [39], Graph Simplification Algorithm (GSA) [40], or Colored Petri Nets (CPN) [41]. The heuristic-based approaches can achieve near-optimal solutions in a reasonable amount of time even for larger problems [20].

As mentioned in the Introduction, the asymmetric MAPP problem is basically an MTSP with the following main features: asymmetric cost matrix, multiple depots, Min-Max optimization objective, and open paths. Table 1 summarizes several literature instances solving an MTSP with some of these features. In addition to these studies on the MTSP, there are a few studies specifically addressing the MAPP problem as the foundation of the introduced MADRP problem. In [42], a GA was introduced to solve the symmetric MAPP problem. This method was developed further in [10] to tackle the symmetric MAPP problem where the minimum number of targets for each agent was applied as an additional constraint for one of the studied test problems. A similar GA was presented in [12] for solving the symmetric MAPP problem with two additional constraints: (1) The maximum number of targets for each agent, and (2) The maximum Euclidean range for each agent. This range restricts the service area of each agent to a circular area in a 2D Euclidean space and proved to be effective in reducing the average final cost achieved for a variety of randomly generated test cases. In the current study, the solution algorithm presented for the MADRP problem is based on a modified version of the GAs proposed in [10,12]. The novelty of this modified GA is that it employs two newly developed initialization methods, the Prioritized Greedy and the Sorted Greedy, in addition to employing the Greedy initialization method that was previously used for the GA proposed in [12]. Note that the Greedy initialization methods for the GAs presented in [10,42] have different structures compared to the one proposed in [12].

Moreover, several studies presented a practical and cooperative route planning method by combining the MTSP and a mapping (routing) service API. In [2], a route planning method was developed using the Google Maps API and the MTSP as the mathematical model for the routing problem solved by a GA. The Google Maps API was used for distance matrix calculation and visualizing the optimized routes. The considered MTSP is asymmetric, single-depot, closed-path with time windows and Min-Sum optimization objective. For the application study, this method was used to optimize the route plan for a fleet of supply trucks (the agents), where the following additional constraints were also applied: (1) The maximum traveling time and distance for each agent, and (2) The maximum number of agents that can be deployed (number of agents is an optimization variable to be minimized alongside the minimization of total traveling time or distance for all agents). The route planning method proposed in [35] also has a similar structure but instead of GA, it uses SA and TS algorithms to solve the routing problem, and it considers multiple periods during which the time windows are applied. For the application test, this method was used to optimize the route plan for a team of university representatives visiting a number of exam locations. A similar route planning method but without time windows was presented in [5], which uses a GA to solve the routing problem. This method was used to optimize the route plan for a fleet of school buses.

Concluding the literature review, to the best of our knowledge, no other study has presented a route planning method based on an AMAPP problem with the limited service area constraints applied in this study. Although the maximum GeoRange for each agent in this study has been inspired by the maximum Euclidean range for each agent applied in

Table 1
Summary of literature instances solving an MTSP with some features similar to the AMAPP problem.

Ref.	Features of the solved MTSP(s)				Additional constraint(s)*		Method of developed solution(s)**
	Cost matrix	Depot(s)	Opt. objective	Paths	Min. and/or Max. no. of targets for each agent	Other constraint(s)	
[13]	Asymmetric	Multiple	Min-MaxMin-Sum	Open	Min.		IP
[14]	AsymmetricSymmetric	Multiple	Min-Sum	Open		Not all agents are needed to be used	IP
[15]	Asymmetric	MultipleSingle	Min-Sum	Closed, Openand closed	Min., Max.		IP
[16]	Asymmetric	Multiple	Min-Sum	Closed	Min.		IP
[18]	AsymmetricSymmetric	Multiple	Min-Sum	Closed			BC
[21]	Symmetric	Multiple	Min-Sum	Closed	Min.		GA, PSO
[22]	Symmetric	Single	Min-Sum	Open	Max.		GA
[24]	AsymmetricSymmetric	Single	Min-Sum	Closed	Max.		GA
[25]	Asymmetric	Single	Min-Sum	Closed	Max.		GA
[26]	Symmetric	Multiple	Min-MaxMin-Sum	Open			GA
[27]	Symmetric	MultipleSingle	Min-MaxMin-Sum	Closed			ACO
[28]	Asymmetric	Multiple	Max-Sum	Closed	Min., Max.		ACO
[29]	Asymmetric	Single	Min-MaxMin-Sum	Closed		Constraint on the workload imbalancement of the agents	ACO
[30]	Symmetric	Single	Min-MaxMin-Sum	Closed			ABC, IWO
[32]	Symmetric	Single	Min-MaxMin-Sum	Closed			PSO
[33]	Symmetric	Single	Min-Max	Closed			NN, TS
[36]	Symmetric	Single	Min-Max	Open			GWO
[37]	Symmetric	Multiple	Min-Sum	Closed	Min., Max.		SA
[38]	AsymmetricSymmetric	MultipleSingle	Min-MaxMin-Sum	Closed			ES
[39]	AsymmetricSymmetric	MultipleSingle	Min-MaxMin-Sum	Open			RDA
[40]	Symmetric	Multiple	Min-Sum	Open			GSA

* The following were not considered as additional constraints: **(1)** The minimum number of targets for each agent equals 1, and **(2)** The maximum number of targets for each agent equals the number of all available targets.

** IP: Integer Programming, BC: Branch-and-Cut, GA: Genetic Algorithm, PSO: Particle Swarm Optimization, ACO: Ant Colony Optimization, ABC: Artificial Bee Colony, IWO: Invasive Weed Optimization, NN: Neural Network, TS: Tabu Search, GWO: Grey Wolf Optimization, SA: Simulated Annealing, ES: Evolution Strategy, RDA: Relative Distances Approach, GSA: Graph Simplification Algorithm.

[12], the latter range can only be applied to the routing problems in a Euclidean space not to a real-world routing problem. Furthermore, in this study, the maximum spatial range of targets for each agent is applied for the first time in the literature. Additionally, the Prioritized Greedy and the Sorted Greedy initialization methods for the GA presented in this study are novel compared to the initialization methods proposed for the similar GAs in the literature.

3. Multi-agent driving route planning problem: general terms and notation

In this section, the necessary foundation from Graph theory is presented, and the notation of the introduced MADRP problem is derived from the notation of the AMAPP problem.

In Graph theory [43,44], a *digraph* (*directed graph*) D is defined as an ordered pair $D = (V, A)$, where $V = \{v_1, \dots, v_n\}$ is a set of n *vertices* (*nodes*) and $A = \{(v_i, v_j) \mid v_i, v_j \in V;\ i \neq j\}$ is a set of *arcs* (*directed edges*). Each arc is an ordered pair of distinct vertices from V. Thus, the arcs are directed (i.e., $(v_i, v_j) \neq (v_j, v_i)$). Arcs (v_i, v_j) and (v_j, v_i) are *opposite* arcs. For an arc (v_i, v_j), the first vertex v_i is its *tail* and the second vertex v_j is its *head*. Accordingly, an arc (v_i, v_j) *leaves* vertex v_i and *enters* vertex v_j. Each arc represents a one-way connection from its tail to its head. A *complete digraph* is a digraph in which every two distinct vertices v_i and v_j are connected by both arcs (v_i, v_j) and (v_j, v_i). In this paper, a complete digraph is denoted by $\mathbb{K}_n(V)$, where n is the number of vertices in the vertex set V. The union of m digraphs $D_1 = (V_1, A_1), \dots, D_m = (V_m, A_m)$ is denoted by $\bigcup_{i=1}^{m} D_i = (\bigcup_{i=1}^{m} V_i, \bigcup_{i=1}^{m} A_i)$ and it is a digraph that consists of all vertices and edges from digraphs $D_1, \dots, D_m$.

In a digraph, a *path* p is a sequence of arcs having the property that for each two consecutive arcs in the sequence, the head of the first arc and the tail of the second arc are the same vertex. A path visits a sequence of k distinct vertices by traversing $k-1$ arcs between those vertices; therefore none of those vertices is visited more than once. For instance, $p_1 = \langle (v_1, v_2), (v_2, v_3), (v_3, v_4) \rangle$ is the path that visits the vertex sequence $\langle v_1, v_2, v_3, v_4 \rangle$. The same vertices ($v_1$, v_2, v_3 and v_4) can be visited in a different order by another path. For example, path $p_2 = \langle (v_3, v_1), (v_1, v_4), (v_4, v_2) \rangle$ visits the vertex sequence $\langle v_3, v_1, v_4, v_2 \rangle$. A path p exists in a digraph $D = (V, A)$ if every arc in p belongs to the arc set A. Sometimes paths are simply represented by the vertex sequence they visit, for instance, $p_1 = \langle v_1, v_2, v_3, v_4 \rangle$ and $p_2 = \langle v_3, v_1, v_4, v_2 \rangle$. This kind of path representation is used in the next section. A path can be perceived as an *open route* visiting every vertex on its way exactly once and ending at a vertex different from its starting vertex. The *length* of a path is the number of its arcs. The set of all paths of length k in a digraph D is denoted by $\boldsymbol{P}_k(D)$. Two paths are *vertex-disjoint* if there is no vertex that is visited by both of them.

A *weighted digraph* is a digraph in which each arc (v_i, v_j) is assigned a *weight* $w(v_i, v_j)$. In a weighted digraph D, if the weight symmetry relation $w(v_i, v_j) = w(v_j, v_i)$ holds true for each pair of opposite edges (v_i, v_j) and (v_j, v_i), D is *symmetrically* weighted; otherwise, D is an *asymmetrically* weighted digraph. In the digraph representation of the MADRP problem in this paper, the weight assigned to each arc (v_i, v_j) is the distance of the shortest driving route in the real road network that starts and ends at two geographic locations represented by vertices v_i and v_j, respectively. In this paper, $\boldsymbol{d}_{sd}(v_i, v_j)$ denotes this so-called *shortest driving distance* assigned as the weight of an arc (v_i, v_j) and we calculate it using the open-source routing API provided by OpenRouteService. The shortest driving route between two geographic locations may differ depending on the direction of travel, and therefore its distance is not necessarily the same in both directions. Accordingly, $\boldsymbol{d}_{sd}(v_i, v_j)$ and $\boldsymbol{d}_{sd}(v_j, v_i)$ are not necessarily equal.

In a weighted digraph D, the *cost* of a path $p \in \boldsymbol{P}_k(D)$ is the sum of weights of the arcs constituting the path p and it is denoted by $c(p)$:

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

We formulate the AMAPP and the MADRP problems using the fundamental principles from Graph theory outlined. Let $\mathbb{A} = \{a_1, \dots, a_m\}$ be the set of m vertices representing the locations at which m agents embark on their journeys, and consider $T = \{t_1, \dots, t_n\}$ to be the set of n vertices representing n target locations, with $n \geq m$. The AMAPP problem can be formulated as follows. For the i^{th} agent ($i = 1, \dots, m$), the configuration space of the problem is the complete digraph $\mathbb{K}_{n+1}(V_i)$, with the vertex set $V_i = T \cup a_i$. Each arc in $\mathbb{K}_{n+1}(V_i)$ is assigned a predetermined weight. The weights are such that for each two opposite arcs, $w(v_x, v_y)$ and $w(v_y, v_x)$ are not necessarily equal (v_x and v_y symbolize two vertices in V_i). This renders $\mathbb{K}_{n+1}(V_i)$ an asymmetrically weighted digraph indicating the asymmetry of the problem. For all m agents together, the configuration space of the problem is digraph $\mathbb{D} = \bigcup_{i=1}^{m} \mathbb{K}_{n+1}(V_i)$, which is the union of m corresponding digraphs $\mathbb{K}_{n+1}(V_i)$. Let p_i denote a path of length $k_i > 0$ in $\mathbb{D}$ that starts at vertex a_i, with $1 \leq i \leq m$. Consider $\boldsymbol{P} = \{p_1, \dots, p_m\}$ to be a set of m pairwise vertex-disjoint paths p_i in $\mathbb{D}$. $\boldsymbol{P}$ is an optimal solution for the AMAPP problem if it satisfies two general conditions:

I. $\sum_{i=1}^{m} k_i = n$ (this implies every vertex in T is visited exactly once).
II. The cost of the path with the largest cost in $\boldsymbol{P}$ Eq. (2) is minimized (a Min-Max problem).

$$c_m(\boldsymbol{P}) = \max_{i=1}^{m} c(p_i) \tag{2}$$

If $\boldsymbol{P}$ only satisfies condition I. ($\sum_{i=1}^{m} k_i = n$), it is a solution for the AMAPP problem but not an optimal solution. Therefore, the objective of the problem is to find a set $\boldsymbol{P}$ of m pairwise vertex-disjoint paths p_i in $\mathbb{D}$ such that both conditions I. and II. are met. A cost metric must be chosen for the problem. The cost metric can be defined in terms of any metric of distance, travel time, transportation cost, etc. The weights of arcs in $\mathbb{K}_{n+1}(V_i)$ are determined using the desired cost metric. For example, when cycling distance is the cost metric of the problem, the weight of each arc in $\mathbb{K}_{n+1}(V_i)$ is the distance of the shortest cycling route from the location represented by the tail of the arc to the location represented by the head of the arc.

In this paper, the MADRP problem is defined as an AMAPP problem with the following terms:

- The agents and targets are located in the real road network, with their locations specified by geographical coordinates.
- The cost metric of the problem is driving distance. Accordingly, each arc in $\mathbb{K}_{n+1}(V_i)$ is assigned a weight equal to the shortest driving distance from the location represented by the tail of the arc to the location represented by the head of the arc, that is, $w(v_x, v_y) = \boldsymbol{d}_{sd}(v_x, v_y)$, where v_x and v_y symbolize two vertices in V_i. Note that $\boldsymbol{d}_{sd}(v_x, v_y)$ and $\boldsymbol{d}_{sd}(v_y, v_x)$ are not necessarily equal, so $\mathbb{K}_{n+1}(V_i)$ is asymmetrically weighted indicating the asymmetry of the problem.

In the MADRP problem, the paths in $\boldsymbol{P}$ represent m driving routes for m agents to perform the task of visiting n targets cooperatively. The time required to complete this task by the agents is determined by the travel time of the agent having the longest route to drive because here it is assumed that the travel time of a driving route only depends on its distance (as a limitation, traffic and other factors impacting the travel time are not considered in this study). Thus, the completion time for this task is minimized if the distance of the longest driving route is minimized. Based on these assumptions, we can simply say the MADRP problem is a Min-Max problem in which a team of m agents must visit n targets in the minimum amount of time, and every target is visited exactly once (this also implies visitation only by one agent). The whole team of agents must be used, so each agent should visit at least one target ($k_i \geq 1$). However, the number of targets visited by each agent can be different.

Moreover, we apply two additional constraints to the MADRP problem to simulate the situations where the service area for each agent is limited. These situations can stem from various factors such as planning

strategy, pricing strategy, or the preference of each agent itself. These two constraints are applied independently of each other. The additional constraints are defined as follows:

I. The first additional constraint is the maximum geodetic range (GeoRange) for each agent. Under this constraint, a target can be assigned to an agent if the target is located within a limited and circular geographic area (geodetic circle) centered at the starting location of the agent. For each agent, the radius of this geodetic circle is given and we call it the maximum geodetic range of the agent. Let $d_g(v_x, v_y)$ denote the geodetic distance (based on WGS84 reference ellipsoid) between two geographic locations represented by vertices v_x and v_y. Consider $R = \{r_1, \ldots, r_m\}$ to be the set of the maximum geodetic ranges for m agents, with r_i being the maximum geodetic range for the ith agent. Under this constraint, the jth target can be assigned to the ith agent if: $d_g(a_i, t_j) \leq r_i$. By applying this additional constraint, we can simulate the situations where each agent can only visit the targets that are located within a limited area around the starting location of the agent. When this constraint is applied, if there is at least one target that is not within the maximum geodetic range of any agent ($\exists\, t_j \in T \mid \forall\, i \in \{1, \ldots, m\},\ d_g(a_i, t_j) > r_i$), the fleet of m agents cannot visit all n targets. This situation is hereafter referred to as the *out-of-GeoRange issue*. In this study, for solving the problem with the first additional constraint, we assumed that the value of the maximum geodetic range is the same for every agent ($\forall\, r_i = r$). This models a fleet of agents (vehicles) that is homogeneous in terms of the maximum geodetic range for each agent.

II. The second additional constraint is the maximum spatial range of targets for each agent. Under this constraint, the j^{th} target can be assigned to the i^{th} agent if $\boldsymbol{d}_{sd}(a_i, t_j)$ is less than or equal to a certain value. This value is given separately for each agent and we call it the maximum spatial range of targets for the agent. Let $S = \{s_1, \ldots, s_m\}$ be the set of the maximum spatial ranges of targets for m agents, where s_i is the maximum spatial range of targets for the i^{th} agent. Under this constraint, the j^{th} target can be assigned to the i^{th} agent if: $\boldsymbol{d}_{sd}(a_i, t_j) \leq s_i$. By applying this additional constraint, we can simulate the situations where each agent can only visit the targets that are not farther than a certain driving distance from the starting location of the agent. When this constraint is applied, if there is at least one target that is not in the maximum spatial range of targets for any agent ($\exists\, t_j \in T \mid \forall\, i \in \{1, \ldots, m\},\ \boldsymbol{d}_{sd}(a_i, t_j) > s_i$), the fleet of m agents cannot visit all n targets. This situation is henceforth referred to as the *out-of-SpatialRange issue*. In this study, for solving the problem with the second additional constraint, we assumed that the value of the maximum spatial range of targets for each agent is the same for all agents ($\forall\, s_i = s$). This models a fleet of agents (vehicles) that is homogeneous in terms of the maximum spatial range of targets for each agent.

4. Methodology of the approximate solution

In this study, we present a solution method that utilizes a GA to solve the MADRP problem. This GA is partly based on the GAs proposed in [10,12], which demonstrated decent performance in solving the symmetric MAPP problem. These GAs provide an easy-to-apply and effective methodology for solving the family of MAPP problems, including the MADRP problem investigated in this paper. In brief, a GA is a metaheuristic that is used to find approximate solutions to optimization and search problems [45]. Genetic Algorithms belong to the broader class of evolutionary algorithms, and employ techniques inspired by evolutionary biology and Darwin's theory of evolution, such as selection, crossover, mutation, and inheritance. In these algorithms, a population of solutions competes and only the best one(s) in terms of a certain objective function survive.

The proposed GA closely resembles the (1 + 1) Evolution Strategy approach [46,47]. Like a classical GA, the GA presented here consists of two phases: initialization and evolution. An initial solution is generated in the initialization phase and then the evolution phase evolves the initial solution towards a near-optimal solution. An initial solution is usually a non-optimal solution that is obtained using simple algorithms. As described in Section 3, a solution for the MADRP problem is a set of pairwise vertex-disjoint paths visiting every target exactly once. A solution is called a *Plan* ($\boldsymbol{P}$) in this context. A Plan is optimal if the longest path in the Plan (in terms of driving distance) is the shortest possible (a Min-Max problem). The solution method can be summarized as follows:

- Initialization phase: a Plan $\boldsymbol{P}$ is generated as the initial Plan.
- Evolution phase: the Plan $\boldsymbol{P}$ is evolved towards a near-optimal Plan by minimizing the cost $c_m(\boldsymbol{P})$ see Eq. (2).

The applied distance metric is the shortest driving distance ($\boldsymbol{d}_{sd}$) that is obtained from a *distance matrix* calculated using the OpenRouteService API. The geographical coordinates of n target locations and m starting locations for m agents are used as the input for the distance matrix module of the API to calculate a $k \times k$ distance matrix, where $k = m + n$. In order to calculate the distance matrix containing the shortest driving distance between each pair of the input coordinates, the transportation mode profile in the API module must be set to driving-car. In this distance matrix, element ij is the distance of the shortest driving route from the input location i to the input location j. The OpenRouteService API uses geographic data from Open Street Map.

4.1. Initialization phase

In this phase, the initial Plan $\boldsymbol{P} = \{p_1, \ldots, p_m\}$ (a starting set of paths) is generated. Let $\mathbb{A} = \{a_1, \ldots, a_m\}$ be the set of m locations at which m agents embark on their journeys, and consider $T = \{t_1, \ldots, t_n\}$ to be the set of n target locations, with $n \geq m$. A Plan is valid if each target appears in only one path $p_i \in \boldsymbol{P}$, which also implies that each target can be assigned to only one agent. Different initial Plans can be generated using various initialization methods. An initialization method is basically an iterative algorithm for selecting and assigning the targets to the agents. In this paper, the algorithm of each initialization method is composed of a loop inside an outer loop. For the ease of referring to iterations of each loop, each iteration of the outer loop is called a *planning round*, while each iteration of the inner loop is simply called an iteration. In this study, maximum one target is assigned to each agent in every planning round of an initialization method. Additionally, the *reference location* of each agent is defined as the location of the last target assigned to the agent. Note that for the first planning round, since the agents have not received any targets yet, the reference location of each agent is the starting location of the agent (a_i). A valid Plan $\boldsymbol{P} = \{p_1, \ldots, p_m\}$ is generated by iterating an initialization method until all targets are assigned to the agents. Three different initialization methods were utilized in this study:

I. *Greedy* initialization method:
   The summarized algorithm of the method is shown in Algorithm 1. In every planning round (iteration of the while loop in Algorithm 1), the order of target assignment is as follows: agent 1, agent 2, …, agent m (a target is assigned to agent 1, then another target is assigned to agent 2 and so on). In each planning round, the nearest unassigned target to the reference location of each agent is selected and assigned to the agent. This algorithm terminates when all targets are assigned to the agents. This initialization method was introduced in [12].

II. *Prioritized Greedy* initialization method:
   The simplified algorithm of the method is shown in Algorithm 2. This method also uses the nearest neighbor heuristic to select the targets. However, it applies a dynamic order for target assignment unlike the static order used in Greedy initialization method. This dynamic order is applied using an agent selection

Algorithm 1 Greedy initialization method.

Input: $\mathbb{A} = \{a_1, \dots, a_m\}$, $T = \{t_1, \dots, t_n\}$
Output: $\boldsymbol{P} = \{p_1, \dots, p_m\}$

$T' = \{t'_1, \dots, t'_n\} \leftarrow T = \{t_1, \dots, t_n\}$ ▷ $t'_1 \leftarrow t_1, \dots, t'_n \leftarrow t_n$
$\mathbb{A}' = \{a'_1, \dots, a'_m\} \leftarrow \mathbb{A} = \{a_1, \dots, a_m\}$ ▷ $a'_1 \leftarrow a_1, \dots, a'_m \leftarrow a_m$
$\boldsymbol{P} \leftarrow \{p_1 \leftarrow [a_1], \dots, p_m \leftarrow [a_m]\}$ ▷ $\boldsymbol{P}$ *is a set of* m *arrays*
while T' is not empty **do**
  for $i \leftarrow 1, \dots, m$ **do**
    find the nearest target t'_x to reference location a'_i ▷ $t'_x \in T'$, $1 \le x \le n$
    append target t'_x to the end of p_i ▷ *Target assignment,* $p_i \in \boldsymbol{P}$
    $a'_i \leftarrow t'_x$ ▷ *Updating* $a'_i \in \mathbb{A}'$
    $T' \leftarrow T' - t'_x$ ▷ *Updating* T'
    if T' is empty **then**
      return $\boldsymbol{P}$
    end if
  end for
end while

algorithm. This algorithm determines which agent receives a target in each iteration. Given m agents and n targets, the method performs $\left\lceil \frac{n}{m} \right\rceil$ planning rounds (iterations of the outer while loop in Algorithm 2). In each round, m iterations are executed unless the method terminates (no target remains unassigned). In every iteration, at first an agent is selected by the agent selection algorithm to receive a target. This algorithm selects the agent for which the distance of the nearest unassigned target to the reference location of the agent is minimum. However, the agent selection algorithm is not applied to all m agents in every iteration (not all m agents are considered for the agent selection process in every iteration). Let $\mathcal{I} = [1, \dots, m]$ be the array of m agents' indices. In the first iteration in each planning round, the algorithm selects an agent $i_1 \in \mathcal{I}_1 = \mathcal{I}$. Then in the second iteration, the algorithm selects an agent $i_2 \in \mathcal{I}_2 = \mathcal{I}_1 - i_1$. Eventually, in the m^{th} iteration, only one agent remains to be selected ($i_m \in \mathcal{I}_m = \mathcal{I}_{m-1} - i_{m-1}$). In each iteration, after the desired agent is selected, the nearest unassigned target to the reference location of the agent is selected and assigned to the agent. The method terminates when all targets are assigned to the agents.

III. *Sorted Greedy* initialization method:

The simplified algorithm of the method is shown in Algorithm 3. The main algorithm of this method can start only if one target has already been assigned to each agent. Hence, at first one target is assigned to each agent using the Prioritized Greedy method as the starter of the algorithm. After that, at the beginning of each planning round (iteration of the while loop in Algorithm 3), the array of agents' indices $\mathcal{I} = [1, \dots, m]$ is sorted based on the distance of each agent's current path p_i in ascending order to obtain $\mathcal{I}_{sorted}$ (each current path p_i starts at location a_i and visits the targets that have been assigned to agent i until the beginning of the given planning round, with $i \in \mathcal{I}$). Consequently, the first index in $\mathcal{I}_{sorted}$ is the index of the agent with the smallest distance of its current path p_i and so on in ascending order to sort the rest of agents' indices. In each planning round, after $\mathcal{I}_{sorted}$ is obtained, the order of agents' indices in $\mathcal{I}_{sorted}$ is used as the order of target assignment in the planning round where the nearest unassigned target to the reference location of each agent is selected and assigned to the agent. The method terminates when all targets are assigned to the agents.

Nine different solution modes were analyzed in this study considering the three initialization methods and the two additional constraints introduced (see Table 2). The solution for the MADRP problem without

Algorithm 2 Prioritized Greedy initialization method.

Input: $\mathbb{A} = \{a_1, \dots, a_m\}$, $T = \{t_1, \dots, t_n\}$
Output: $\boldsymbol{P} = \{p_1, \dots, p_m\}$

$T' = \{t'_1, \dots, t'_n\} \leftarrow T = \{t_1, \dots, t_n\}$ ▷ $t'_1 \leftarrow t_1, \dots, t'_n \leftarrow t_n$
$\mathbb{A}' = \{a'_1, \dots, a'_m\} \leftarrow \mathbb{A} = \{a_1, \dots, a_m\}$ ▷ $a'_1 \leftarrow a_1, \dots, a'_m \leftarrow a_m$
$\boldsymbol{P} \leftarrow \{p_1 \leftarrow [a_1], \dots, p_m \leftarrow [a_m]\}$ ▷ $\boldsymbol{P}$ *is a set of* m *arrays*
while T' is not empty **do**
  $\mathcal{I} \leftarrow [1, \dots, m]$ ▷ $\mathcal{I}$ *is the array of* m *agents' indices*
  while $\mathcal{I}$ is not empty **do**
    $i \leftarrow$ find agent $i \in \mathcal{I}$ for which the distance of the nearest unassigned target to reference location a'_i is minimum
    find the nearest target t'_x to reference location a'_i ▷ $t'_x \in T'$, $1 \le x \le n$
    append target t'_x to the end of p_i ▷ *Target assignment,* $p_i \in \boldsymbol{P}$
    $\mathcal{I} \leftarrow \mathcal{I} - i$ ▷ *Updating* $\mathcal{I}$
    $a'_i \leftarrow t'_x$ ▷ *Updating* $a'_i \in \mathbb{A}'$
    $T' \leftarrow T' - t'_x$ ▷ *Updating* T'
    if T' is empty **then**
      return $\boldsymbol{P}$
    end if
  end while
end while

Algorithm 3 Sorted Greedy initialization method.

Input: $\mathbb{A} = \{a_1, \dots, a_m\}$, $T = \{t_1, \dots, t_n\}$
Output: $\boldsymbol{P} = \{p_1, \dots, p_m\}$

$T' = \{t'_1, \dots, t'_n\} \leftarrow T = \{t_1, \dots, t_n\}$ ▷ $t'_1 \leftarrow t_1, \dots, t'_n \leftarrow t_n$
$\mathbb{A}' = \{a'_1, \dots, a'_m\} \leftarrow \mathbb{A} = \{a_1, \dots, a_m\}$ ▷ $a'_1 \leftarrow a_1, \dots, a'_m \leftarrow a_m$
$\boldsymbol{P} \leftarrow \{p_1 \leftarrow [a_1], \dots, p_m \leftarrow [a_m]\}$ ▷ $\boldsymbol{P}$ *is a set of* m *arrays*
assign one target to each agent using Prioritized Greedy method, so one target is appended at the end of each p_i and $\mathbb{A}'$ is also updated ▷ $p_i \in \boldsymbol{P}$
$\mathcal{I} \leftarrow [1, \dots, m]$ ▷ $\mathcal{I}$ *is the array of* m *agents' indices*
while T' is not empty **do**
  $\mathcal{I}_{sorted} \leftarrow$ sort the agents' indices in $\mathcal{I}$ based on the distance of each agent's current path p_i in ascending order, so the first index will be the index of the agent with the smallest distance of its current path p_i and so on in ascending order to sort the rest of agents' indices ▷ $p_i \in \boldsymbol{P}$
  for all $i \in \mathcal{I}_{sorted}$ **do**
    find the nearest target t'_x to reference location a'_i ▷ $t'_x \in T'$, $1 \le x \le n$
    append target t'_x to the end of p_i ▷ *Target assignment,* $p_i \in \boldsymbol{P}$
    $a'_i \leftarrow t'_x$ ▷ *Updating* $a'_i \in \mathbb{A}'$
    $T' \leftarrow T' - t'_x$ ▷ *Updating* T'
    if T' is empty **then**
      return $\boldsymbol{P}$
    end if
  end for
end while

the additional constraints was also considered in order to enable us to evaluate the effect of limiting the service area for each agent by our two additional constraints that were applied independently of each other.

4.2. Evolution phase

After an initial Plan $\boldsymbol{P}$ is generated in the initialization phase, the evolution phase evolves $\boldsymbol{P}$ towards a near-optimal Plan (final Plan) by minimizing the cost $c_m(\boldsymbol{P})$ see Eq. (2). The evolution phase is an iterative process, where each iteration is called a generation step. In every generation step, three genetic-inspired operators along with another

Table 2
Analyzed solution modes.

Solution mode	Initialization method			Additional constraint	
	Greedy	Prioritized Greedy	Sorted Greedy	Max. GeoRange for each agent	Max. spatial range of targets for each agent
1	×				
2	×			×	
3	×				×
4		×			
5		×		×	
6		×			×
7			×		
8			×	×	
9			×		×

heuristic-based operator are applied to the paths $p_i \in \boldsymbol{P}$ with the aim of reducing the cost $c_m(\boldsymbol{P})$. The mechanism of the evolution phase is the same as in a classical GA [45] with one exception: there is only one solution (one Plan $\boldsymbol{P}$) to be evolved (i.e., the population size is 1 since the initialization phase generates only one Plan $\boldsymbol{P}$).

In every generation step, the developed operators are applied sequentially to a Plan $\boldsymbol{P}$ to generate a new Plan $\boldsymbol{P}'$, then:

- If $c_m(\boldsymbol{P}') < c_m(\boldsymbol{P})$, Plan $\boldsymbol{P}'$ is kept for the next generation step and Plan $\boldsymbol{P}$ is discarded.
- If $c_m(\boldsymbol{P}') > c_m(\boldsymbol{P})$, Plan $\boldsymbol{P}$ is kept for the next generation step and Plan $\boldsymbol{P}'$ is discarded.

Note that in every generation step, the output Plan generated by each operator is the input Plan for the next operator (sequential application of the operators), and the cost evaluation is conducted after applying the last operator. Consequently, only one new Plan $\boldsymbol{P}'$ is evaluated in each generation step.

In every generation step, the following evolutionary operators are applied in the same order as they are listed here:

I. *Crossover* operator:
It is stochastically applied with the probability of $\mathcal{P}_{crossover}$. It stochastically employs either the best-worst selection or the random selection to select two paths, then each of them is randomly cleaved into two parts (i.e., cleaving position (index) can be different for each of them) and the parts are swapped. In the case of employing the best-worst selection, the path with the smallest cost in Plan $\boldsymbol{P}$ (the best path) and the path with the largest cost in Plan $\boldsymbol{P}$ (the worst path) are selected for the cleaving and swap process. The probability of employing the best-worst selection is $\mathcal{P}_{best-worst}$, while the random selection is employed with the probability of $1 - \mathcal{P}_{best-worst}$. The schematic diagram of this operator is illustrated in Fig. 1(a) and Fig. 1(b). Note that in these figures, the size of the cell representing each path does not correspond to the length of the path. The focus of these figures is on visualizing the random cleaving position (index) for the selected paths.

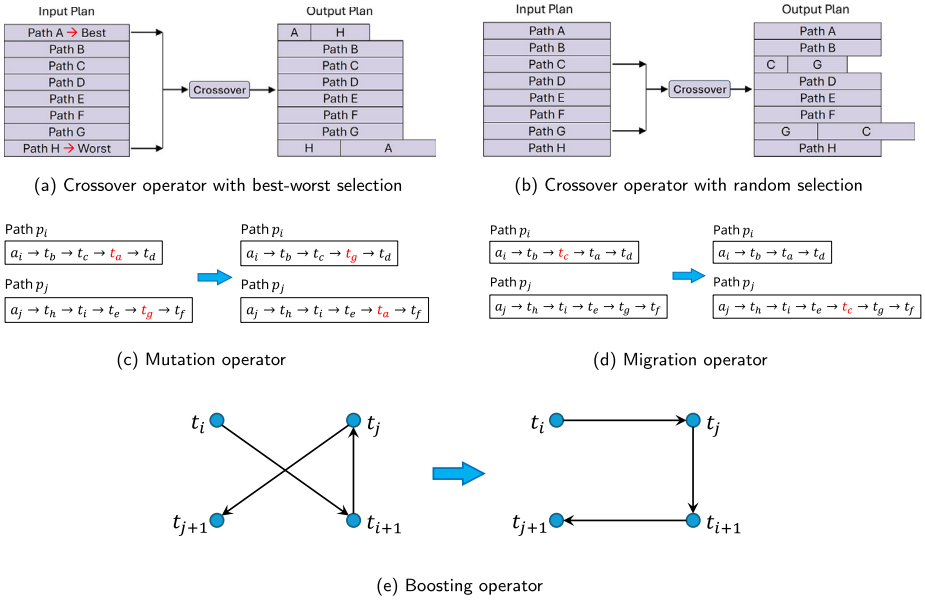
II. *Mutation* operator:
It is stochastically applied with the probability of $\mathcal{P}_{mutation}$. It randomly selects two paths and swaps two randomly selected targets between them. The schematic diagram of this operator is shown in Fig. 1(c).

III. *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 at a random position (index) in p_j. Note that in this procedure, the length of p_i decreases, while the length of p_j increases. The schematic diagram of this operator is shown in Fig. 1(d).

IV. *Boosting* operator:
It is stochastically applied with the probability of $\mathcal{P}_{boost}$. It improves the quality of paths by applying the 2-OPT algorithm [48]. This algorithm examines whether the inequality (3) between each four targets t_i, t_{i+1}, t_j, t_{j+1} of a path is valid or not. If it is valid, arcs (t_i, t_{i+1}) and (t_j, t_{j+1}) are substituted with arcs (t_i, t_j) and (t_{i+1}, t_{j+1}), respectively. This decreases the cost of the path and improves its quality by reducing the occurrence of intersecting arcs in it. Note that for a symmetric problem, the terms $w(t_{i+1}, t_j)$ and $w(t_j, t_{i+1})$ in the inequality (3) cancel each other out because $w(t_{i+1}, t_j) = w(t_j, t_{i+1})$. The schematic diagram of this operator is illustrated in Fig. 1(e).

$$w(t_i, t_{i+1}) + w(t_{i+1}, t_j) + w(t_j, t_{j+1}) > w(t_i, t_j) + w(t_j, t_{i+1}) + w(t_{i+1}, t_{j+1}) \tag{3}$$

Here, for simplicity, the algorithms of both the initialization phase and the evolution phase are described without applying the additional



(a) Crossover operator with best-worst selection

(b) Crossover operator with random selection

(c) Mutation operator

(d) Migration operator

(e) Boosting operator

Fig. 1. Schematic diagrams of the evolution operators. ($a_i \in \mathbb{A} = \{a_1, \ldots, a_m\}$ and $t_i \in T = \{t_1, \ldots, t_n\}$, where $\mathbb{A}$ is the set of m starting locations of the m agents and T is the set of n target locations. Thus, the alphabetical indices of a and t belong to the index sets $\{1, \ldots, m\}$ and $\{1, \ldots, n\}$, respectively.).

Table 3
Time complexity of the components of the proposed GA.

	Initialization Phase			Evolution Phase				
	Greedy	Prioritized Greedy	Sorted Greedy	Crossover	Mutation	Migration	Boosting	Cost Evaluation
Time Complexity	$O(n)$	$O(m \times n)$	$O(m^2 + n^2/m)$	$O(n)$	$O(1)$	$O(1)$	$O(n^2/m)$	$O(n)$

m and n denote the number of agents and targets, respectively.

constraints (solution modes 1, 4 and 7). Applying these constraints only makes one difference in the algorithms: any target assignment (including path modifications in the evolution phase) can be done only if the involved targets satisfy the applied constraint with respect to the corresponding agent. Note that in solution modes 2, 5 and 8, geodetic distance (d_g) is only used for checking whether the first additional constraint is satisfied or not, and for all other distance calculations, the distance metric is the shortest driving distance ($\boldsymbol{d}_{sd}$) calculated using the OpenRouteService API.

After describing the algorithms for the components of the proposed GA, the time complexity of each component is presented in Table 3. The time complexity of the Evolution Phase is $O\left(g \times \left(n + 1 + 1 + \frac{n^2}{m} + n\right)\right) \approx O\left(g \times \frac{n^2}{m}\right)$, with $n \geq m > 1$, and m, n, and g denoting the number of agents, targets, and generation steps, respectively. Accordingly, considering $g \gg m > 1$ and $n \geq m$, the estimated upper bound for the time complexity of the entire proposed GA is approximately $O\left(g \times \frac{n^2}{m}\right)$, regardless of the initialization method applied. Thus, the proposed GA runs in approximately polynomial time.

5. Results and discussion

5.1. Benchmark results

We benchmarked the performance of our route planning method using the most recent and relevant results from Ref. [39], where Ergüven and Polat introduced the so-called Relative Distances Approach (RDA) to solve the Multi-Depot Open-Path Multiple Traveling Salesman Problem (MDOP-MTSP), the generalization of the MAPP problem. Ergüven and Polat [39] achieved the Best-Known Solutions (BKS) for a wide variety of MDOP-MTSPs based on the TSPLIB[1] instances and benchmarked their results against the results of three previous studies, [14,26,40]. In general, depots in their traditional meaning do not exist in the MAPP problems, but the starting location of each agent can be translated as a single-agent depot in the broader context of the MDOP-MTSP. Hence, we specifically benchmarked the performance of our solution method using the results obtained for the MDOP-MTSPs with single-agent depots in [39]. Note that according to the best of our knowledge, there is no other study that solves the exact same practical problem as the MADRP problem.

Furthermore, in all of these studies [14,26,39,40], similarly to the majority of MTSP benchmark references, the only optimization objective considered for most of the benchmark problems was minimizing the total traveling distance of all paths (Min-Sum). Thus, for such benchmark problems, we minimized the cost $c_{tot}(\boldsymbol{P})$:

$$c_{tot}(\boldsymbol{P}) = \sum_{i=1}^{m} c(p_i) \tag{4}$$

Additionally, we did not apply any of our limited service area constraints to the benchmark problems because: (1) these constraints have been designed to improve the solutions of the route planning problems with the Min-Max optimization objective, and therefore their impact on the solutions of the problems with the Min-Sum objective is not positive; (2) they can only be applied to real road network problems (see Section 3), while the TSPLIB instances used in [14,26,39,40] do not contain the necessary data for applying these constraints; and (3) in these studies, limited service area constraints were not applied to the problems. Thus, to make a fair comparison, we solved the benchmark problems only using solution modes 1, 4, and 7 (see Table 2).

Moreover, none of these studies provided the starting locations of the agents. The only relevant information was mentioned in [39], where the authors stated that they chose the starting locations that allowed their method to obtain better results than those in [14,26,40] for the same benchmark problems. Hence, it can be deduced that all of these studies selected the starting locations that enabled their own methods to achieve competitive results against their benchmark references. Accordingly, we also adopted the same approach.

The evolution operators 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 $\mathcal{P}_{boost} = 0.3$. Since these stochastic probabilities are involved in the evolution phase, for each of the solution modes applied in this section (solution modes 1, 4, and 7), the simulations were run 20 times using MATLAB R2022b and the result (final cost) of the simulation run that obtained the minimum final cost is reported. Here, the final cost refers to $c_{tot}(\boldsymbol{P}_{final})$ or $c_m(\boldsymbol{P}_{final})$ depending on the optimization objective considered for each benchmark problem see Eq. (4) and (2) for $c_{tot}(\boldsymbol{P}_{final})$ and $c_m(\boldsymbol{P}_{final})$, respectively). For each simulation run, the number of generation steps for the evolution phase was set to 150,000. Here, a simulation run refers to one-time execution of the proposed solver by applying one of the solution modes 1, 4 or 7. The configuration of the PC used was as follows: CPU: AMD Ryzen™ 5-5500U, RAM: 8 GB, OS: Windows 11 Pro. The average runtime $\overline{\mathcal{T}}$ of one simulation run is also reported but only for indicative purposes since the PC configuration and the programming language used for the simulations were not reported in [39]. Additionally, the gap G_{bks} is also reported which shows the gap between the Best Known Solution (BKS) value and the best solution value we obtained. G_{bks} is calculated as $G_{bks} = 100 \times (f_{best} - f_{bks})/f_{bks}$, where f_{bks} denotes the Best Known Solution (BKS) value and f_{best} represents the best result among the results achieved by our method using solution modes 1, 4, and 7. A negative G_{bks} value indicates an improvement over the BKS.

For the MDOP-MTSP with single-agent depots, three sets of benchmark problems based on TSPLIB instances were solved:

I. Problem set A that includes 14 symmetric problems based on six different TSPLIB instances, with sizes ranging from 51 to 439 nodes. For these problems, we compared the results of our method with the results of RDA from Ref. [39] and SModel (a graph simplification model) from Ref. [40]. All of these problems were solved by considering Min-Sum optimization objective, the same as in [39,40].
II. Problem set B consisting of six asymmetric problems based on two different TSPLIB instances, which have 45 and 48 nodes, respectively. For these problems, we compared the results of our method with the results of RDA from Ref. [39] and TR (an MTSP to TSP transformation method with IP) from Ref. [14]. All of these problems were solved by considering Min-Sum optimization objective, the same as in [14,39].
III. Problem set C that contains three symmetric problems based on three different TSPLIB instances, with sizes ranging from 51 to 101 nodes. For these problems, we compared the results of our method

[1] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

Table 4
Benchmark results of Problem set A: the proposed GA vs. RDA vs. SModel. Optimization objective: Min-Sum.

TSPLIB instance	No. of agents	SModel	RDA	Proposed GA solution mode 1	Proposed GA solution mode 4	Proposed GA solution mode 7	G_{bks} [%]	$\overline{\mathcal{T}}$ [s] for the proposed GA
eil51	2	403	**400**	**389**	**389**	392	**−2.75**	23.367
eil51	4	365	**363**	**363**	371	**363**	0	14.254
eil51	5	356	**352**	**352**	**352**	355	0	13.135
berlin52	2	7409	**6705**	**6649**	6654	**6649**	**−0.84**	24.156
berlin52	3	7019	**6764**	6411	6520	**6378**	**−5.71**	20.167
berlin52	7	5818	**5607**	**5140**	5342	5255	**−8.33**	12.578
st70	3	613	**600**	**589**	595	**589**	**−1.83**	29.044
st70	4	579	**571**	577	576	**569**	**−0.35**	23.860
pr136	2	101,736	**92,316**	**95,384**	96,820	96,056	+3.32	147.989
pr136	3	99,358	**90,681**	**92,308**	94,965	93,822	+1.79	107.561
pr136	8	88,254	**83,745**	**81,960**	**81,960**	**81,960**	**−2.13**	34.988
lin318	2	47,065	**44,096**	**42,764**	43,385	43,508	**−3.02**	808.110
lin318	10	40,986	**39,539**	**40,570**	41,678	41,447	+2.61	167.335
pr439	5	105,894	**102,557**	**104,990**	111,782	111,449	+2.37	691.801

Table 5
Benchmark results of Problem set B: the proposed GA vs RDA vs TR. Optimization objective: Min-Sum.

TSPLIB instance	No. of agents	TR	RDA	Proposed GA solution mode 1	Proposed GA solution mode 4	Proposed GA solution mode 7	G_{bks} [%]	$\overline{\mathcal{T}}$ [s] for the proposed GA
ftv44	2	1460	**1450**	1443	1439	**1433**	**−1.17**	19.034
ftv44	3	1407	**1355**	**1342**	**1342**	**1342**	**−0.96**	14.491
ftv44	5	1317	**1224**	1225	1231	**1217**	**−0.57**	10.971
ftv47	2	1623	**1567**	1571	1569	**1561**	**−0.38**	22.655
ftv47	3	1540	**1483**	1469	1469	**1462**	**−1.42**	16.663
ftv47	5	1471	**1330**	1379	1386	**1342**	+0.90	11.894

Table 6
Benchmark results of Problem set C: the proposed GA vs RDA vs IPGA. Optimization objective: Min-Sum.

TSPLIB instance	No. of agents	IPGA	RDA	Proposed GA solution mode 1	Proposed GA solution mode 4	Proposed GA solution mode 7	G_{bks} [%]	$\overline{\mathcal{T}}$ [s] for the proposed GA
eil51	3	409	**378**	**377**	380	**377**	**−0.26**	17.776
eil76	3	526	**504**	494	494	**489**	**−2.98**	35.756
eil101	3	597		**593**	598	596	**−0.67**	56.981

Table 7
Benchmark results of Problem set C: the proposed GA vs. IPGA. Optimization objective: Min-Max.

TSPLIB instance	No. of agents	IPGA	Proposed GA solution mode 1	Proposed GA solution mode 4	Proposed GA solution mode 7	G_{bks} [%]	$\overline{\mathcal{T}}$ [s] for the proposed GA
eil51	3	159	135	**133**	**133**	**−16.35**	15.972
eil76	3	201	177	**173**	175	**−13.93**	29.698
eil101	3	231	215	**214**	**214**	**−7.36**	50.556

with the results of RDA from Ref. [39] and IPGA (an improved partheno-genetic algorithm) from Ref. [26]. For the problems of this set, apart from benchmarking the results of our method by considering the Min-Sum optimization objective, we also compared our results for the Min-Max optimization objective with those of IPGA reported in [26]. Note that in [39], the Min-Max optimization objective was not considered for these problems.

Regarding the MDOP-MTSP with single-agent depots and Min-Max optimization objective, in [39], the best solutions were only reported for four problems based on "berlin52" instance by applying task orders, which means that, for instance, for two targets t_1 and t_2, the solver must first assign target t_1 (the target with the smaller index indicating its task order precedence) to an agent before assigning target t_2. Hence, any other order of task assignment is excluded from the solution space. Therefore, it is not fair to compare our results for the Min-Max optimization objective with those in [39]. Obviously, the results of our method for the Min-Max optimization objective outperform those due to considering the whole solution space.

Tables 4 and 5 show the benchmark results for the problems of sets A and B, respectively. Tables 6 and 7 present the benchmark results for the problems of set C. Prior to this study, RDA provided the Best-Known Solution (BKS) values for all benchmark problems investigated here, except for the problems in Table 7 and one problem in Table 6 where IPGA provided the BKS values. For each problem, if there is more than one solution method from Ref. [14,26,39,40] providing the results, the BKS value is shown with bold numbers. Additionally, the best result achieved by the proposed GA for each problem and the negative G_{bks} values are also shown with bold numbers (a negative G_{bks} value indicates an improvement over the corresponding BKS value).

Among the 14 problems of set A, the proposed GA achieved a result better than or equal to the BKS value for 10 problems (see Table 4). For eight of these problems, the proposed GA obtained new BKS values, where the maximum improvement over the previous BKS values is 8.33 % (G_{bks} = −8.33 %). Most of these new BKS values were achieved by employing either the Greedy initialization (solution mode 1) or the Sorted Greedy initialization (solution mode 7). For the four problems

Table 8
Theoretical vs data-driven time complexity evaluation.

Problem 1	Problem 2	m_1 No. of agents for Problem 1	m_2 No. of agents for Problem 2	n_1 No. of targets in Problem 1	n_2 No. of targets in Problem 2	$\frac{(n_2/n_1)^2}{m_2/m_1}$	$\frac{\overline{\mathcal{T}}_2}{\overline{\mathcal{T}}_1}$
lin318 with 10 agents	lin318 with 2 agents	10	2	308	316	5.263	4.829
pr136 with 8 agents	pr136 with 2 agents	8	2	128	134	4.384	4.230
eil51 with 5 agents	pr439 with 5 agents	5	5	46	434	89.015	52.668
berlin52 with 2 agents	lin318 with 2 agents	2	2	50	316	39.942	33.454
st70 with 4 agents	pr136 with 3 agents	4	3	66	133	5.414	4.508
berlin52 with 3 agents	eil51 with 2 agents	3	2	49	49	1.500	1.159
ftv47 with 5 agents	ftv44 with 2 agents	5	2	42	42	2.500	1.600
ftv44 with 5 agents	ftv47 with 3 agents	5	3	39	44	2.121	1.519

$\overline{\mathcal{T}}_1$ and $\overline{\mathcal{T}}_2$ denote the average runtime of Problem 1 and Problem 2, respectively.

where the proposed GA achieved a result worse than the BKS value, the maximum difference with the BKS values is only 3.32 % ($G_{bks} = +3.32$ %).

Out of the six problems of set B, which includes asymmetric problems, the proposed GA applying the Sorted Greedy initialization (solution mode 7) obtained new BKS values for five problems (see Table 5), where the maximum improvement over the previous BKS values is 1.42 % ($G_{bks} = -1.42$ %). For the only problem where the proposed GA achieved a result worse than the BKS value, the maximum difference with the BKS value is only 0.9 % ($G_{bks} = +0.9$ %).

Regarding the problems of set C, in the case of considering the Min-Sum optimization objective, the proposed GA obtained new BKS values for all the problems (see Table 6), where the maximum improvement over the previous BKS values is 2.98 % ($G_{bks} = -2.98$ %). These new BKS values were achieved by employing either the Greedy initialization (solution mode 1) or the Sorted Greedy initialization (solution mode 7). Furthermore, the proposed GA achieved even better results in the case of considering the Min-Max optimization objective, where it obtained new BKS values for all the problems (see Table 7), with the maximum improvement of 16.35 % ($G_{bks} = -16.35$ %) compared to the previous BKS values. These new BKS values were achieved by employing either the Prioritized Greedy initialization (solution mode 4) or the Sorted Greedy initialization (solution mode 7).

Analysis of the variation in the average runtime $\overline{\mathcal{T}}$ with respect to the number of agents (m) and targets (n) indicates that the proposed GA runs in approximately polynomial time, with $O\left(g \times \frac{n^2}{m}\right)$ as the estimated upper bound, and without bottlenecks in practice. This is consistent with the theoretical analysis of time complexity presented in Section 4.2. Considering $O\left(g \times \frac{n^2}{m}\right)$, when the number of generation steps (g) is constant (it was the same for all simulation runs), the estimated upper bound for the change in the runtime is equal to the change in $\frac{n^2}{m}$. This relation was investigated using the $\overline{\mathcal{T}}$ achieved for various benchmark problems (see Table 8). The results presented in Table 8 indicate that $O\left(g \times \frac{n^2}{m}\right)$ is a good estimate of the upper bound for the time complexity of the proposed GA.

For a method that involves stochastic elements, it is standard practice to execute the method multiple times and compare the best value, mean, and standard deviation of the results against the same type of benchmark results. However, except for the average results reported for three problems in [39], only the best results were reported in all benchmark references [14,26,39,40]. Accordingly, we adopted the same approach and reported our best results in order to ensure the comparability of the results. Moreover, except for [26], all other benchmark references [14,39,40] did not mention the number of observations, i.e., the number of times their method was executed to obtain results and report the best value among them. In [26], the number of observations considered was 10, which means that the best result obtained from 10 executions of IPGA for each benchmark problem was reported. Hence, IPGA was executed with 10 different random seeds. In this study, 20 observations were considered to report the best result for each benchmark problem solved by each solution mode (20 simulation runs using each solution mode). Thus, 20 random seeds were tested for each solution mode (the same 20 initial random seeds applied to each solution mode). For the sake of maximum possible fairness in comparison with the benchmark results of IPGA reported in [26], we note that even by considering only the first 10 observations, all three solution modes 1, 4, and 7 achieved the same best results that are compared with the IPGA's best results in Tables 6 and 7.

Furthermore, to benchmark the average performance of the proposed GA, Table 9 compares the average results of the proposed GA with those of RDA for the three problems for which only the average results of RDA were reported in [39]. These problems are based on ulysses16, a Geo TSPLIB instance. For Geo instances, TSPLIB provides the geographical coordinates of the nodes, and the standard practice is to calculate the distance matrix using the geodetic distance between the nodes. However, in [39], ulysses16 was treated as a Euclidean instance, i.e., the geographical coordinates were considered as Euclidean coordinates and the distance matrix was calculated using Euclidean distance. Accordingly, we adopted the same approach to enable benchmarking our average results against those of RDA. For each solution mode of the proposed GA, the final costs of 20 observations (20 simulation runs) were averaged to obtain the average result, while the number of observations considered to calculate the average results of RDA was not mentioned in [39]. In Table 9, G represents the gap between the RDA's average result and the best average result we obtained. G is calculated as $G = 100 \times (\overline{f}_{best} - \overline{f}_{RDA})/\overline{f}_{RDA}$, where $\overline{f}_{RDA}$ denotes the RDA's average result and $\overline{f}_{best}$ represents the best average result among the average results achieved by our method using solution modes 1, 4, and 7. A negative G value indicates an improvement over the RDA's average result. As shown in Table 9, for all problems, the proposed GA achieved a smaller average result compared to the average result of RDA. The improvement over the average results of RDA is even more significant, with a minimum of 23.66 % ($G = -23.66$ %), in the case of considering the Min-Max optimization objective, the main objective for which our method was developed.

The benchmark results demonstrate the superiority and robustness of the proposed GA even without applying our limited service area constraints, as it achieved new Best Known Solution (BKS) values for 19 benchmark problems and a result equal to the BKS value for another two problems out of the 26 benchmark problems investigated in total.

5.2. Real-world test case

A real-world test case was designed to validate the developed route planning method and demonstrate its applicability. This test case consists of 5 agents and 61 targets spread out within a rectangular geographic area with the size of around $12.24 \times 14.48 = 177.24$ km^2 located in Budapest urban area (capital of Hungary). Google Earth Pro was used for generating the test case and retrieving the geographical coordinates of the locations (latitude-longitude). The test case is shown in Fig. 2. In this figure, the red pins depict the starting locations of the agents,

Table 9
Average results for three problems based on ulysses16 instance: the proposed GA vs. RDA.

TSPLIB instance: ulysses16. Optimization objective: Min-Sum						
No. of agents	RDA	Proposed GA solution mode 1	Proposed GA solution mode 4	Proposed GA solution mode 7	G [%]	$\overline{\mathcal{T}}$ [s] for the proposed GA
2	47.962	**46.226**	46.483	46.612	**−3.62**	5.621
3	42.583	**41.784**	42.140	41.986	**−1.88**	4.106
4	39.330	**38.047**	39.169	38.721	**−3.26**	2.613
TSPLIB instance: ulysses16. Optimization objective: Min-Max						
No. of agents	RDA	Proposed GA solution mode 1	Proposed GA solution mode 4	Proposed GA solution mode 7	G [%]	$\overline{\mathcal{T}}$ [s] for the proposed GA
2	33.010	**25.200**	**25.200**	**25.200**	**−23.66**	5.609
3	25.871	**17.814**	**17.814**	17.949	**−31.14**	4.041
4	21.300	**15.963**	**15.963**	**15.963**	**−25.06**	2.556

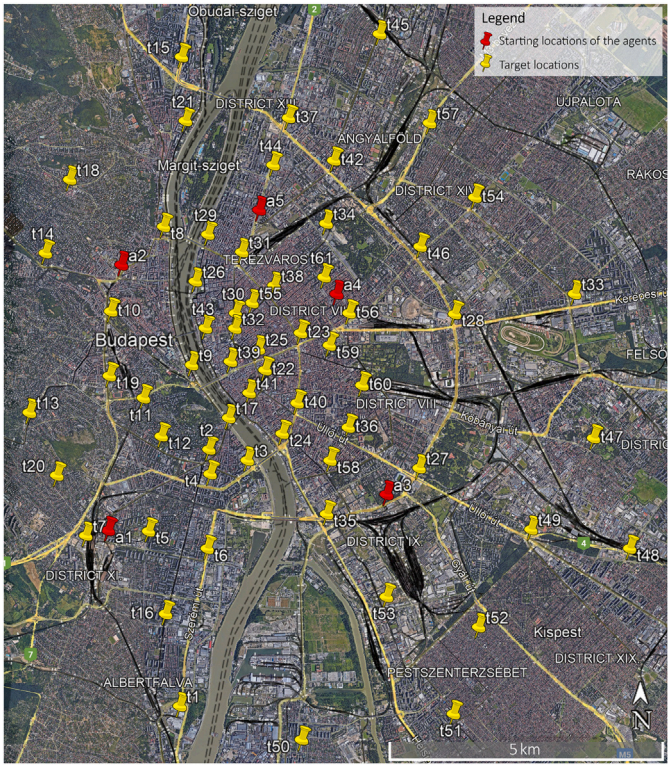


Fig. 2. The test case consisting of 5 agents (red pins) and 61 targets (yellow pins) spread out in Budapest urban area (capital of Hungary).

and the yellow pins show the target locations. The target locations were selected with the focus on the inner city and the downtown area. The starting locations of the agents were selected on the eastern and western edges of the inner city. The Danube River flows from the north to the south of the city of Budapest. The quantity of the selected locations is larger in the area located on the east bank of the Danube River, reflecting the higher population density.

The evolution operators 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 $\mathcal{P}_{boost} = 0.3$. Since these stochastic probabilities are involved in the evolution phase, for each solution mode (see Table 2), the simulations were run 100 times using MATLAB R2022b and the average and minimum results for the cost of final Plans are reported. For each simulation run, the number of generation steps for the evolution phase was set to 150,000. Here, a simulation run refers to one-time execution of the proposed solver by applying one of the solution modes (see Table 2). The configuration of the PC used was as follows: CPU: AMD Ryzen™ 5-5500U, RAM: 8 GB, OS: Windows 11 Pro.

Here, $\boldsymbol{P}_{initial}$ and $\boldsymbol{P}_{final}$ denote the initial Plan and the final Plan, respectively. Additionally, the cost of initial Plan $c_m(\boldsymbol{P}_{initial})$ is called the initial cost and the cost of final Plan $c_m(\boldsymbol{P}_{final})$ is called the final cost see Eq. (2) for the cost $c_m(\boldsymbol{P})$). In every simulation run for the same solution mode, the same initial Plan is generated, and therefore the initial cost $c_m(\boldsymbol{P}_{initial})$ is the same. However, in each simulation run for the same solution mode, the final cost $c_m(\boldsymbol{P}_{final})$ can vary because the obtained final Plan can be different due to the application of several stochastic probabilities in the evolution phase. Accordingly, in order to evaluate the simulation results in one simulation run, the following parameters were considered: the initial cost $c_m(\boldsymbol{P}_{initial})$; the average final cost $\overline{c_m}(\boldsymbol{P}_{final})$ calculated by averaging the 100 final costs achieved in 100 simulation runs; $c_m(\boldsymbol{P}_{final})_{min}$ representing the minimum final cost obtained in 100 simulation runs; r denoting the maximum geodetic range for each agent; s denoting the maximum spatial range of targets for each agent; and the average runtime $\overline{\mathcal{T}}$ calculated by dividing the total runtime of 100 simulation runs by 100.

For the presented test case, the developed solution numerically calculates the minimum value for r and s such that the out-of-GeoRange or the out-of-SpatialRange issue would not occur in the solution modes where the additional constraint imposes either r or s. Accordingly, r and s are case-sensitive variables in this study. r is imposed in solution modes 2, 5 and 8 where the maximum geodetic range for each agent is limited, while s is imposed in solution modes 3, 6 and 9 where the maximum spatial range of targets for each agent is restricted.

The computational results are shown in Table 10. In brief, the developed solution achieved:

- The smallest average final cost (26,866.94 m) and the smallest minimum final cost (24,582.80 m) in the case of employing the Prioritized Greedy initialization along with limiting the maximum geodetic range for each agent (solution mode 5).
- The shortest average runtime (13.117 s) in the case of employing the Prioritized Greedy initialization along with restricting the maximum spatial range of targets for each agent (solution mode 6).
- The smallest initial cost (34,295.26 m) in the case of employing the Sorted Greedy initialization without any additional constraints (solution mode 7).

Furthermore, it can be observed that:

- In the absence of any additional constraints, employing the Sorted Greedy initialization led to the smallest initial cost, the smallest average final cost, and the shortest average runtime compared to the results achieved in the case of employing either of the other initialization methods (solution mode 7 compared to solution modes 1 and 4). However, the smallest minimum final cost in the absence of the additional constraints was obtained using the Prioritized Greedy initialization (solution mode 4 compared to solution modes 1 and 7).
- If the service area for each agent was limited by applying either of the additional constraints, employing the Prioritized Greedy initialization resulted in the smallest initial cost, the smallest average

Table 10
Computational results for the real-world test case.

Solution mode	$c_m(\boldsymbol{P}_{initial})$ [m]	$\overline{c_m}(\boldsymbol{P}_{final})$ [m]	$c_m(\boldsymbol{P}_{final})_{min}$ [m]	r [m]	s [m]	$\overline{\mathcal{T}}$ [s]
1	42,964.02	28,147.42	25,906.29			13.716
2	37,977.50	26,937.21	24,815.05	4750		13.384
3	36,904.46	27,608.56	25,124.57		6420	13.182
4	46,307.70	28,156.05	24,994.12			13.767
5	37,818.20	26,866.94	24,582.80	4750		13.358
6	36,700.05	27,406.51	25,124.57		6420	13.117
7	34,295.26	28,056.85	25,258.41			13.562
8	38,429.83	27,319.42	25,124.57	4750		13.428
9	37,311.68	27,520.66	25,124.57		6420	13.213

final cost and the shortest average runtime compared to the results achieved in the case of employing either of the other initialization methods along with applying the same additional constraint (solution mode 5 compared to solution modes 2 and 8; solution mode 6 compared to solution modes 3 and 9).

- Regardless of which initialization method was used, limiting the service area for each agent by either of the additional constraints always led to a smaller average final cost compared to the average final cost achieved by employing the same initialization method without applying any additional constraints (solution modes 2 and 3 compared to solution mode 1; solution modes 5 and 6 compared to solution mode 4; solution modes 8 and 9 compared to solution mode 7). This reduction in the average final cost was greater when the maximum geodetic range for each agent was limited and its maximum value was around 4.6 % ($\overline{c_m}(\boldsymbol{P}_{final})$ in solution mode 5 compared to $\overline{c_m}(\boldsymbol{P}_{final})$ in solution mode 4). A similar trend was observed regarding the average runtime with the difference that the reduction in the average runtime was larger when the maximum spatial range of targets for each agent was restricted. The maximum value of this reduction in the average runtime was around 4.7 % ($\overline{\mathcal{T}}$ in solution mode 6 compared to $\overline{\mathcal{T}}$ in solution mode 4). These reductions in the average final cost and the average runtime demonstrate that limiting the service area for each agent by either of the additional constraints improved the performance of the developed solution. Additionally, it can be deduced that limiting the maximum geodetic range for each agent was more effective for decreasing the average final cost, while restricting the maximum spatial range of targets for each agent was more effective for reducing the average runtime.
- Except for the case of employing the Prioritized Greedy initialization along with restricting the maximum spatial range of targets for each agent, limiting the service area for each agent by either of the additional constraints always led to a smaller minimum final cost compared to the minimum final cost achieved by employing the same initialization method without applying any additional constraints (solution modes 2 and 3 compared to solution mode 1; solution mode 5 compared to solution mode 4; solution modes 8 and 9 compared to solution mode 7). This reduction in the minimum final cost further indicates the positive impact of the additional constraints on the overall performance of the developed solution.
- In the case of employing either the Greedy initialization or the Prioritized Greedy initialization, limiting the service area for each agent by either of the additional constraints led to a smaller initial cost compared to the initial cost achieved by employing the same initialization method without applying any additional constraints (solution modes 2 and 3 compared to solution mode 1; solution modes 5 and 6 compared to solution mode 4). This decrease in the initial cost was more significant when the maximum spatial range of targets for each agent was restricted (solution mode 3 compared to solution modes 1 and 2; solution mode 6 compared to solution modes 4 and 5).

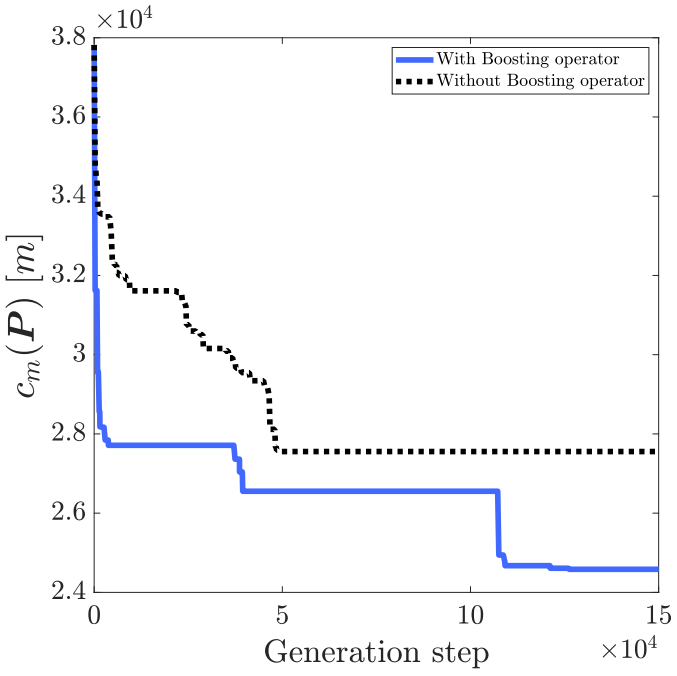
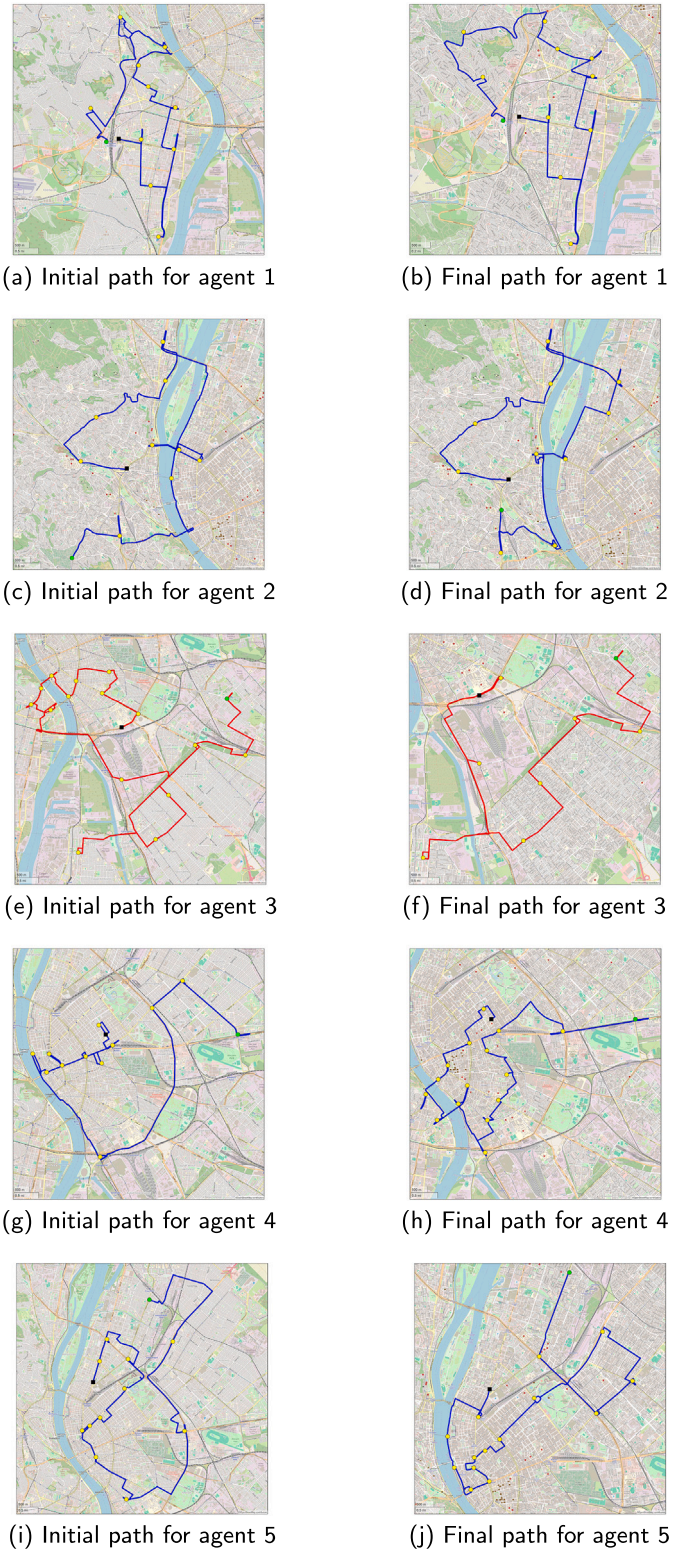


Fig. 3. Evolution of the cost $c_m(\boldsymbol{P})$ during the evolution phase of the two simulation runs where solution mode 5 achieved the minimum final cost $c_m(\boldsymbol{P}_{final})_{min}$ among all simulation runs for solution mode 5 with and without applying the Boosting operator, respectively.

- In the case of employing the Sorted Greedy initialization, limiting the service area for each agent by either of the additional constraints resulted in a larger initial cost, a smaller average final cost, a smaller minimum final cost and a shorter average runtime compared to the results achieved by employing the same initialization method without applying any additional constraints (solution modes 8 and 9 compared to solution mode 7). Thus, even though in this case the additional constraints increased the initial cost, the impact of them on the overall performance of the developed solution was still positive.
- The calculated values for r and s were approximately equal to 25 % and 34 % of the diameter of the geographic area of the test case, respectively (this diameter was around 18.96 km).

It is worth mentioning that although applying the Boosting operator (the 2-OPT algorithm) is quite effective in further reducing the final cost, it can cause the solution to get stuck in a local optimum for some consecutive generation steps. Fig. 3 shows the evolution of the cost $c_m(\boldsymbol{P})$ during the evolution phase of the two simulation runs where solution mode 5 achieved the minimum final cost $c_m(\boldsymbol{P}_{final})_{min}$ among



(a) Initial path for agent 1 (b) Final path for agent 1

(c) Initial path for agent 2 (d) Final path for agent 2

(e) Initial path for agent 3 (f) Final path for agent 3

(g) Initial path for agent 4 (h) Final path for agent 4

(i) Initial path for agent 5 (j) Final path for agent 5

Fig. 4. The paths in the initial Plan (left) vs the paths in the final Plan (right) generated in the simulation run where solution mode 5 achieved the minimum final cost $c_m(\boldsymbol{P}_{final})_{min}$ among all 100 simulation runs for solution mode 5. (: the path with the longest distance in each Plan, : the other paths in each Plan; In each path: : the starting location of the agent, : the last target the agent visits, : the other targets visited by the agent.).

all simulation runs for solution mode 5 with and without applying the Boosting operator, respectively. It can be observed that without applying the Boosting operator, the proposed GA managed to reduce the cost $c_m(\boldsymbol{P})$ more smoothly, but roughly only during the first 50,000 generation steps and after that no further reduction was achieved. However, by applying the Boosting operator, even though the proposed method ultimately achieved a smaller final cost, it sometimes needed a large number of generation steps to escape from a local optimum.

In order to visualize a path on the map, it is required to have the geographic trajectory of the path, which is the continuous series of geographical coordinates (latitude-longitude) representing the path. However, each path $p_i \in \boldsymbol{P}_{final}$ obtained by the developed solution is a discrete series of geographical coordinates. The geographic trajectory of a path $p_i \in \boldsymbol{P}_{final}$ can be generated by putting together the geographic trajectory of the shortest driving route between each two consecutive geographical coordinates in p_i. This was achieved using the directions module of the OpenRouteService API (the transportation mode profile in the API module must be set to driving-car). This API module takes the paths $p_i \in \boldsymbol{P}_{final}$ as the input and generates the geographic trajectory of each path.

Fig. 4 illustrates the initial Plan and the final Plan generated in one of the simulation runs for solution mode 5 which achieved the smallest average final cost among all solution modes. In this figure, the path with the longest distance in each Plan is shown in red and the other paths are depicted in blue; and in each path, the black square is the starting location of the agent, the green circle is the last target the agent visits and the other targets visited by the agent are depicted by the yellow circles. Note that in the calculation of the shortest driving route between each pair of locations, the road driving rules are applied. Because of the turnaround rules, it may happen that an agent has to drive on the same road section in both directions. The initial Plan and the final Plan illustrated in Fig. 4 specifically belong to the simulation run where solution mode 5 achieved the minimum final cost $c_m(\boldsymbol{P}_{final})_{min}$ among all 100 simulation runs for solution mode 5. In this simulation run, the cost $c_m(\boldsymbol{P})$ was 37,818.20 metres in the initial Plan (the distance of the path shown in Fig. 4(e)) and it reduced to 24,582.80 metres in the final Plan (the distance of the path depicted in Fig. 4(f)). As shown in Fig. 4(e) and Fig. 4(f), agent 3 had the path with the longest distance in both the initial Plan and the final Plan. The evolution of the cost $c_m(\boldsymbol{P})$ during the evolution phase of this simulation run is shown with the blue line in Fig. 3.

Comparing our solution method with previously proposed solution methods using a similar GA to solve an MAPP problem [10,12], the solution method presented here is novel because it employs two newly developed initialization methods, the Prioritized Greedy and the Sorted Greedy. Furthermore, our limited service area constraints were not applied in [10,12] (the limited service area constraint applied in [12] is only applicable to the routing problems in a Euclidean space).

6. Conclusions

This paper presents a cooperative driving route planning method based on an Asymmetric Multi-Agent Path Planning (AMAPP) problem with two additional constraints limiting the service area for each agent: (1) The maximum geodetic range for each agent, and (2) The maximum spatial range of targets for each agent. A solution method employing a Genetic Algorithm (GA) is presented to solve the route planning problem. This GA minimizes the driving distance of the longest route in the route plan provided for a fleet of agents (vehicles) so that the fleet can visit a given number of target locations in the least amount of time. The presented GA benefits from two novel initialization methods, the Prioritized Greedy and the Sorted Greedy, which enhance its performance.

The results obtained in a real-world test case show that limiting the service area for each agent by either of the additional constraints improves the performance of the developed solution. More specifically, limiting the maximum geodetic range for each agent is more effective for decreasing the average driving distance of the longest route in the route plan, while restricting the maximum spatial range of targets for each agent is more effective for reducing the average runtime of the developed solution. Additionally, the results demonstrate that using the Prioritized Greedy initialization causes the employed GA to perform better when the limited service area constraints are applied, while using the Sorted Greedy initialization helps the employed GA to fare better in the absence of any additional constraints. Furthermore, out of the 26 benchmark problems investigated without applying the additional constraints, the proposed GA achieved new Best Known Solution (BKS) values for 19 problems and a result equal to the BKS value for another two problems, demonstrating its superiority and robustness even without applying our limited service area constraints.

These outcomes show that the developed route planning method enhances the efficiency of route planning for a fleet of vehicles when the aim is to optimize open paths in a cooperative manner, and the service area for each agent is limited. This situation can be observed in different mobility and logistics services such as crowd-sourced delivery (e.g., TOURMIX), group ride-sharing, general home delivery, and airport shuttles. The service area for each agent can be limited due to various factors such as planning strategy, pricing strategy, or the preference of each agent itself. This route planning method can be extended by considering other relevant aspects such as dynamic target locations, traffic conditions, time windows or multi-modal itinerary planning. Our future work will focus on improving the computational complexity of the proposed method and incorporating the capability of processing dynamic target locations, ultimately enabling the development of an industry-level multi-agent route planning method based on this framework.

CRediT authorship contribution statement

Ali Maktabifard: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Dávid Földes:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

Dávid Földes would like to express his gratitude to the Hungarian Academy of Sciences for awarding him the Bolyai János Research Scholarship (BO/00393/22). This scholarship provided essential financial support that enabled the completion of this research.

Data availability

Data will be made available on request.

References

[1] D.E. Gomes, M.I.D. Iglésias, A.P. Proença, T.M. Lima, P.D. Gaspar, Applying a genetic algorithm to a m-tsp: case study of a decision support system for optimizing a beverage logistics vehicles routing problem, Electronics 10 (18) (2021) 2298, https://doi.org/10.3390/electronics10182298

[2] A. Király, J. Abonyi, Redesign of the supply of mobile mechanics based on a novel genetic optimization algorithm using Google maps API, Eng. Appl. Artif. Intell. 38 (2015) 122–130, https://doi.org/10.1016/j.engappai.2014.10.015

[3] M. Zhu, X.-Y. Liu, X. Wang, An online ride-sharing path-planning strategy for public vehicle systems, IEEE Trans. Intell. Transp. Syst. 20 (2) (2018) 616–627, https://doi.org/10.1109/TITS.2018.2821003

[4] W. Herbawi, M. Weber, Evolutionary multiobjective route planning in dynamic multi-hop ridesharing, in: Evolutionary Computation in Combinatorial Optimization, EVOCOP 2011, Lecture Notes in Computer Science, vol. 6622, Springer, 2011, pp. 84–95, https://doi.org/10.1007/978-3-642-20364-0_8

[5] M. Ozmen, H. Sahin, Real-time optimization of school bus routing problem in smart cities using genetic algorithm, in: 2021 6th International Conference on Inventive Computation Technologies (ICICT), IEEE, 2021, pp. 1152–1158, https://doi.org/10.1109/ICICT50816.2021.9358666

[6] J. Wang, Y. Zhang, X. Xing, Y. Zhan, W.K.V. Chan, S. Tiwari, A data-driven system for cooperative-bus route planning based on generative adversarial network and metric learning, Ann. Oper. Res. 339 (1–2) (2022) 1–27, https://doi.org/10.1007/s10479-022-04842-w

[7] S. Venkatachalam, K. Sundar, S. Rathinam, A two-stage approach for routing multiple unmanned aerial vehicles with stochastic fuel consumption, Sensors 18 (11) (2018) 3756, https://doi.org/10.3390/s18113756

[8] L.P. Behnck, D. Doering, C.E. Pereira, A. Rettberg, A modified simulated annealing algorithm for suavs path planning, IFAC-PapersOnLine 48 (10) (2015) 63–68, https://doi.org/10.1016/j.ifacol.2015.08.109

[9] O. Cheikhrouhou, I. Khoufi, A comprehensive survey on the multiple traveling salesman problem: applications, approaches and taxonomy, Comput. Sci. Rev. 40 (2021) 100369, https://doi.org/10.1016/j.cosrev.2021.100369

[10] T. Kalmár-Nagy, G. Giardini, B.D. Bak, The multiagent planning problem, Complexity 2017 (2017) 1–12, https://doi.org/10.1155/2017/3813912

[11] T. Bektas, The multiple traveling salesman problem: an overview of formulations and solution procedures, Omega 34 (3) (2006) 209–219, https://doi.org/10.1016/j.omega.2004.10.004

[12] A. Maktabifard, D. Földes, B.D. Bak, Constrained multi-agent path planning problem, in: Computational Logistics, ICCL 2023, Lecture Notes in Computer Science, vol. 14239, Springer, 2023, pp. 450–466, https://doi.org/10.1007/978-3-031-43612-3_28

[13] J. Gao, Y. Li, Y. Xu, S. Lv, A two-objective ILP model of op-matsp for the multi-robot task assignment in an intelligent warehouse, Appl. Sci. 12 (10) (2022) 4843, https://doi.org/10.3390/app12104843

[14] M. Assaf, M. Ndiaye, Solving an open path multiple depot multiple traveling salesman problem after transformation, in: 2017 7th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO), IEEE, 2017, pp. 1–4, https://doi.org/10.1109/ICMSAO.2017.7934866

[15] I. Kara, T. Bektas, Integer linear programming formulations of multiple salesman problems and its variations, Eur. J. Oper. Res. 174 (3) (2006) 1449–1458, https://doi.org/10.1016/j.ejor.2005.03.008

[16] M.M. Aguayo, F.N. Avilés, S.C. Sarin, H.D. Sherali, A two-index formulation for the fixed-destination multi-depot asymmetric travelling salesman problem and some extensions, Informatica 33 (2022) 671–692, https://doi.org/10.15388/22-INFOR485

[17] A.I. Ali, J.L. Kennington, The asymmetric m-travelling salesmen problem: a duality based branch-and-bound algorithm, Discrete Appl. Math. 13 (2–3) (1986) 259–276, https://doi.org/10.1016/0166-218X(86)90087-9

[18] T. Bektaş, L. Gouveia, D. Santos, New path elimination constraints for multi-depot routing problems, Networks 70 (3) (2017) 246–261, https://doi.org/10.1002/net.21760

[19] G. Laporte, Y. Nobert, A cutting planes algorithm for the m-salesmen problem, J. Oper. Res. Soc. 31 (11) (1980) 1017–1023, https://doi.org/10.1057/jors.1980.188

[20] Y. Harrath, A.F. Salman, A. Alqaddoumi, H. Hasan, A. Radhi, A novel hybrid approach for solving the multiple traveling salesmen problem, Arab. J. Basic Appl. Sci. 26 (1) (2019) 103–112, https://doi.org/10.1080/25765299.2019.1565193

[21] H. Zhou, M. Song, W. Pedrycz, A comparative study of improved ga and PSO in solving multiple traveling salesmen problem, Appl. Soft Comput. 64 (2018) 564–580, https://doi.org/10.1016/j.asoc.2017.12.031

[22] P. Singamsetty, J. Thenepalle, An efficient genetic algorithm for solving open multiple travelling salesman problem with load balancing constraint, Decis. Sci. Lett. 10 (4) (2021) 525–534, https://doi.org/10.5267/j.dsl.2021.5.003

[23] S. Yuan, B. Skinner, S. Huang, D. Liu, A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms, Eur. J. Oper. Res. 228 (1) (2013) 72–82, https://doi.org/10.1016/j.ejor.2013.01.043

[24] S. Linganathan, P. Singamsetty, Genetic algorithm to the bi-objective multiple travelling salesman problem, Alex. Eng. J. 90 (2024) 98–111, https://doi.org/10.1016/j.aej.2024.01.048

[25] E. Osaba, E. Onieva, F. Diaz, R. Carballedo, P. Lopez, A. Perallos, An asymmetric multiple traveling salesman problem with backhauls to solve a dial-a-ride problem, in: 2015 IEEE 13th International Symposium on Applied Machine Intelligence and Informatics (SAMI), IEEE, 2015, pp. 151–156, https://doi.org/10.1109/SAMI.2015.7061865

[26] P. Lou, K. Xu, J. Yan, Z. Xiao, An improved partheno-genetic algorithm for open path multi-depot multiple traveling salesmen problem, J. Phys. Conf. Ser. IOP Publishing 1848 (2021) 012002, . https://doi.org/10.1088/1742-6596/1848/1/012002

[27] L.-C. Lu, T.-W. Yue, Mission-oriented ant-team aco for min–max mtsp, Appl. Soft Comput. 76 (2019) 436–444, https://doi.org/10.1016/j.asoc.2018.11.048

[28] S. Ghafurian, N. Javadian, An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesmen problems, Appl. Soft Comput. 11 (1) (2011) 1256–1262, https://doi.org/10.1016/j.asoc.2010.03.002

[29] S. Wang, Y. Liu, Y. Qiu, Q. Zhang, F. Huo, Y. Huangfu, C. Yang, J. Zhou, Cooperative task allocation for multi-robot systems based on multi-objective ant colony system, IEEE Access 10 (2022) 56375–56387, https://doi.org/10.1109/ACCESS.2022.3165198

[30] P. Venkatesh, A. Singh, Two metaheuristic approaches for the multiple traveling salesperson problem, Appl. Soft Comput. 26 (2015) 74–89, https://doi.org/10.1016/j.asoc.2014.09.029
[31] X. Dong, Q. Lin, M. Xu, Y. Cai, Artificial bee colony algorithm with generating neighbourhood solution for large scale coloured traveling salesman problem, IET Intell. Transp. Syst. 13 (10) (2019) 1483–1491, https://doi.org/10.1049/iet-its.2018.5359
[32] C. Wei, Z. Ji, B. Cai, Particle swarm optimization for cooperative multi-robot task allocation: a multi-objective approach, IEEE Rob. Autom. Lett. 5 (2) (2020) 2530–2537, https://doi.org/10.1109/LRA.2020.2972894
[33] T. Matsuura, K. Numata, Solving min-max multiple traveling salesman problems by chaotic neural network, in: International Symposium on Nonlinear Theory and Its Applications, IEICE, 2014, pp. 237–240, https://doi.org/10.34385/proc.46.B1L-C1
[34] H. Qu, Z. Yi, H. Tang, A columnar competitive model for solving multi-traveling salesman problem, Chaos Solitons Fractals 31 (4) (2007) 1009–1019, https://doi.org/10.1016/j.chaos.2005.10.059
[35] H. Yapicioglu, Multiperiod multi traveling salesmen problem considering time window constraints with an application to a real world case, Netw. Spat. Econ. 18 (4) (2018) 773–801, https://doi.org/10.1007/s11067-017-9367-9
[36] F. Deng, Y. Wei, Y. Liu, B. Yan, J. Wu, Y. Cui, S. Pan, Distribution route optimisation of emergency supplies based on improved grey wolf optimisation algorithm, Int. J. Logist. Res. Appl. (2023) 1–17, https://doi.org/10.1080/13675567.2023.2278479
[37] Y. Zhang, X. Han, Y. Dong, J. Xie, G. Xie, X. Xu, A novel state transition simulated annealing algorithm for the multiple traveling salesmen problem, J. Supercomput. 77 (10) (2021) 11827–11852, https://doi.org/10.1007/s11227-021-03744-1
[38] K. Karabulut, H. Öztop, L. Kandiller, M.F. Tasgetiren, Modeling and optimization of multiple traveling salesmen problems: an evolution strategy approach, Comput. Oper. Res. 129 (2021) 105192, https://doi.org/10.1016/j.cor.2020.105192
[39] E. Ergüven, F. Polat, Relative distances approach for multi-traveling salesmen problem, Knowl.-Based Syst. 300 (2024) 112160, https://doi.org/10.1016/j.knosys.2024.112160
[40] X. Wang, D. Liu, M. Hou, A novel method for multiple depot and open paths, multiple traveling salesmen problem, in: 2013 IEEE 11th International Symposium on Applied Machine Intelligence and Informatics (SAMI), IEEE, 2013, pp. 187–192, https://doi.org/10.1109/SAMI.2013.6480972
[41] F.H. Essani, S. Haider, An algorithm for mapping the asymmetric multiple traveling salesman problem onto colored petri nets, Algorithms 11 (10) (2018) 143, https://doi.org/10.3390/a11100143
[42] G. Giardini, T. Kalmar-Nagy, Genetic algorithm for multi-agent space exploration, in: AIAA Infotech@ Aerospace 2007 Conference and Exhibit, AIAA, 2007, pp. 2824, https://doi.org/10.2514/6.2007-2824
[43] R. Diestel, Graph Theory, fifth ed., Springer Berlin, Heidelberg, 2017, https://doi.org/10.1007/978-3-662-53622-3
[44] J. Bang-Jensen, G.Z. Gutin, Digraphs: Theory, Algorithms and Applications, second ed., Springer London, 2008, https://doi.org/10.1007/978-1-84800-998-1
[45] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, first ed., Addison-Wesley Publishing Co., Inc., USA, 1989.
[46] J. Cai, G. Thierauf, Evolution strategies for solving discrete optimization problems, Adv. Eng. Softw. 25 (2–3) (1996) 177–183, https://doi.org/10.1016/0965-9978(95)00104-2
[47] G. Srivastava, P. Venkatesh, A. Singh, An evolution strategy based approach for cover scheduling problem in wireless sensor networks, Int. J. Mach. Learn. Cybern. 11 (9) (2020) 1981–2006, https://doi.org/10.1007/s13042-020-01088-5
[48] G.A. Croes, A method for solving traveling-salesman problems, Oper. Res. 6 (6) (1958) 791–812, https://doi.org/10.1287/opre.6.6.791