A Genetic Algorithm Approach to Multi-Agent Mission Planning Problems*

Branko Miloradović, Baran Çürüklü, Mikael Ekström, and Alessandro V. Papadopoulos[0000−0002−1364−8127]

Mälardalen University, School of Innovation, Design and Engineering,
Högskoleplan 1, 721 23 Västerås , Sweden
{branko.miloradovic,baran.curuklu,mikael.ekstrom,alessandro.papadopoulos}@mdh.se

Abstract. Multi-Agent Systems (MASs) have received great attention from scholars and engineers in different domains, including computer science and robotics. MASs try to solve complex and challenging problems (e.g., a mission) by dividing them into smaller problem instances (e.g., tasks) that are allocated to the individual autonomous entities (e.g., agents). By fulfilling their individual goals, they lead to the solution to the overall mission. A mission typically involves a large number of agents and tasks, as well as additional constraints, e.g., coming from the required equipment for completing a given task. Addressing such problem can be extremely complicated for the human operator, and several automated approaches fall short of scalability. This paper proposes a genetic algorithm for the automation of multi-agent mission planning. In particular, the contributions of this paper are threefold. First, the mission planning problem is cast into an Extended Colored Traveling Salesperson Problem (ECTSP), formulated as a mixed integer linear programming problem. Second, a precedence constraint reparation algorithm to allow the usage of common variation operators for ECTSP is developed. Finally, a new objective function minimizing the mission makespan for multi-agent mission planning problems is proposed.

Keywords: Multi-agent Systems, Multi-agent Mission Planning, Extended Colored Traveling Salesperson (ECTSP), Genetic Algorithms.

1 Introduction

The popularity of the multi-agent systems (MAS) keeps increasing due to advances in key technologies such as, sensors, energy sources, computing units, and wireless communication, among others [10]. The immediate consequences of these advances are improved price-performance ratio, diverse product range, and availability of components, and finally, improved complete systems. This in turn pushes advances in application domains, which results in a pull effect, resulting in better technologies. A recent example in this regard is the Unmanned Aerial Vehicle technologies (UAV), and applications, such as inspection of critical infrastructures, monitoring construction sites and natural

* This work was supported by the project Aggregate Farming in the Cloud (AFarCloud) European project, with project number 783221 (Call: H2020-ECSEL-2017-2), and by the Knowledge Foundation with the FIESTA project.

environments, also critical applications such as search and rescue [44]. However, the airborne problem domain is not the only one where MASs are used. Ground domain (e.g., search and rescue missions [12]) and underwater domain (e.g., seabed mapping [24]) missions benefit widely from the use of MASs.

In this context, planning consists of assigning all the tasks in a mission to agents in such a way that the plan is feasible, given a global mission objective, resources, and constraints. As in many cases, agents can have different equipment, e.g., in terms of sensors, actuators, or batteries. This results in a heterogeneous MAS configuration. Tasks in a plan can have precedence constraints (PC) in addition to equipment, or sensor requirements. The planning problem may also include choosing the optimal set of agents for the mission. Task precedence is only considered within the agent's plan, i.e., one agent cannot execute a task that precedes a task assigned to another agent. This problem is modeled as a novel TSP variation formally proposed in this paper.

This type of combinatorial problem can be solved by an exact method or with a meta-heuristic approach. Exact methods guarantee an optimal solution, usually by mapping a problem into a tree or a graph. Later, search through the nodes, prune infeasible branches and backtrack from dead-ends. Meta-heuristics solve problems differently, which sometimes leads to a sub-optimal solution. In addition, as the search space increases most planning approaches fail to produce a plan within a reasonable time. This is especially important in scenarios that require immediate access to the plan for the progress of the mission. In the general case, however, the time taken for an initial plan to be produced is less critical compared to that of re-planning. If re-planning takes too long, the state of the MAS can change while the planning process is being done. In these cases, where the necessity for a feasible plan outweighs the need for optimality, sub-optimal planners are usually preferred. The planner proposed in this work is based on a Genetic Algorithm (GA) [16], and has been adapted to the specific problem of heterogeneous multi-agent mission planning, which is the focal point of this paper.

In this paper, the problem of mission planning for MAS is tackled. The main contributions of this paper are to: (i) Cast the multi-agent mission planning problem to a novel Extended Colored Traveling Salesperson Problem (ECTSP), and formulate such problem as a Mixed-Integer Linear Programming (MILP) problem; (ii) Develop a reparation algorithm to allow usage of common variation operators for ECTSP; and (iii) Propose a new optimization criterion useful for multi-agent mission planning problems. Furthermore, examples from the applications of Autonomous Underwater Vehicles (AUVs) and some randomly generated generic multi-agent missions are used in order to illustrate different contributions.

A preliminary version of this work appeared in Miloradović *et al*. [34]. Additional contributions are: (i) the precedence constraint representation within the MILP problem, following the paradigm of two-commodity network flow model, (ii) the description of the proposed objective function has been further detailed and analyzed, and (iii) the results section has been enriched with a more extensive evaluation of the proposed solution, also in terms of cost and required computational resources.

In Section 2 an overview of the related work is given. A formal problem formulation is given in Section 3. The genetic solver is explained in Section 4. Results are presented in Section 5, and finally Section 6, concludes the paper.

2 Related Work

In order to classify MAS problems, Gerkey and Matarić [13] proposed a domain independent taxonomy with three decoupled axis for multi-robot task allocation (MRTA) problems. Authors make several assumptions in the paper, however, the most important one is that tasks are assumed to be independent of each another. This means that there are no ordering constraints, synchronization or any other interrelatedness between tasks. In order to overcome this restriction, Korsah *et al*. [21] extended MRTA with the degree of interrelatedness. Later, Nunes *et al*. [38] decomposed this dimension further into synchronization and precedence constraints. The term *synchronization constraints* (SC) is used when there are temporal relations between tasks and *precedence constraints* (PC) when there are only ordering relations. PC can be seen as a subset of SC, however the focus, in this paper, will be explicitly on PC. In Operations Research these constraints are usually indicated as General Precedence Relationships [35]. The TSP variation that will be described, and addressed in this paper, can be labeled as Single-Task robots, Single-Robot tasks, Time-Extended Assignments with Synchronization and Precedence (ST-SR-TA-PC) with robots being heterogeneous.

The aforementioned task allocation is solved by using planning algorithms (planners). A planner breaks a mission plan into smaller pieces, tasks, that are sent to the appropriate agent for execution, i.e., it does the allocation of tasks to appropriate agents. Most of the approaches presented here are domain independent or can be easily translated into domain independent solutions, thus can be used in different scenarios, such as a group of Autonomous Underwater Vehicles (AUVs) or ground vehicles.

The mission planning problem for a swarm of AUVs without task interrelatedness can be solved using multi-objective harmony search algorithm [24]. A research framework on mission planning for swarms of UAVs has been proposed by Zhou *et al*. [48]. The problem of mission planning for a swarm of UAVs can be solved using the algorithm approach as shown in [40,42]. The problem is modeled as a constraint satisfaction problem and solved using multi-objective GA. This work has been further extended to utilize re-planning and analysis of operator training in the control center [43]. For a similar problem of a mission planning for cooperative UAV teams, a solution was proposed by Bello-Orgaz *et al*. [4], that use GA with a weighted linear combination of the mission's makespan and the fuel consumption as the optimization criterion. This approach was further improved by Ramirez-Atencia *et al*. [41], by using weighted random generator strategies for the creation of new individuals. All of the problems in these papers can be seen as variations of the TSP.

Khamis *et al*. [20] presented a survey on MRTA problems, connecting the planning problem to the multiple TSP problem. In addition, the authors provide a comprehensive overview of centralized and decentralized approaches for solving MRTA. The original TSP formulation is expressed as an integer linear program and it was introduced by Dantzig and colleagues [9]. By this day many different approaches were developed and proposed in order to solve TSP. In today's state of the art two inexact approaches are on average ahead of the rest Lin-Kernighan (LKH) [19] and GA with Edge Assembly Crossover (EAX) [46]. The original problem definition is later extended to an mTSP [3]. An approach using sub-tours was proposed by Giardini *et al*. [14] to solve multiple TSP (mTSP). The idea was to divide a graph into subgraphs which are solved using

GA. Each subgraph represents a tour for one of the salesmen. Another extension of the original TSP is done by adding Precedence Constraint (TSPPC) [22]. mTSP and TSPPC were later combined into an mTSPPC [47], although a formal problem formulation was not given.

A two-commodity flow formulation of TSP with Time Windows was given by Langevin *et al*. [11,25] solving problem instances of up to 60 cities. This formulation of the TSP is extended to handle precedence constraints [36] and solved by using GA. It is shown that other TSP/VRP variants can be extended in the similar manner as well [15,23]. Recently, serial [31], radial [26], and a more complete version of a Colored TSP (CTSP[1]) has been proposed by Meng *et al*. [30] in order to solve multiple bridge machine planning in industry.

In this paper, their approach has been further generalized by adding Two-Commodity Flow model for PC, multi depots (with different starting and ending points). This means that the solution to the problem is a Hamiltonian path and not a cycle, thus ECTSP is an open variant of TSP.

3 ECTSP Formulation

Before introducing the theoretical background of the ECTSP, the connection between the theoretical model and multi-agent mission will be explained in more detail.

3.1 Mapping between MRTA and ECTSP

In MRTA, cities and salespersons correspond to tasks and agents, respectively. A salesperson's colors map to an equipment type of an agent, e.g., camera, gripper, and different types of sensory equipment. Following the same rules, a color, which is associated with a task, indicates the required equipment, i.e., equipment an agent should have in order to successfully complete that task. In contrast to the classical TSP formulation, tasks are neither instantaneous nor have a predefined duration. The duration of a task is estimated based on the agent's capabilities. Moreover, equipment heterogeneity is not the sole determinant for the differences between the agents. Every agent may have a different speed, therefore the duration of task execution may depend on the selected agent. In addition to equipment requirements and duration, tasks may have an additional parameter, i.e., precedence constraints. Some tasks might need to be completed before or after some other task in a mission. Precedence constraints in ECTSP are essentially an ordering constraint. This means that interrelated tasks will be allocated to the same agent. Following Korsah's definition [21], it can be concluded that ECTSP has only intra-schedule dependencies. Although actions are durative, this problem in its core is temporally simple [8], i.e., although actions have a duration, actions are ordered, and no concurrent actions are possible.

For example, Fig. 1 shows two salespersons starting from two different source depots (σ_1 and σ_2, respectively). Each of them has two different colors and visits a certain number of tasks with matching colors. Every salesperson ends its tour in the destination

[1] CTSP is an abbreviation used in the literature for a Clustered TSP as well

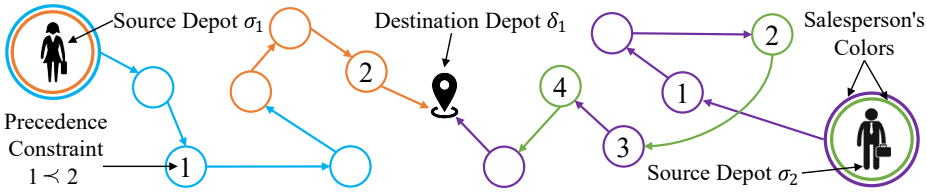


Fig. 1: An Illustration of the ECTSP.

depot δ_1. Cities that have precedence constraints are marked with numbers in the figure (1 has to be visited before 2, i.e., $1 \prec 2$). The need for precedence constraint is apparent, as explained in this multi-agent mission example: Let's assume that an agent has two tasks, e.g., survey an area with camera for data collection, and send data back. Obviously, these two tasks are interrelated—the data cannot be sent before it is acquired. Thus the only possible ordering between the two tasks is to gather the data first, before sending it to the receiving destination. Note, further, that there is no constraint on the delay between the end of the first task, and the start of the second task.

3.2 Problem Formulation

Suppose that an ECTSP has m salespersons, $s \in \mathcal{S} := \{s_1, s_2, \ldots, s_m\}$, n cities, $v \in \mathcal{V} := \{v_1, v_2, \ldots, v_n\}$, k colors, $c \in \mathcal{C} := \{c_1, c_2, \ldots, c_k\}$, q source depots $\sigma \in \Sigma := \{\sigma_1, \ldots, \sigma_q\}$, and w destination depots, $\delta \in \Delta := \{\delta_1, \ldots, \delta_w\}$ where $m, n, k, q, w \in \mathbb{N}^+$. Each salesperson s starts from a source depot σ and finishes its tour at a destination depot δ. Source and destination depots are not considered to be cities. The superset containing all of the cities $\mathcal{V}$ and depots is defined as $\widetilde{\mathcal{V}} := \mathcal{V} \cup \{\Sigma, \Delta\}$. In addition, for the simplicity, a superset containing all source depot and city elements is defined as $\mathcal{V}^\Sigma := \mathcal{V} \cup \Sigma$. In the same manner a superset containing all elements of destination depot and cities is defined as $\mathcal{V}^\Delta := \mathcal{V} \cup \Delta$. This problem can be formulated over a directed graph $\mathcal{G} = (\widetilde{\mathcal{V}}, \mathcal{E})$, where $\mathcal{E} : \widetilde{\mathcal{V}} \times \widetilde{\mathcal{V}} \mapsto \mathbb{R}_0^+$. An edge $e \in \mathcal{E}$, connecting vertexes $i, j \in \widetilde{\mathcal{V}}$ can be expressed as

$$e(i,j) = \begin{cases} \omega_{ij}, & \text{if } i \text{ is connected to } j \\ 0, & \text{otherwise,} \end{cases}$$

where $\omega_{ij} \geq 0$ represents the cost of edge $e(i,j)$. The decision variable $x_{ijs} \in \{0,1\}$ can be defined as

$$x_{ijs} = \begin{cases} 1, & \text{if } s \in \mathcal{S} \text{ travels from } i \in \widetilde{\mathcal{V}} \text{ to } j \in \widetilde{\mathcal{V}}, \\ 0, & \text{otherwise.} \end{cases}$$

Every city $i \in \mathcal{V}^\Sigma$ has a weight $\xi(i)$, $\xi : \mathcal{V}^\Sigma \mapsto \mathbb{R}_0^+$ (with $\xi(i) = 0$ when $i \in \Sigma$). Also, every city $i \in \mathcal{V}$ is associated with a color $f_c(i)$, with $f_c : \mathcal{V} \mapsto \mathcal{C}$. Each salesperson $s \in \mathcal{S}$ has a set of colors $\mathcal{C}_s \subseteq \mathcal{C}$ assigned to it—source and destination depots do not have colors. In contrast to city color matrix that was defined by Li *et al.* [26], here

a color matrix of a salesperson s, $\mathcal{A}_s \in \{0,1\}^{n\times n}$, shows openness of cities towards a salesperson s, and is defined as $\mathcal{A}_s := [a_{ijs}]$, with

$$a_{ijs} = \begin{cases} 1, & f_c(v_i) \in \mathcal{C}_s \wedge f_c(v_j) \in \mathcal{C}_s \wedge \pi_{ij} = 1 \\ 0, & \text{otherwise,} \end{cases}$$

where $\Pi = [\pi_{ij}]_{n\times n}$ is the adjacency matrix indicating the precedence relations among the cities, where $\pi_{ij} = 1 \iff i \prec j$, and 0 otherwise. The definition of the color matrix $\mathcal{A}_s$ can be extended to include the depots as:

$$\overline{a}_{ijs} = \begin{cases} a_{ijs}, & i,j \in \mathcal{V}, \\ 1, & (i \in \Sigma, j \in \mathcal{V}^{\Delta}) \vee (i \in \mathcal{V}^{\Sigma}, j \in \Delta), \\ 0, & (i,j \in \Sigma) \vee (i,j \in \Delta). \end{cases}$$

A salesperson s is allowed to only visit the cities specified in its extended color matrix $\mathcal{A}_s$:

$$x_{ijs} \leq \overline{a}_{ijs}, \qquad \forall i \in \mathcal{V}^{\Sigma}, \forall j \in \mathcal{V}^{\Delta}, \forall s \in \mathcal{S}, i \neq j. \tag{1}$$

Furthermore, only one salesperson $s \in \mathcal{S}$ can enter (Eq. 2) and leave (Eq. 3) each:

$$\sum_{s\in\mathcal{S}} \sum_{i\in\mathcal{V}^{\Sigma}} x_{ijs} = 1, \qquad \forall j \in \mathcal{V}, i \neq j, \tag{2}$$

$$\sum_{s\in\mathcal{S}} \sum_{j\in\mathcal{V}^{\Delta}} x_{ijs} = 1, \qquad \forall i \in \mathcal{V}, i \neq j. \tag{3}$$

The final destination of a salesperson s is always a destination depot:

$$\sum_{i\in\mathcal{V}^{\Sigma}} \sum_{j\in\Delta} x_{ijs} = 1, \qquad \forall s \in \mathcal{S}. \tag{4}$$

Note that some salespersons can go directly from a source depot to a destination depot, i.e., $x_{ijs} = 1, i \in \Sigma, j \in \Delta$. This means that the salesperson is not used in the final plan.

The starting location of a salesperson s is always a source depot:

$$\sum_{i\in\Sigma} \sum_{j\in\mathcal{V}^{\Delta}} x_{ijs} = 1, \qquad \forall s \in \mathcal{S}. \tag{5}$$

The number of salespersons $\mathcal{B}_\sigma$ in each source depot $\sigma \in \Sigma$ is given, and it is such that $\sum_{i\in\Sigma} \mathcal{B}_i = |\mathcal{S}|$, and:

$$\sum_{s\in\mathcal{S}} \sum_{j\in\mathcal{V}^{\Delta}} x_{ijs} = \mathcal{B}_i, \qquad \forall i \in \Sigma, \tag{6}$$

And it defines the initial deployment of the salespersons over the source depots. In order to ensure that the same agent enters and exits a certain city:

$$\sum_{i\in\mathcal{V}^{\Sigma}} x_{ijs} = \sum_{k\in\mathcal{V}^{\Delta}} x_{jks}, \qquad \forall j \in V, \forall s \in S. \tag{7}$$

A salesperson s cannot travel from a city i to the same city i:

$$x_{iis} = 0, \qquad \forall i \in \widetilde{\mathcal{V}}, \forall s \in \mathcal{S}. \tag{8}$$

The description above concludes the formulation of the Colored TSP with multiple source and destination depots, and heterogeneous salespersons. The extension with intra-scheduled dependencies is presented below by using Two-commodity Network Flow (TNF) paradigm.

3.3 Two-commodity network flow model for precedence constraints

As a simple example, assume that salesperson s delivers a full container of some commodity (e.g., a bottle of gas), and picks up an empty container, for each customer. At all times, the salesperson will have a total of n containers (full + empty). A salesperson s has two distinct commodities $y_{ijs}, z_{ijs} \in \mathbb{N}^0$, when moving from city i to city j, in the network with n_s number of nodes (cities), with

$$n_s = \sum_{i \in \mathcal{V}^\Sigma} \sum_{j \in \mathcal{V}} x_{ijs}, \qquad \forall s \in \mathcal{S}. \tag{9}$$

The commodity y_{ijs} is supplied by n_s units at a selected starting node, $i \in \Sigma$, and consumed by one unit at each node that is not the starting node. On the other hand, the commodity z_{ijs} is consumed by n_s units at the starting node, and supplied by one unit at the other nodes. This kind of network flow of commodities is described by two properties:

(i) Sum of all commodities y_{ijs} and z_{ijs} in any feasible tour must be equal to n_s
(ii) The quantity of commodity y_{ijs} exiting from a node is decreasing by one unit, while the quantity of commodity z_{ijs} is increasing by one unit as the tour proceeds.

The flow conservation equations for the commodity y_{ijs} and z_{ijs}, are defined as follows:

$$\sum_{j \in \mathcal{V}^\Delta} y_{ijs} - \sum_{j \in \mathcal{V}^\Sigma} y_{jis} = -\sum_{j \in \mathcal{V}^\Delta} x_{ijs}, \qquad \forall i \in \mathcal{V}, \forall s \in \mathcal{S}. \tag{10}$$

$$\sum_{j \in \mathcal{V}^\Delta} z_{ijs} - \sum_{j \in \mathcal{V}^\Sigma} z_{jis} = \sum_{j \in \mathcal{V}^\Delta} x_{ijs}, \qquad \forall i \in \mathcal{V}, \forall s \in \mathcal{S}. \tag{11}$$

In addition to the flow conservation equations (Eq. 10 and Eq. 11), it is necessary to define the flow conservation equations for the edges going from the set Σ of source depots.

$$\sum_{i \in \Sigma} \sum_{j \in \mathcal{V}} y_{ijs} - \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} y_{ijs} = \sum_{i \in \mathcal{V}^\Sigma} \sum_{j \in \mathcal{V}} x_{ijs}, \qquad \forall s \in \mathcal{S}, \tag{12}$$

$$\sum_{i \in \Sigma} \sum_{j \in \mathcal{V}} z_{ijs} - \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} z_{ijs} = -\sum_{i \in \mathcal{V}^\Sigma} \sum_{j \in \mathcal{V}} x_{ijs}, \qquad \forall s \in \mathcal{S}. \tag{13}$$

Finally, if node i must precede node j precedence constraints can be imposed as follows:

$$\sum_{k \in \mathcal{V}} y_{iks} \geq \sum_{l \in \mathcal{V}} y_{jls}, \qquad \forall i, j \in \mathcal{V}, \forall s \in \mathcal{S} : i \prec j, \tag{14}$$

$$\sum_{k \in \mathcal{V}} z_{iks} \leq \sum_{l \in \mathcal{V}^{\Delta}} z_{jls}, \qquad \forall i, j \in \mathcal{V}, \forall s \in \mathcal{S} : i \prec j. \tag{15}$$

Finally, we introduce big-M constraints:

$$y_{ijs} \leq M \cdot x_{ijs}, \qquad \forall i \in \mathcal{V}^{\Sigma}, \forall j \in \mathcal{V}^{\Delta}, \forall s \in \mathcal{S}, \tag{16}$$

$$z_{ijs} \leq M \cdot x_{ijs}, \qquad \forall i \in \mathcal{V}^{\Sigma}, \forall j \in \mathcal{V}^{\Delta}, \forall s \in \mathcal{S}. \tag{17}$$

Let us assume that node i precedes node j ($i \prec j$), then three different precedence relationship cases between nodes can be defined:

1. Node j must be visited exactly β nodes after node i,
2. Node j must be visited in less than β nodes after node i,
3. Node j must be visited in more than β nodes after node i.

For the simplicity, and without loss of generality, the focus will be only on the case 3, with β equal to 1, described in (Eq. 14) and (Eq. 15). This implies that city j can be visited any time, as long as it is after city i.

If there is a constraint that implies that node j must be visited immediately after node i, then this is the case 1 where β is equal to 1.

In this case, edge contraction can be used to reduce the size of a graph. For a graph $G = (V, E)$ and an edge $e = (i, j) \in E$, $i \neq j$, the contraction of e in G is an operation that results in the graph $G' = (V', E')$, where the resulting graph G' has one less edge than G. If S defines a set of edges that are incident to vertices i and j, which are replaced with a single vertex v, then $S' = \{S \setminus e\}$ is a set of edges incident to v. After edge contraction, the weight of the new vertex is a sum of weights i and j, added to a weight associated with the cost ω_{ij} of going from i to j, i.e., $\xi_x = \xi_i + \xi_j + \omega_{ij}$. The same approach can be used in the case of n consecutive nodes with PC.

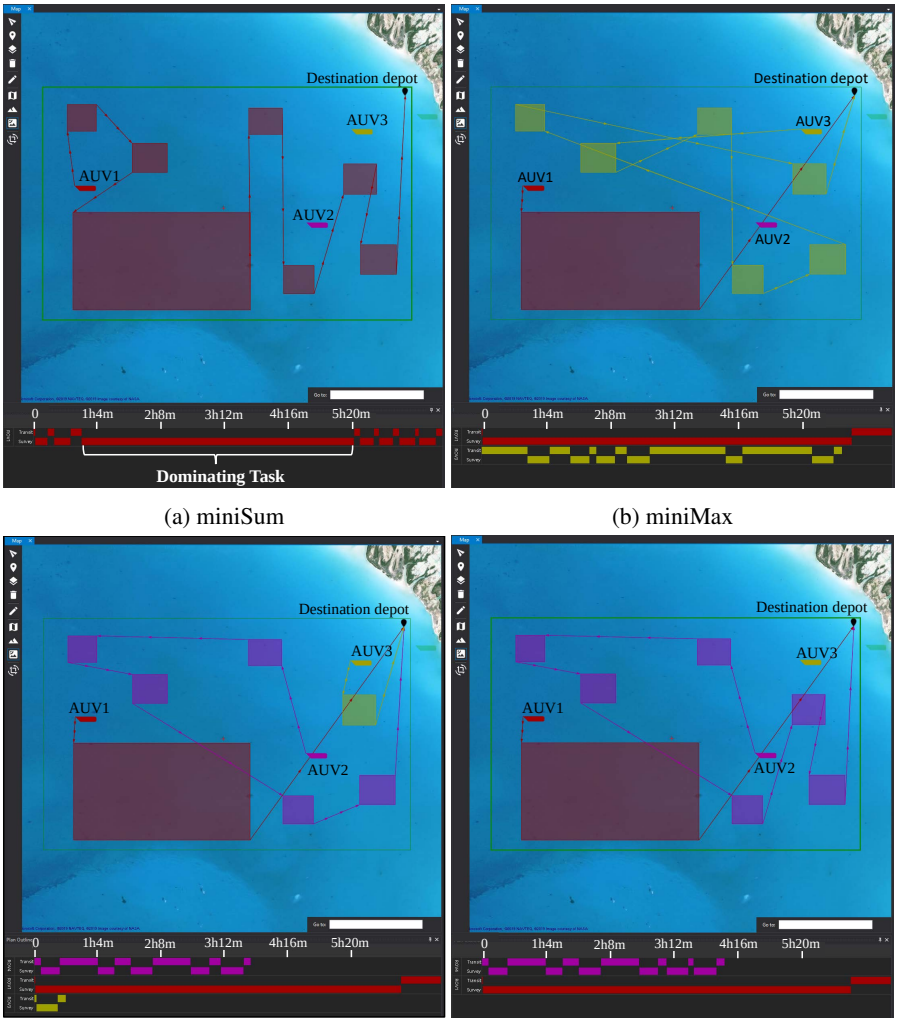
Few assumptions are made in this model. Tasks with ordering constraints have the same color, i.e., the same salesperson can perform these tasks. In addition, tasks can only have one PC relation. Ordering constraints cannot be shared between the salesmen, i.e., if node i precedes node j, both nodes are visited by the same salesperson s. Omniscience is assumed, as well as atomic tasks, and deterministic effects.

3.4 Objective Function

MAS is a structure composed of multiple interacting agents, in which the agents fulfill various goals, or have specific purposes. In the context of this work, there is an overall goal that will be achieved by an agent population when they perform the tasks that are assigned to them. This overall goal is mathematically represented with a function commonly known as the objective function, that must be either maximized or minimized. An overview of the most common objective functions in MRTA problems is given by

Nunes *et al*. [38]. As can be seen in this work, a mission can have many different objective functions, however, the most common optimization criterion is the minimization with respect to time, i.e., the total mission duration. Other common optimization criteria are the minimization of the path, or energy consumption.

The duration of a mission is a straightforward concept in terms of single-agent missions. However, a mission duration can be defined in many different ways in the context



(a) miniSum (b) miniMax

(c) miniMax + miniAvg (d) miniMax + miniSum

Fig. 2: Different planning outputs based on four different objective functions. Identical sets of AUVs are assumed in all four cases. Bottom part of the figures shows the complete plans represented as Gantt charts.

of the MAS. A mission management tool is used for defining missions to be planned with the planner presented in Sect. 4. Fig. 2 shows the obtained plans for different objective functions described below.

One definition is the sum of all tasks in a mission (miniSum) [28]:

$$f_s = \min_x \sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{V}^\Sigma} \sum_{j \in \mathcal{V}^\Delta} (\omega_{ij} + \xi(i)) x_{ijs}. \tag{18}$$

The interval from the start of the first task to the end of the last task (miniMax) is another common definition [4]:

$$f_m = \min_x \max_s \sum_{i \in \mathcal{V}^\Sigma} \sum_{j \in \mathcal{V}^\Delta} (\omega_{ij} + \xi(i)) x_{ijs}. \tag{19}$$

Nonetheless, neither of these approaches is suitable for the problem presented in this paper. The former approach can produce a plan where one agent is doing much more work than the other agents (see Fig. 2a), where only one agent (indicated in red) is used and it covers all tasks. In an actual underwater mission involving AUVs, this can mean that the extraction vessels and the crew have to stay in the open sea for a longer time, thus increasing the overall cost of the mission. The latter approach minimizes the maximum cost of an agent over all the agents. This approach works well if tasks are instantaneous or have the same duration. However, if there is a task that dominates all the other tasks, e.g., its duration is longer than the makespan of any other agent's plan, then the mission will not be optimized at all, as shown in Fig. 2b. It can be observed that AUV3's (yellow coloured agent) makespan is shorter than the duration of the one single task belonging to AUV2 (red coloured agent) (Fig. 2b, see the bottom part of the figure showing the Gantt chart representing the plan). In this case, the planner would try to minimize the longest plan belonging to the agents, and since that plan would consist of one very long task it wouldn't be possible for the planner to minimize the complete plan further. In addition, the other agents that have shorter makespans would just be ignored. This issue is partly mitigated in [1] where a linear aggregation function is used and the optimization criterion was to minimize the maximum and average task completion times (miniMax + miniAvg), and total idle times. This approach favors the usage of as many agents as possible, although it may mean increase in cost for every agent added to the solution. Such a solution is shown in Fig. 2c. Thus, the final plan includes all three available agents.

All these objective functions have their usefulness in different cases, or scenarios. However, as discussed above, they also have weaknesses. In order to address these problems, in this paper a new optimization criterion has been proposed (Eq. 20). It is defined as a weighted combination of miniSum (Eq. 18) and miniMax (Eq. 19):

$$J = w_1 f_m + w_2 f_s, \tag{20}$$

subject to constraints from (1) to (17), where $w_1, w_2 > 0$ are user defined weights. How this objective function affects the final solution is shown in Fig. 2d.

4 Genetic Mission Planner (GMP)

Multi-agent missions can be conducted in a harsh and challenging environment (e.g., underwater, or airborne, missions) where many different problems may arise. Since some of these problems cannot be foreseen beforehand, changes in the mission, while it is being executed, are common. Although the initial plan making is not bounded by time, since no agent needs to be deployed, the re-planning is. Re-planning, in this context, can be seen as planning again with new initial conditions. Since multi-agent missions are usually costly and autonomy of agents is limited (especially in airborne missions), the re-planning process should be very fast. Since this is an NP-hard problem, exact solution algorithms do not scale well. This is one of the reasons for the use of a meta-heuristic approach over exact methods. GA and its numerous variations have been widely used for solving combinatorial optimization problems such as TSP [27,37,45], Vehicle Routing Problem (VRP) [29,2], job shop scheduling problems [5,17], and resource constrained problems [6,18].

In this paper, the encoding of chromosomes is done in a same manner as in Miloradović *et al*. [33]. Two arrays of integer identifiers are used to represent different types of genes. Task genes and agent genes are encoded in the first array, whereas the second array consists of task parameter genes (Fig. 3). Tasks have 4 parameters: (i) precedence relations, (ii) required equipment, (iii) task duration, and (iv) task location. The length of a chromosome can be fixed prior to the mission if the length of a plan is known, i.e., the number of tasks and the number of agents to be used is fixed. However, if a planner can introduce new tasks or decides not to use all available agents, variable chromosome length is preferred (Brie *et al*. [6]). In this paper, a mix of these two approaches is used, i.e., chromosome length is fixed, however, besides Active Genes (AG), i.e., tasks and agents, a certain number of inactive dummy genes is introduced. The reason for this is that, although the number of tasks in a mission is fixed, the planner is allowed to optimize the number of agents involved. This implies that the number of dummy genes is in the range from 0 to $(m-1)$, while the chromosome length is fixed to the size of $(n+m)$.

The initial population is generated with the respect to given constraints, however, the task allocation process is randomized. At the beginning of the search process, the minimum number of agents necessary for the successful completion of the mission is determined. Then a number of agents in the chromosome is randomly picked in the range of that determined minimum and the maximum number of available agents. As-

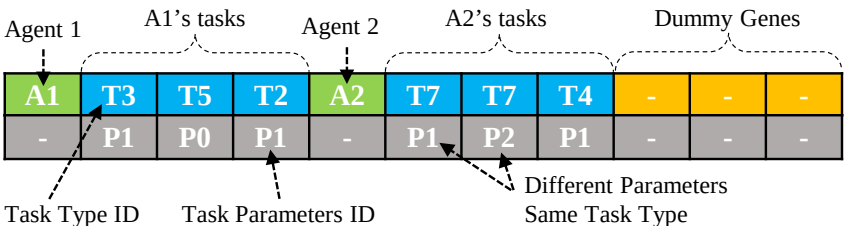


Fig. 3: An example of an individual in the population.

sume a multi-UAV mission during which patches of agricultural fields will be scanned through the execution of 10 tasks. Assume further that some of these tasks require a thermal camera, whereas others humidity sensors. In addition, there are three agents available, one with a thermal camera and the other two with humidity sensors. Then the number of agents per chromosome in the initial population will range from 2 to 3. After this step, a list of possible tasks for every agent is created. The number of tasks per agent is randomly determined based on the previously created list. Two or more agents may have the ability to do the same task, e.g., a task requires a thermal camera, and there are three agents with thermal camera available. In this specific case, a task is randomly allocated to one of the valid agents. Task allocation is repeated until there are no more tasks left to allocate and the whole process is repeated for every chromosome in the population.

4.1 Variation Operators

The two main variation operators of a GA are crossover and mutation. Various different types have been proposed for solving GA TSP [7] and VRP problems [39].

Crossover In a classical implementation of a GA for TSP, chromosomes are arrays consisting of genes (city identifiers). Crossover operators can mix individuals in many possible ways and produce feasible solutions. The only restriction is city duplicates in a potential solution.

N-point crossover operators usually produce offspring which violate given constraints, thus a repair procedure is usually required. This led to the development of an ordered based (OB) crossover operators, which tend to retain ordering from parents in the process of creating offspring like Partially Mapped Crossover (PMX). As a result, sub-tours are passed to the offspring without change. This type of operator creates a non-repeating sequence of cities thus, there is no need for a repairing procedure. Another crossover operator that shows promising results when applied to the TSP problem is Edge Recombination Crossover (ERX). The idea is to preserve edges between nodes from parents and pass them to the offspring. However, for the ECTSP, the abovementioned procedure is less trivial, since precedence and color constraints need to be taken into consideration, in order to avoid the creation of infeasible solutions.

Mutation The mutation is the source of variability as it allows genetic diversity in the population. Every individual has a certain probability to be selected for mutation. In this work, two types of mutation schemes are introduced. One operates on the task genes, whereas the other mutates agent genes.

Task gene mutation consists of swap gene and inserts gene mutation operations (Algorithm 1). Task swap mutation swaps two task genes in a chromosome, meaning that it can both swap tasks within a single agent or between two agents. In the first case, only the ordering of tasks changes. In the second case, tasks are exchanged between the two agents. This is done with respect to color constraints, i.e., tasks can be exchanged only between agents capable of performing those tasks. Insert mutation randomly chooses a

Algorithm 1 Task Swap/Insert Mutation

```
1: procedure TASKMUTATION(population)
2:    Select random chromosome i from the pop.
3:    Select random task gene from i
4:    case (1) - Task Swapping
5:      Create a list of valid tasks for swapping
6:      Choose a task gene randomly from that list
7:      Swap tasks
8:    case (2) - Task Insertion
9:      Create a list of suitable agents for chosen task
10:     Randomly select agent and position and insert gene from step 3.
11: return modified chromosome
```

task gene, removes it from its current location and inserts it in a new location in a chromosome. Similarly to the task gene mutation, the insertion can be within the same agent or to a different one. While performing insert mutation, color constraints are respected as in the previous case.

Agent genes mutate in two different ways, by adding agent genes to the chromosome (growth mutation) or by removing agent genes from the chromosome (shrink mutation) (Algorithm 2). Agent growth mutation adds a new agent gene to the plan if the limit of available agents is not reached in that specific chromosome. The new agent gene is inserted at a random location in the chromosome. After insertion of a new agent gene, all tasks from that location to the next agent gene, or end of the chromosome are allocated to the newly added agent. If the new agent may not be able to perform any of its tasks, randomly reallocation to other agents is performed, keeping the feasibility of the plan intact. Agent shrink mutation removes one agent from a chromosome, reallocating its tasks to other agents if possible, i.e., if such action do not break the feasibility of the plan. If such a scenario cannot be avoided the agent shrink mutation is skipped

Algorithm 2 Agent Growth/Shrink Mutation

```
1: procedure AGENTMUTATION(population)
2:    Select random chromosome i from the pop.
3:    case (1) - Agent Growth
4:      If there are available agents
5:        Select random position to insert new agent
6:        Validate that no constraint is violated
7:        Add new agent
8:        Check if new agent's plan is valid
9:        If not, reassign invalid tasks to other agents
10:   case (2) - Agent Shrink
11:     If the minimum is not reached
12:       Select random agent gene for removal
13:       Remove that gene
14:       Reassign its tasks to other agents with the respect to constraints
15: return modified chromosome
```

for that instance. Removing the only agent from a plan or removing the only agent with the required equipment for a specific task are examples in this regard.

Both gene and agent mutation algorithms consider color constraints ensuring that the mutation process does not produce infeasible solutions with respect to colors. However, neither the crossover nor the mutation operators have the ability to produce feasible solutions with respect to precedence constraints. In order to overcome this issue, the Precedence Constraint Reparation (PCR) algorithm was developed.

4.2 Precedence Reparation Algorithm

The PCR is applied once after the creation of the initial population. Then it is applied in every generation after crossover (when used) and mutation. The algorithm (Algorithm 3) starts by iterating through allocated tasks for each agent. If a conflict is identified, the algorithm continues searching through tasks until the type of conflict is recognized. Two conflict types can occur. Firstly, when both tasks that have precedence relations are allocated to the same agent, and at the same time the ordering is wrong. Secondly, when tasks are not allocated to the same agent. This type has three sub-cases, which are explained in more detail below.

In case (1) task genes are simply swapped. In case (2) there are three different sub-cases. In sub-case (1) both tasks are allocated to agents able to handle defined constraints, then a uniform random mechanism is used to determine which of the tasks will be re-allocated and which one will retain the previous allocation. In sub-case (2) one of the tasks is allocated to an agent that cannot fulfill the task's constraints. In that case, the invalid task is allocated to the agent of the task with valid constraints. In sub-case (3) both of the tasks are allocated to agents that cannot fulfill their constraints. In this case, a search is performed to find if there is an agent available that can fulfill both of the tasks. If that agent exists, both tasks are re-allocated to that agent. In every case, the location of the re-allocated task within an agent is randomly determined with

Algorithm 3 Precedence Constraint Reparation

```
1:  procedure PCR(population)
2:    Iterate through the chromosome and find a task with a PC violation
3:    case (1) - Tasks are allocated to the same agent
4:      Swap conflicting tasks
5:    case (2) - Tasks are not allocated to the same agent
6:      sub-case (1) - Both agents are suitable for the allocated tasks
7:          Randomly choose which task to re-allocate
8:          Move chosen task with respect to the PC
9:      sub-case (2) - One agent is not suitable for the allocated task
10:         Reallocate invalid task to the agent of a valid task w.r.t. PC
11:     sub-case (3) - Both agents are unsuitable for the allocated tasks
12:         Create a list of suitable agents
13:         Randomly select an agent
14:         Reallocate both tasks to that agent w.r.t. PC
15: return fixed population
```

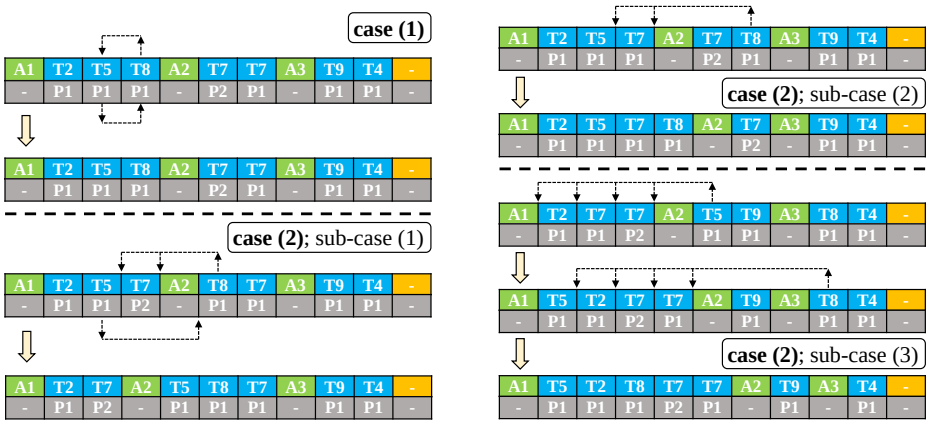


Fig. 4: An example of the mechanism of PCR, showing both cases and all three possible sub-cases of case (2).

respect to the PC. The graphical representation of how PCR works is given in Fig. 4. An example of the aforementioned cases with possible solutions will be described in detail in the next paragraph.

Assume that task T5 has to be executed before task T8, i.e., T5 $\prec$ T8, and that only agents A1 and A2 have the necessary equipment to perform these tasks. In case (1) both tasks are allocated to the appropriate agent (A1), and a simple task swap is performed in order to fix the precedence conflict. The first sub-case of the case (2) shows the problem when each task is allocated to a different agent, however, both A1 and A2 are able to execute the allocated task. Now the option is to either move T8 to A1 after T5 or T7, or as an alternative, move T5 to A2 and place before T8 (the latter option is shown in Fig. 4). In the sub-case (2) T8 is allocated to A3 although A3 does not meet the requirements of T8. The solution includes moving T8 to A1, and place either after T5 or T7 (the latter option is shown in Fig. 4). Sub-case (3) shows a problem when both of the tasks are allocated to agents that do not meet the tasks' requirements. In this case, first the task that has precedence (T5) over the other task (T8) is reallocated to an appropriate agent, in this case A1. In the example in Fig. 4, T5 is placed right after agent gene A1. After this is done T8 has to be reallocated to A1, as well. Possible locations of insertion of T8 are after T5, T2, T7, or another T7 task. In Fig. 4, T8 is shown to be inserted after T2. As already mentioned, when making a decision of where to re-allocate a certain task uniform random function is used.

4.3 Fitness Function

The fitness function evaluates every individual in the population by calculating a quantitative measure of its performance referred to as fitness, or cost. The actual search is done in a solution space consisting of either only feasible, or both feasible and infeasible solutions. Allowing infeasible solutions in the population is a design choice.

The first approach is used in this paper with respect to mutation operator. This operator is defined in such a way that only feasible solutions can be produced. However, crossover operator may produce infeasible solutions since it does not take into account color or precedence of tasks. In this case, where operators are not bound to always produce valid chromosomes, the fitness function has a penalty/award system in order to deal (penalize) with invalid chromosomes [32]. The former approach performs a search over reduced solution space, whereas in the latter approach implementation of variation operators is simpler.

In the fitness function, the candidate solution that is being evaluated is first divided into agent's plans, i.e., sub-plans per each agent. Each plan is evaluated separately and results are combined afterward to form an overall chromosome cost. The objective function that is being optimized is given in Eq. (20).

5 Simulation Results

The experimental platform was i7-8700 @ 4.4GHz with 16GB of DDR4 RAM. The GMP is implemented in the C++ programming language as single-threaded process. Ten different and increasingly difficult multi-agent mission scenarios[2] are randomly generated and used for evaluating the proposed approach. Tasks can have three different colors. Agents are randomly created with up to three different colors. Instances range from 10 to 500 tasks and 1 to 10 agents. The amount of precedence constraints per instance is randomly generated in a range from 5–20% of total task number. The objective function (Eq. 20) is used with w_1 and w_2 being 1 and 0.1, respectively. Every instance is run 30 times. The detailed instance settings are given in Table 1.

[2] The link to benchmark scenarios: https://github.com/mdh-planner/ECTSP

Table 1: Settings for different simulation scenarios.

Instance	**Task's settings**						**Agent's settings**					
	Eq1	**Eq2**	**Eq3**	**#PC**	**#Tasks**	**ATD**	**Eq1**	**Eq2**	**Eq3**	**#Agents**	**SD**	**DD**
1	-	10	-	1	10	594.7	-	1	-	1	1	1
2	-	10	20	5	30	552.5	-	1	2	2	2	1
3	27	23	-	5	50	485.5	2	2	-	3	3	2
4	21	23	31	13	75	491.6	1	3	3	4	4	2
5	36	33	31	6	100	528.9	4	2	2	5	5	3
6	44	52	54	25	150	519.9	3	3	5	6	6	3
7	58	65	77	14	200	475	5	2	1	7	7	4
8	89	102	109	51	300	518.4	2	5	5	8	8	4
9	149	127	124	60	400	515.8	7	3	5	9	9	5
10	172	163	165	30	500	500.4	5	6	7	10	10	5

*(#PC - Number of precedence constraints; #Tasks - The total number of tasks; ATD - Average Task Duration in seconds; #Agents - The total number of available agents; SD - The number of source depots; DD - The number of destination depots)

Due to space limitations, only selected representative results will be shown. This includes the comparison between different mutation probabilities. Among tested values (5–30%), 10% is the lowest mutation probability that does not lead to the degradation of performance. For this reason, in all the following benchmarks the mutation probability will be fixed to 10%. In the same way, it is found that the lowest percentage for the individuals to be kept and transferred to the next generation (elitism) is 5%. Three different crossover operators were tested: (i) single point, (ii) partially mapped, and (iii) edge recombination crossover. While single point crossover performed poorly, as expected on combinatorial problems, edge recombination crossover beat partially mapped crossover by a small margin. This is an expected result, since ERX tries to maintain edges connecting two tasks. This comes particularly handy in the case of precedence constraints. In the following tests, ERX was used, with different probability factors.

To determine the best combination of variation operators, four different combinations were tested (Table 2).

It can be seen that the gap between settings "X, M, & PC" and the others increase as the complexity of the instances increase. For simpler instances, the difference in best and median values is not large, except for the setting with crossover and mutation without precedence reparation algorithm. This setting performed really poorly on all instances but the first one, and that is the reason it will be excluded from further discussion. The difference between the best values for the other three settings for the first 5 instances ranges from 0% to 7%. This gap increases for harder instances, especially in the case of only mutation. The solutions this setting produces for harder instances lags behind the other two settings. From this test it can be concluded that only "M & PC" and "X, M & PC" settings are competitive. The difference between these two settings never exceeded 10%. This required a closer look so a new set of tests was conducted to determine the best setting for provided benchmark.

Table 2: Simulation results for 10k generation and 200 population over different variation operators settings.

Inst.	**Different Variation Operators Settings**															
	M $[\times 10^5]$				**M & PC** $[\times 10^5]$				**X(70%) & M** $[\times 10^5]$				**X(70%), M, & PC** $[\times 10^5]$			
	best	std.	mdn.	t $[s]$	best	std.	mdn.	t $[s]$	best	std.	mdn.	t $[s]$	best	std.	mdn.	t $[s]$
1	0.79	**0**	0.79	2.60	0.79	0	0.79	2.76	0.79	0.03	0.82	28.0	0.79	0	0.79	26.34
2	1.00	0.06	1.09	6.08	1.01	0.06	1.08	6.62	1.68	0.11	2.18	65.69	0.98	0.04	1.01	75.09
3	0.93	0.11	1.09	9.21	0.96	0.06	1.03	9.93	2.38	0.06	2.23	111.4	0.93	0.05	1.04	127.9
4	1.45	0.16	1.70	12.68	1.41	0.07	1.52	13.42	3.62	0.03	3.75	167.9	1.36	0.03	1.40	197.0
5	1.18	0.08	1.31	16.49	1.14	0.08	1.26	17.14	3.40	0.06	3.56	231.5	1.14	0.05	1.20	261.6
6	1.99	0.19	2.26	23.49	1.42	0.07	1.56	23.17	4.60	0.10	4.77	333.6	1.41	0.07	1.52	421.1
7	3.06	0.15	3.34	22.42	2.88	0.14	3.16	24.30	12.1	0.03	12.2	443.7	2.70	0.11	2.84	602.7
8	4.05	0.40	4.85	40.35	2.64	0.10	2.78	41.86	8.97	0.10	9.20	707.7	2.48	0.11	2.78	642.4
9	5.56	0.45	6.52	56.48	3.02	0.08	3.22	58.43	10.3	0.13	10.6	1025	2.90	0.10	3.13	1820
10	5.70	0.35	6.50	72.08	3.46	0.09	3.66	75.11	12.2	0.15	12.6	1396	3.40	0.11	3.67	1352

*(Inst. - test instance; M - Mutation; X - Crossover, PC - Precedence Constraint Reparation; mdn. - median; std. - standard deviation; best - best solution found)

The new test consisted of comparisons between a setting with "M & PC", "X(10%), M, & PC", and "X(70%), M, & PC". This time, the population size has been increased to 500. Obtained results are presented in Table 3.

The difference between results is quite small up to Instance 7, after this, the gap started increasing in the favor of "X(70%), M, & PC", while setting "X(10%), M, & PC" started performing worse than the other two. On hardest instances, the gap between "M & PC" and "X(70%, M, & PC" is less than 10%, however the difference in execution time is 16.3x faster in favor of "M & PC". In order to gain a more solid statistical insight into the difference between these three settings, a series of statistical tests were conducted.

We formulate the **null hypothesis:** *"Using different variation operator settings has no significant effect on the final result"* is true. It is assumed that the samples used in these experiments are random and independent.

Since the sample data can be highly skewed and have extended tails, an average of that data can produce a value that behaves non-intuitively, thus the median was used instead of the mean. The non-parametric Mann-Whitney-Wilcoxon test is used. Since multiple comparisons are performed, the Benjamini-Hochberg (B-H) procedure is applied in order to adjust the false discovery rate. The B-H critical value is calculated as $(i/m) \cdot Q$, where i is the individual p-value rank, m is the number of tests and Q is the false discovery rate percentage set to 1%. The variation operator settings that had the best median value is compared to all other setups in all 3 scenarios. Since there is a statistically significant difference between the results (Table 4), the null hypothesis is rejected.

Statistical tests show that for larger instances there is a benefit of using ERX crossover operator with 70% probability. In comparison with GA setting with 10% ERX probability, there is no significant statistical difference for the first 6 instances. Instances

Table 3: Simulation results for 10k generation and 500 population over different variation operators settings.

Inst.	**Different Variation Operators Settings**											
	M & PC [$\times 10^5$]				**X(10%) & M, & PC** [$\times 10^5$]				**X(70%), M, & PC** [$\times 10^5$]			
	best	std.	mdn.	t [s]	best	std.	mdn.	t [s]	best	std.	mdn.	t [s]
1	0.79	0	0.79	8.47	0.79	0	0.79	36.10	0.79	0	0.79	29.02
2	0.98	0.06	1.08	18.82	0.98	0.03	1.03	81.69	0.98	0.03	1.01	82.09
3	0.93	0.07	1.01	27.66	0.92	0.05	1.00	128.6	0.92	0.05	1.01	138.3
4	1.38	0.07	1.48	36.01	1.36	0.02	1.40	179.0	1.36	0.02	1.40	221.8
5	1.15	0.06	1.28	45.48	1.13	0.05	1.21	120.0	1.14	0.04	1.20	319.0
6	1.38	0.06	1.52	60.77	1.45	0.04	1.54	149.0	1.38	0.06	1.51	556.1
7	2.83	0.14	3.11	65.10	2.71	0.10	2.87	181.8	2.65	0.07	2.79	837.1
8	2.49	0.09	2.68	110.1	2.59	0.09	2.74	349.3	2.46	0.08	2.61	1479
9	2.88	0.07	3.00	151.3	2.97	0.10	3.17	893.0	2.78	0.07	2.91	2341
10	3.25	0.08	3.43	194.9	3.49	0.08	3.68	1122	3.08	0.09	3.34	3172

*(Inst. - test instance; M - Mutation; X - Crossover, PC - Precedence Constraint Reparation; mdn. - median; std. - standard deviation; best - best solution found)

Table 4: Statistical test of the gathered results shown in Table 2.

X(70%), M, & PC vs.	**Inst. 1 p-val.**	**Inst. 2 p-val.**	**Inst. 3 p-val.**	**Inst. 4 p-val.**	**Inst. 5 p-val.**	**Inst. 6 p-val.**	**Inst. 7 p-val.**	**Inst. 8 p-val.**	**Inst. 9 p-val.**	**Inst. 10 p-val.**	**Crit. value**
X(10%), M, & PC	1	0.068	0.085	0.395	0.371	0.006	6×10^{-6}	10^{-5}	10^{-10}	4×10^{-11}	3×10^{-4}
M & PC	1	0.001	0.559	10^{-4}	0.002	0.050	8×10^{-11}	0.008	4×10^{-6}	10^{-4}	7×10^{-4}

7–10 are performed better with 70% probability of crossover. The comparison with "M & PC" is somewhat similar, except that for instance 4 there is statistically significant difference and for instance 8 there is not.

The comparison of "X(10%), M, & PC" and "M, & PC" is shown in Table 5. For instances 1, 3, and 6 there is no statistically significant difference. Instances 2, 4, 5, 7, and 8 show difference in favor of "X(10%), M, & PC", while instances 9 and 10 (marked with "$*$") show that there is statistically significant difference in favor of "M, & PC". This means that performance of these two settings are dependent on the instance itself and no general assumption can be made. On the contrary, setting "X(70%), M, & PC" shows solid improvement in performance over other settings as the size of the problem increases. It can be concluded that "X(70%), M, & PC" performs equally good or outperforms all other GA settings presented in this paper. The drawback is the execution time, as it gets up to 16x slower than "M, & PC" and up to 2.8x slower than "X(10%), M, & PC". If time is critical, it might be better idea to go for "M, & PC" even though the solution is a bit worse. Obvious example can be a re-planning of an ongoing mission. For the initial planning "X(70%), M, & PC" can be used to get the best possible plan, but if there is a need for a re-plan, "M & PC" can be used to get good-enough plan in a shorter period of time. These differences are graphically presented in Fig. 5, where Fig. 5a, 5c, and 5e show the maximum running time for different GA settings including variation in population size and number of generations for the largest instance (Instance 10). On the other hand, Fig. 5b, 5d, and 5f show how the median cost changes with the change of population size and number of generations for every instance for three different GA settings described in Table 3.

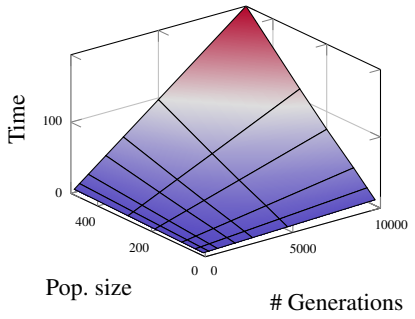
Finally, the convergence of the GA for "X(70%), M, & PC" setting is shown in Fig. 6. The horizontal axis shows the number of function evaluations, i.e., how many times the fitness of an individual was evaluated. The vertical axis shows cumulative probability, ranging from 0 to 1. All instances are presented, except instance 1. Instance 1 appears to be too simple, and it is solved to optimality on every run.

The Empirical Cumulative Distribution Function (ECDF) shows the convergence rate of every instance to the best known solution for that instance. Each sub-figure shows three curves with the distribution of the solutions within predefined distances

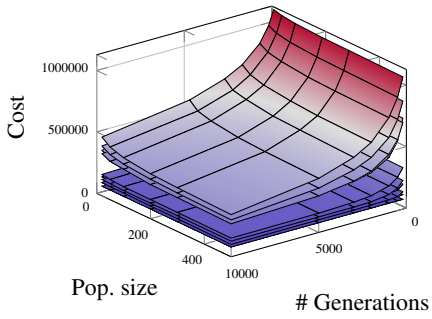
Table 5: Statistical test with p-value = 0.05.

X(10%), M, & PC vs.	**Inst. 1 p-val.**	**Inst. 2 p-val.**	**Inst. 3 p-val.**	**Inst. 4 p-val.**	**Inst. 5 p-val.**	**Inst. 6 p-val.**	**Inst. 7 p-val.**	**Inst. 8 p-val.**	**Inst. 9 p-val.**	**Inst. 10 p-val.**
M & PC	1	0.021	0.530	$2\cdot10^{-4}$	0.012	0.706	$5\cdot10^{-8}$	0.027	$5\cdot10^{-8}*$	$10^{-10}*$

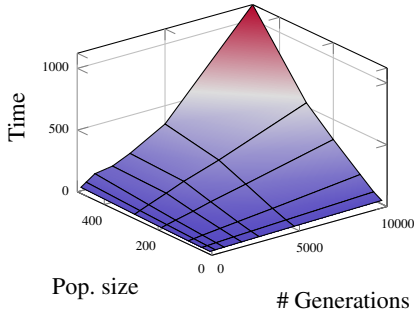
from the best one: The black dotted curve shows the distribution of solutions within 5% of the best solution, the blue dashed curve shows the distribution within 10%, and finally, the red dash-dotted curve shows the distribution of solutions within 15% from the best known solution.



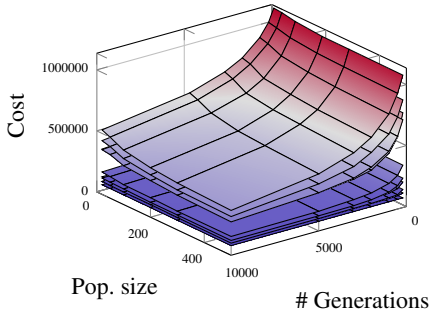
(a) Average times for instance 10 with GA set to 10% mutation and no crossover.



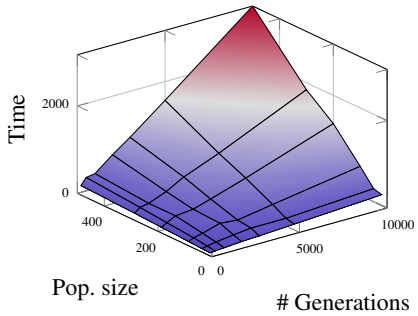
(b) Median cost for all instances with GA set to 10% mutation and no crossover.



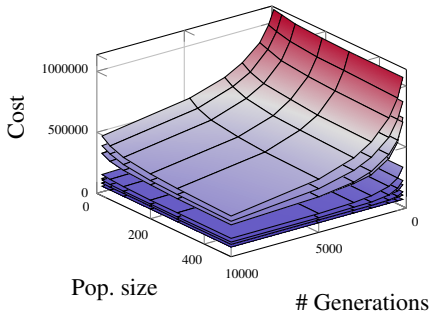
(c) Average times for instance 10 with GA set to 10% mutation and 10% crossover.



(d) Median cost for all instances with GA set to 10% mutation and 10% crossover.



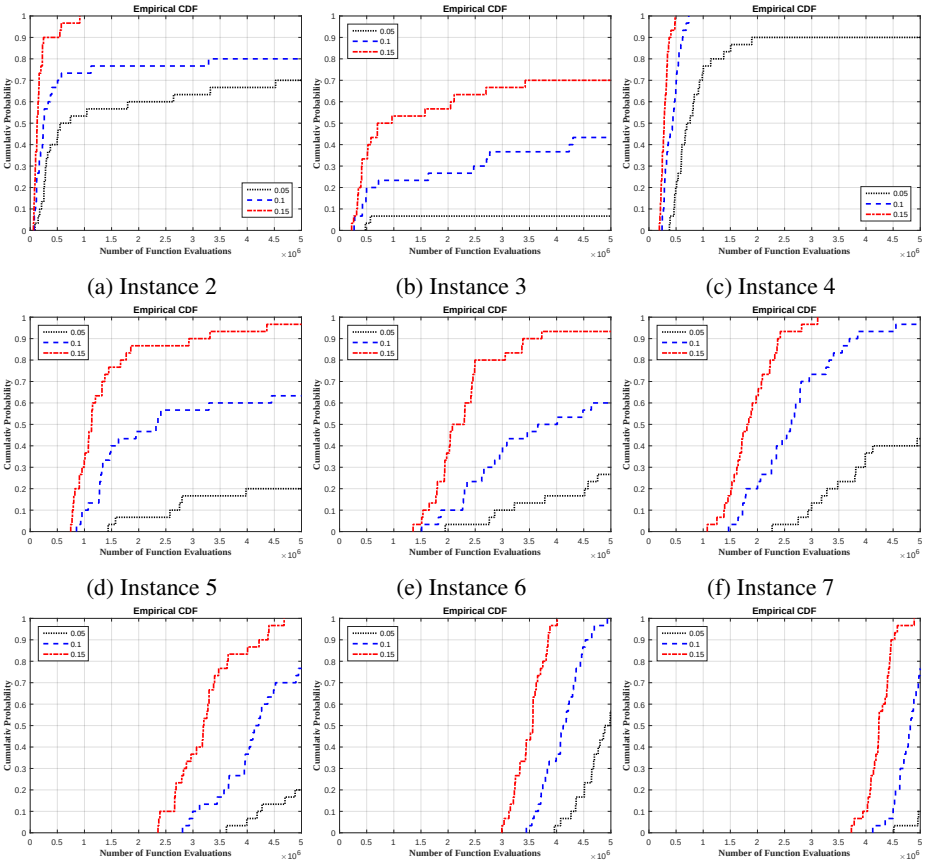
(e) Average times for instance 10 with GA set to 10% mutation and 70% crossover.



(f) Median cost for all instances with GA set to 10% mutation and 70% crossover.

Fig. 5: Analysis of computation time and cost for different GA settings.

By analyzing the ECDF, a few things can be noticed. First, it gives deeper insight into the distribution of the solutions. Secondly, it provides insights on how to tune the algorithm parameters. If a given range from the best solution is reached late in the optimization process (e.g., Instance 9 and 10) it means that the algorithm would probably benefit if the number of generations is increased. On the other hand, if the value of 1, for the cumulative probability, is reached very fast, the number of generations can be decreased. The analysis of the ECDF can also help in understanding the underlying behavior of the algorithm on different instances, and give more insight on what are the "hard" instances for a specific algorithm. For example, Instance 3 is very hard to solve, since only less than 10% of the solutions are within 5% distance from the best one. While Instance 4, which has more tasks, reaches distance of 5% in 90% of the times. This kind of analysis can be also misleading in a sense that maybe the best known



(a) Instance 2 (b) Instance 3 (c) Instance 4

(d) Instance 5 (e) Instance 6 (f) Instance 7

(g) Instance 8 (h) Instance 9 (i) Instance 10

Fig. 6: Empirical CDF for instances 2–10 with GA settings: 10% mutation, 70% crossover, 500 population size, and 10000 generations.

solution for Instance 3 is in fact optimal solution, while the best known solution for instance 4 could be a sub-optimal solution found in local minimum, where most of the population converged. This analysis also shows that the hardness of the instance does not necessarily increase with the increase of number of tasks and agents.

6 Conclusion

This paper proposed a genetic algorithm for the automation of multi-agent mission planning. First, it was shown how multi-agent mission planning can be cast into a MILP formulation. The novel formulation is based on the colored TSP and it is named ECTSP. It is concluded that ECTSP can be used to model different multi-agent missions, from different real-world domains. In this work, a GA was designed to be used for ECTSP's objective function optimization with improvements in the variation operators aimed at handling given constraints. As demonstrated, the objective function presented in this work is more suitable in specific multi-agent mission cases than other solutions found in the literature. In addition, a PCR algorithm is presented that helps improve the convergence of the GA.

By analysis of the presented benchmark results for different GA settings it can be concluded that the best results are achieved when both mutation and crossover operator (ERX) are used combined with PCR algorithm. In addition, if there is a need for a good tradeoff between convergence time and quality of the solution, the GA can be run with only mutation and PCR. Benchmark results show that the speed gain can be up to 16x, with the degradation in the cost of up to 10%.

Acknowledgements

Special thanks to Afshin E. Ameri for developing GUI for the MMT.

References

1. Alighanbari, M., Kuwata, Y., How, J.P.: Coordination and control of multiple UAVs with timing constraints and loitering. In: Proceedings of the 2003 American Control Conference, 2003. vol. 6, pp. 5311–5316 vol.6 (June 2003)
2. Baker, B.M., Ayechew, M.: A genetic algorithm for the vehicle routing problem. Computers & Operations Research **30**(5), 787–800 (2003)
3. Bektas, T.: The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega **34**(3), 209–219 (2006)
4. Bello-Orgaz, G., Ramirez-Atencia, C., Fradera-Gil, J., Camacho, D.: GAMPP: Genetic algorithm for UAV mission planning problems. In: Intelligent Distributed Computing IX. vol. 616, pp. 167–176. Springer International Publishing, Cham (2016)
5. Bhatt, N., Chauhan, N.R.: Genetic algorithm applications on job shop scheduling problem: A review. In: International Conference on Soft Computing Techniques and Implementations (ICSCTI). pp. 7–14 (2015)
6. Brie, A.H., Morignot, P.: Genetic planning using variable length chromosomes. In: International Conference on Automated Planning and Scheduling. pp. 320–329. ICAPS (2005)

7. Contreras-Bolton, C., Parada, V.: Automatic combination of operators in a genetic algorithm to solve the traveling salesman problem. PLOS ONE **10**(9), e0137724 (2015)
8. Cushing, W., Kambhampati, S., Mausam, Weld, D.S.: When is temporal planning really temporal? In: Proceedings of the 20th International Joint Conference on Artifical Intelligence. pp. 1852–1859. IJCAI'07 (2007)
9. Dantzig, G., Fulkerson, R., Johnson, S.: Solution of a large-scale traveling-salesman problem. Journal of the Operations Research Society of America **2**(4), 393–410 (1954)
10. Dorri, A., Kanhere, S.S., Jurdak, R.: Multi-agent systems: A survey. IEEE Access **6**, 28573–28593 (2018)
11. Finke, G., Claus, A., Gunn, E.: A two-commodity network flow approach to the traveling salesman problem. Congressus Numerantium **41**(1), 167–178 (1984)
12. Frasheri, M., Cürüklü, B., Ekström, M., Papadopoulos, A.V.: Adaptive autonomy in a search and rescue scenario. In: Proceedings of the 12th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO). pp. 150–155 (Sep 2018)
13. Gerkey, B.P., Matarić, M.J.: A formal analysis and taxonomy of task allocation in multi-robot systems. The International Journal of Robotics Research **23**(9), 939–954 (2004)
14. Giardini, G., Kalmár-Nagy, T.: Genetic algorithm for combinatorial path planning: The subtour problem. Mathematical Problems in Engineering pp. 1–31 (2011)
15. Gouveia, L., Pesneau, P., Ruthmair, M., Santos, D.: Combining and projecting flow models for the (precedence constrained) asymmetric traveling salesman problem. Networks **71**(4), 451–465 (2018)
16. Holland, J.H.: Genetic algorithms. Scientific American **267**(1), 66–73 (1992)
17. Qing-dao-er ji, R., Wang, Y.: A new hybrid genetic algorithm for job shop scheduling problem. Computers & Operations Research **39**(10), 2291–2299 (2012)
18. Kadri, R.L., Boctor, F.F.: An efficient genetic algorithm to solve the resource-constrained project scheduling problem with transfer times: The single mode case. European Journal of Operational Research **265**(2), 454–462 (2018)
19. Keld, H.: An effective implementation of K-opt moves for the Lin-Kernighan TSP heuristic. Mathematical Programming Computation **1**, 119–163 (2009)
20. Khamis, A., Hussein, A., Elmogy, A.: Multi-robot task allocation: A review of the state-of-the-art. In: Cooperative Robots and Sensor Networks 2015, pp. 31–51. Springer (2015)
21. Korsah, G.A., Stentz, A., Dias, M.B.: A comprehensive taxonomy for multi-robot task allocation. The International Journal of Robotics Research **32**(12), 1495–1512 (2013)
22. Kubo, M., Kasugai, H.: The precedence constrained traveling salesman problem. Journal of the Operations Research Society of Japan **34**(2), 152–172 (1991)
23. Kuschel, T.: Two-commodity network flow formulations for vehicle routing problems with simultaneous pickup & delivery and a many-to-many structure. Informations- und Kommunikationssysteme in Supply Chain Management, Logistik und Transport **5**, 153–169 (2008)
24. Landa-Torres, I., Manjarres, D., Bilbao, S., Del Ser, J.: Underwater robot task planning using multi-objective meta-heuristics. Sensors **17**(4:762), 1–15 (2017)
25. Langevin, A., Desrochers, M., Desrosiers, J., Gélinas, S., Soumis, F.: A two-commodity flow formulation for the traveling salesman and the makespan problems with time windows. Networks **23**(7), 631–640 (1993)
26. Li, J., Zhou, M., Sun, Q., Dai, X., Yu, X.: Colored traveling salesman problem. IEEE Transactions on Cybernetics **45**(11), 2390–2401 (2015)
27. Ma, F., Li, H.: An algorithm in solving the TSP based on the improved genetic algorithm. 2009 First International Conference on Information Science and Engineering (ICISE2009) pp. 106–108 (December 2009)
28. Maoudj, A., Bouzouia, B., Hentout, A., Toumi, R.: Multi-agent approach for task allocation and scheduling in cooperative heterogeneous multi-robot team: Simulation results. In: 13th International Conference on Industrial Informatics (INDIN). pp. 179–184 (July 2015)

29. Masum, A.K.M., Shahjalal, M., Faruque, F., Sarker, I.H.: Solving the vehicle routing problem using genetic algorithm. International Journal of Advanced Computer Science and Applications **2**(7), 126–131 (2011)
30. Meng, X., Li, J., Dai, X., Dou, J.: Variable neighborhood search for a colored traveling salesman problem. IEEE Transactions on Intelligent Transportation Systems **19**(4), 1018–1026 (2018)
31. Meng, X., Li, J., Zhou, M., Dai, X., Dou, J.: Population-based incremental learning algorithm for a serial colored traveling salesman problem. IEEE Transactions on Systems, Man, and Cybernetics: Systems **48**(2), 277–288 (Feb 2018)
32. Miloradović, B., Çürüklü, B., Ekström, M.: A genetic planner for mission planning of cooperative agents in an underwater environment. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1–8 (December 2016)
33. Miloradović, B., Çürüklü, B., Ekström, M.: A genetic mission planner for solving temporal multi-agent problems with concurrent tasks. Lecture Notes in Computer Science **10386 LNCS**, 481–493 (2017)
34. Miloradović., B., Çürüklü., B., Ekström., M., Papadopoulos., A.V.: Extended colored traveling salesperson for modeling multi-agent mission planning problems. In: Proceedings of the 8th International Conference on Operations Research and Enterprise Systems - Volume 1: ICORES,. pp. 237–244. INSTICC, SciTePress (2019)
35. Monma, C.: Sequencing with general precedence constraints. Discrete Applied Mathematics **3**(2), 137–150 (1981)
36. Moon, C., Kim, J., Choi, G., Seo, Y.: An efficient genetic algorithm for the traveling salesman problem with precedence constraints. European Journal of Operational Research **140**(3), 606–617 (2002)
37. Nian, L., Jinhua, Z.: Hybrid genetic algorithm for TSP. In: Seventh International Conference on Computational Intelligence and Security. pp. 71–75 (December 2011)
38. Nunes, E., Manner, M., Mitiche, H., Gini, M.: A taxonomy for task allocation problems with temporal and ordering constraints. Robotics and Autonomous Systems **90**, 55–70 (2017)
39. Puljić, K., Manger, R.: Comparison of eight evolutionary crossover operators for the vehicle routing problem. Mathematical Communications **18**(2), 359–375 (May 2013)
40. Ramirez-Atencia, C., Bello-Orgaz, G., R-Moreno, M.D., Camacho, D.: Solving complex multi-UAV mission planning problems using multi-objective genetic algorithms. Soft Computing **21**(17), 4883–4900 (2017)
41. Ramirez-Atencia, C., Del Ser, J., Camacho, D.: Weighted strategies to guide a multi-objective evolutionary algorithm for multi-UAV mission planning. Swarm and Evolutionary Computation **44**, 480–495 (2019)
42. Ramirez-Atencia, C., R-Moreno, M.D., Camacho, D.: Handling swarm of UAVs based on evolutionary multi-objective optimization. Progress in Artificial Intelligence **6**(3), 263–274 (2017)
43. Ramirez-Atencia, C., Rodríguez-Fernández, V., Gonzalez-Pardo, A., Camacho, D.: New artificial intelligence approaches for future uav ground control stations. In: 2017 IEEE Congress on Evolutionary Computation (CEC). pp. 2775–2782 (2017)
44. Valavanis, K.P., Vachtsevanos, G.J.: Handbook of unmanned aerial vehicles. Springer Publishing Company, Incorporated (2014)
45. Yu, Y., Chen, Y., Li, T.: A new design of genetic algorithm for solving TSP. In: Fourth International Joint Conference on Computational Sciences and Optimization. pp. 309–313 (April 2011)
46. Yuichi, N., Shigenobu, K.: A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem. Informs Journal on Computing **25**, 346–363 (2013)

47. Zhong, W.: Multiple Traveling Salesman Problem with Precedence Constraints Based on Modified Dynamic Tabu Artificial Bee Colony Algorithm. Journal of Information and Computational Science **11**(4), 1225–1232 (2014)
48. Zhou, X., Wang, W., Wang, T., Li, X., Li, Z.: A research framework on mission planning of the UAV swarm. In: 12th System of Systems Engineering Conference. pp. 1–6 (June 2017)