GMP: A Genetic Mission Planner for Heterogeneous Multirobot System Applications

Branko Miloradović, *Student Member, IEEE*, Baran Çürüklü, Mikael Ekström, *Senior Member, IEEE*, and Alessandro Vittorio Papadopoulos, *Senior Member, IEEE*

***Abstract*—The use of multiagent systems (MASs) in real-world applications keeps increasing, and diffuses into new domains, thanks to technological advances, increased acceptance, and demanding productivity requirements. Being able to automate the generation of mission plans for MASs is critical for managing complex missions in realistic settings. In addition, finding the right level of abstraction to represent any generic MAS mission is important for being able to provide general solution to the automated planning problem. In this article, we show how a mission for heterogeneous MASs can be cast as an extension of the traveling salesperson problem (TSP), and we propose a mixed-integer linear programming formulation. In order to solve this problem, a genetic mission planner (GMP), with a local plan refinement algorithm, is proposed. In addition, the comparative evaluation of CPLEX and GMP is presented in terms of timing and optimality of the obtained solutions. The algorithms are benchmarked on a proposed set of different problem instances. The results show that, in the presence of timing constraints, GMP outperforms CPLEX in the majority of test instances.**

***Index Terms*—Extended colored traveling salesperson problem (ECTSP), genetic algorithm (GA), multirobot mission planning, multirobot systems.**

I. Introduction

MULTIROBOT systems (MASs) have raised increasing attention in different domains, ranging from underwater [1] to airborne [2] missions. A mission can be defined as a specific set of tasks that a single or a group of agents[1] is in charge of performing. In addition, the tasks of a mission may include precedence constraints (PCs), or may require heterogeneous capabilities, thereby limiting the selection of the agents to be involved in the mission itself, based on the equipped sensors and actuators. The mission planning problem, therefore, is at a higher level of abstraction than the path planning problem, and does not take into account the presence of static or dynamic obstacles when generating the plan. It rather focuses on the assignment of tasks to the different agents included in the missions, and on the definition of the sequence of tasks to be executed to complete the mission.

In most real-world applications, the mission plan is generated manually by domain experts. As a result, the feasibility of the plan, with respect to the mission constraints, is the primary goal, whereas its optimality—in terms of completion time, or energy, or any other objective—is usually considered as a secondary goal. Moreover, the problem can easily become intractable when the number of tasks, or the number of agents, and the mission constraints increases. This article focuses on the development of techniques for tractable and scalable automated mission planning. These aspects become particularly relevant when mission planning is conducted either in the main planning phase, which typically occurs offline and does not have stringent timing requirements, or during the execution of the mission, when a replan request may be triggered by unforeseeable events, for example, the agent equipment fails, a task is no longer accessible by an agent, and a new plan needs to be generated in a much shorter time scale.

In the literature, different ways of modeling and representing mission planning problems have been presented. The most common approach is to cast a mission planning problem into a mixed-integer linear programming (MILP) [3] formulation of a well-known problem [e.g., traveling salesman problem (TSP), vehicle routing problem (VRP), or team orienteering problem]. If such mapping is not possible, the mission planning problem can be formulated as a resource constraint problem [4]–[6]. Emerging multirobot applications combine the capabilities of different types of robots in the same mission, introducing additional constraints that have not been considered in previous formulations. For example, robots may have different equipment and, therefore, cannot execute all the tasks in a mission, or there may be logical PCs among tasks. This work presents an extended version of the colored traveling salesperson problem (CTSP) to capture these new aspects of multirobot mission planning.

When the model of the mission planning problem is defined, several different choices on how to obtain the mission plan can be made, based on the mission requirements. For example, if the optimization time is limited, the usage of metaheuristic algorithms or a combination of them is shown to be a good choice [7], [8]. On the other hand, if the focus of the mission planning is toward the optimality of the solution,

Manuscript received 10 July 2020; revised 8 December 2020 and 16 February 2021; accepted 30 March 2021. Date of publication 13 May 2021; date of current version 16 September 2022. This work was supported by the AFarCloud European Project under Grant 783221 (Call: H2020-ECSEL-2017-2), DPAC under Grant 20150022, and FIESTA—Federated Choreography of an Integrated Embedded Systems Software Architecture, funded by the Swedish Knowledge Foundation. This article was recommended by Associate Editor H. Su. *(Corresponding author: Branko Miloradović.)*

The authors are with the Division of Intelligent Future Technologies, Mälardalen University, 72123 Västerås, Sweden (e-mail: branko.miloradovic@mdh.se; baran.curuklu@mdh.se; mikael.ekstrom@mdh.se; alessandro.papadopoulos@mdh.se).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TCYB.2021.3070913.

Digital Object Identifier 10.1109/TCYB.2021.3070913

[1]Terms *robot* and *agent* are used interchangeably throughout this work.

exact methods are preferred (e.g., branch-and-cut). This work proposes a genetic mission planning (GMP) approach for heterogeneous multirobot systems. The problem is first formalized as a generalization of CTSP [9], which has been widely used in several disciplines, such as robotics, communication systems, and logistics. The generalization includes: 1) PCs, extending the reformulated Miller–Tucker–Zemlin (RMTZ) [10] formulation; 2) the possibility of having multiple depots as the source and destination for the robot; 3) the presence of a *duration* associated with the execution of the task; and 4) a new objective function for the minimization of the total mission duration in the presence of multiple robots. The presented formulations define in a nonambiguous way the problem addressed in this article, and they can be implemented in optimization tools, for example, CPLEX, for computing the optimal solution for the mission planning problem. Unfortunately, exact methods suffer from the dimensionality problem and, therefore, the optimal solution cannot be always computed in a reasonable amount of time. The proposed GMP allows for faster computation of a suboptimal (yet feasible) solution, which can potentially be used even for replanning purposes. In this work, we compare the performance of the two approaches showing that the GMP formulation can provide significant benefits in terms of computation time, while providing either optimal or close-to-optimal solutions.

The main contributions of this article are: 1) a new formulation of the extended CTSP (ECTSP) problem as a MILP problem, with PCs and subtour elimination expressed as an extension of the RMTZ [10] formulation; 2) the proposal of the genetic mission planner (GMP) algorithm, with a local refinement method, for the solution of ECTSP; and 3) a comparative evaluation of the CPLEX implementation of the proposed MILP formulation with a two-commodity flow network (2CFN) formulation [11], [12], and with the GMP approach in terms of solution quality and computational time.

II. Related Work

Dantzig *et al.* [13] were the first ones to introduce the TSP modeled as an integer linear programming (ILP) problem. This TSP formulation is symmetric, that is, the distance between two cities is the same in both directions, consequently halving the number of possible solutions. Since then, TSP has been extended in many directions in order to model various problems. The original problem definition of TSP was extended to utilize multiple salesman (mTSP) [14]. This can be seen as a basis for modeling multirobot missions. Another important variation of TSP is created by adding PCs among different cities to be visited (TSPPC) [15]. The introduction of PC meant that the graph describing TSP is now directed. This variation of the TSP is referred to as the asymmetric TSP (ATSP). A comparative study of different ATSP problem formulations is given by Öncan *et al.* [16]. An experimental comparison of different models and algorithms for ATSP has been provided by Roberti and Toth [17].

Recently, a CTSP formulation has been given by Meng *et al.* [18] in order to model and solve multibridge machine planning in the industry. A similar model has been used to model collision-free scheduling [19] and path-planning [20] in multibridge machining systems. Xu *et al.* [21] developed the Delaunay-triangulation-based variable neighborhood search in order to solve large-scale CTSPs. This formulation was extended with PCs, and multiple source/destination depots, and it is called ECTSP [22]. It is used for modeling heterogeneous multiagent missions. The two-commodity network flow model [11] was used and adapted to model PCs in CTSP. A different approach to a similar problem is taken by Karaman *et al.* [23] where the process algebra is used to model the problem that is later solved with the genetic algorithm (GA). The problem formulation presented in this article encloses all of the mentioned TSP variations. A distributed approach, which runs concurrently on all the vehicles, of multirobot task allocation (MRTA) is presented by Zhao *et al.* [24]. The algorithm iterates between a task inclusion, a consensus, and a task removal phase.

The analogous group of problems includes the VRP. Wang and Lu [25] presented a capacitated green VRP and solved it by means of a memetic algorithm with competition. This approach proves to be efficient and effective in solving this kind of problem. Zhang *et al.* [26] employed a variation of VRP for vehicle outsourcing and profit balancing. The problem has multiple optimization criteria, and the novel multiobjective local search algorithm has been proposed as an effective solution to this problem.

The previously described problems are notoriously hard to solve, since they are extensions of TSP, which is NP-hard and, therefore, the main research challenge is their scalability. Many different approaches have been devised in order to address the specific class of problems. For example, a branch-and-cut algorithm was used to solve ATSP, with PCs, of size of up to 200 nodes [27]. Even though scalability remains an issue, these formulations have been adopted in real-world applications. For example, the problem of traffic management in air-transport systems was modeled using a time-dependent ATSP with time windows and PCs, and was solved by a modified nearest neighbor heuristic with a local search [28]. Another example is the coordinated-measuring machine inspection process [29] that has been modeled and addressed as a generalized ATSP with PCs, where two approaches are compared, that is: 1) CPLEX and 2) ant-colony optimization (ACO). While the former provides solutions only for small to mid-sized problems, the latter performs well also on larger instances.

Mixed-integer programming (MIP) formulations are commonly used to model different mission planning problems. A mixed-integer nonlinear problem formulation has been proposed and used for modeling mission planning in the context of unmanned aerial vehicles (UAVs) [30]. This formulation was then approximated as a MILP problem, and solved with the Gurobi solver. Problem instances of 2 vehicles and 15 waypoints are routinely solved to optimality, while for larger problem instances, only a suboptimal solution was provided. A MILP problem formulation of the real-world routing problems was provided and solved with different greedy algorithm variations by Yuan *et al.* [31]. A MIP formulation of a multirobot mission planning problem was given by Flushing *et al.* [32],

where a two-layer solution was proposed: 1) the selection of sequences of tasks with GA and 2) the service scheduling with iterative local search.

Multirobot mission planning can be described with an MRTA taxonomy. Several taxonomies were proposed to describe MRTA problems [33]–[35]. More recently, the TAMER model [36] was proposed, which utilizes the entity-relationship model in order to give a more comprehensive taxonomy of mission planning problems. Following these MRTA-related taxonomies, the ECTSP can be categorized as single-task robots, single-robot tasks, and time-extended assignments (ST-SR-TA) [33] with intraschedule dependencies (ID) [34], PC [35], multiple source/destination depots, and heterogeneous capabilities [36]. Although these taxonomies help identify different problem configurations, not all of them have a mathematical problem formulation. The next section formally addresses one of these problem configurations, which we called ECTSP.

The focus in this section was mostly on areas of MRTA and operations research. The approaches found in these areas related to modeling of different multiagent missions that are used as a basis for the ECTSP model proposed in this article. Surveyed solver solutions can be devised in two groups: 1) optimal solvers (CPLEX, Gurobi, etc.) and 2) metaheuristic approaches (GA, ACO, bee colony, etc.). We used the same approach on the proposed model of ECTSP, that is, we solved the problem using CPLEX and a GA-based solver, and analyzed the obtained results.

III. Problem Formulation of the Extended Colored TSP

In this section, a new mathematical formulation of the ECTSP is given. In the following, the problem is formulated following the typical terminology of TSP, where "salespersons" represent robots, "cities" represent tasks, "colors" represent the equipment, and "node weight" is an estimated duration of a task. The main difference, compared to the formulation in [22], is in the way PCs are represented. Such a formulation increases the number of decision variables, but it reduces the number of required constraints, leading to a different outcome in our experiments (see Section VI).

Let $s \in \mathcal{S}$ be a salesperson, in a set $\mathcal{S} \coloneqq \{s_1, s_2, \ldots, s_m\}$ of m salespersons, which need to visit n cities in a set $\mathcal{V} \coloneqq \{v_1, v_2, \ldots, v_n\}$. Also, let c be a color that varies in a set $C \coloneqq \{c_1, c_2, \ldots, c_k\}$ of k colors, σ be a source depot in a set $\Sigma \coloneqq \{\sigma_1, \ldots, \sigma_q\}$ of q source depots, and δ be a destination depot in a set $\Delta \coloneqq \{\delta_1, \ldots, \delta_w\}$ of w destination depots; $m, n, k, q, w \in \mathbb{N}^+$. Each salesperson s starts from a source depot σ and finishes its tour at a destination depot δ. The source and destination depots are not considered to be cities. The superset containing all of the cities $\mathcal{V}$ and depots is defined as $\widetilde{\mathcal{V}} \coloneqq \mathcal{V} \cup \{\Sigma, \Delta\}$. In addition, for simplicity, a superset containing all source depot and city elements is defined as $\mathcal{V}^{\Sigma} \coloneqq \mathcal{V} \cup \Sigma$. In the same way, a superset containing all elements of destination depot and cities is defined as $\mathcal{V}^{\Delta} \coloneqq \mathcal{V} \cup \Delta$. An edge $e \in \mathcal{E}$, connecting vertices $i, j \in \widetilde{\mathcal{V}}$ has the cost of ω_{ijs}, where $\omega_{ijs} \geq 0$ represents the cost of traversing edge $e(i, j)$ by salesperson s.

The decision variable $x_{ijs} \in \{0, 1\}$, which defines if a salesperson $s \in \mathcal{S}$ travels from city i to city j, is defined as

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

Every vertex $i \in \mathcal{V}$ has a duration $\xi(i, s)$ that depends on the salesperson $s \in \mathcal{S}$ visiting it, where $\xi : \mathcal{V} \mapsto \mathbb{R}_0^+$. In the case of a source depot, that is, $i \in \Sigma$, the duration $\xi(i, s) = 0 \ \forall s \in \mathcal{S}$. The duration represents the amount of time salesperson s is required to stay in city i to complete the task. Also, every city $i \in \mathcal{V}$ is associated with a color $f_c(i)$, with $f_c : \mathcal{V} \mapsto C$. Each salesperson $s \in \mathcal{S}$ has a set of colors $C_s \subseteq C$ assigned to it (source and destination depots are colorless). A color matrix of a salesperson s, $\mathcal{A}_s \in \{0, 1\}^{n \times n}$, defines which cities allow visits from a salesperson s, and is defined as $\mathcal{A}_s \coloneqq [a_{ijs}]$, with

$$a_{ijs} = \begin{cases} 1, & f_c(i) \in C_s \wedge f_c(j) \in C_s \wedge \pi_{ij} = 1 \\ 0, & \text{otherwise} \end{cases}$$

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

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

This problem can be formulated over a directed graph $\mathcal{G} = (\widetilde{\mathcal{V}}, \mathcal{E})$, with $\widetilde{\mathcal{V}}$ being a set of vertices, and $\mathcal{E} : \widetilde{\mathcal{V}} \times \widetilde{\mathcal{V}} \mapsto \mathbb{R}_0^+$ being a set of edges.

The (mission planning) goal is to find Hamiltonian tours for a set of salespersons $\mathcal{S}$, so that all the vertices are visited exactly once by a salesperson that possesses the same color of the vertex, while optimizing the objective function (1).

We define the objective function

$$J = \left(w_1 Q + w_2 \sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{V}^{\Sigma}} \sum_{j \in \mathcal{V}^{\Delta}} \left(\omega_{ijs} + \xi(i, s)\right) x_{ijs} \right) \tag{1}$$

that is a weighted combination of the longest tour over all the salespersons (Q) and the sum of all tours of all the salespersons. Weights w_1 and w_2 in the objective function are user defined. The usefulness of this specific objective function over alternatives, for example, the minMax function, in multirobot mission planning problems, with durative actions, was shown by Miloradović *et al.* [22].

The optimization problem includes the following constraints involving the decision variable x_{ijs}:

$$x_{ijs} \leq \overline{a}_{ijs}, \qquad \forall i \in \mathcal{V}^{\Sigma} \quad \forall j \in \mathcal{V}^{\Delta} \quad \forall s \in \mathcal{S} \tag{2}$$

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

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

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

$$\sum_{i\in\Sigma}\sum_{j\in\mathcal{V}^{\Delta}} x_{ijs} = 1 \qquad \forall s \in \mathcal{S} \tag{6}$$

$$\sum_{s\in\mathcal{S}}\sum_{j\in\mathcal{V}^{\Delta}} x_{ijs} = \mathcal{B}_i \qquad \forall i \in \Sigma \tag{7}$$

$$\sum_{i\in\mathcal{V}^{\Sigma}} x_{ijs} = \sum_{k\in\mathcal{V}^{\Delta}} x_{jks} \forall j \in \mathcal{V} \quad \forall s \in \mathcal{S} \tag{8}$$

$$x_{iis} = 0 \qquad \forall i \in \widetilde{\mathcal{V}} \quad \forall s \in \mathcal{S} \tag{9}$$

$$\sum_{i\in\mathcal{V}^{\Sigma}}\sum_{j\in\mathcal{V}^{\Delta}} \big(\omega_{ijs} + \xi(i,s)\big)x_{ijs} \le Q \qquad \forall s \in S. \tag{10}$$

In particular, visiting cities that are not in the extended color matrix ($\mathcal{A}_s$) of salesperson s is disallowed (2). Only one salesperson $s \in \mathcal{S}$ can enter (3), and leave (4) each city, and it can do it exactly once. The final destination of a salesperson s must always be one of the destination depots (5), while the starting location must always be at one of the source depots (6). Note that some salespersons can go directly from a source depot to a destination depot, that is, $x_{ijs} = 1, i \in \Sigma, j \in \Delta$. This means that the salesperson is not used in the final plan. The number of salespersons $\mathcal{B}_\sigma$ in each source depot $\sigma \in \Sigma$ is assumed to be provided *a priori*, and it is such that $\sum_{i\in\Sigma} \mathcal{B}_i = |\mathcal{S}|$. The definition of the initial deployment of the salespersons over the source depots is given by (7). It is necessary to ensure that the same salesperson enters and exits a certain city. This constraint is given by (8). Finally, a salesperson s is forbidden to travel from a city i to the same city i (9). The part of the objective function (1) that minimizes the longest tour of a salesperson over all the salespersons is expressed by introducing inequality (10) in conjunction with the new variable Q.

On the other hand, the PCs are defined on the basis of the ATSP with PCs formulation by Sarin *et al.* [37] and the RMTZ formulation for the subtour elimination in ATSP by Gouveia and Pires [10]. These formulations are further extended and adapted to be used with the CTSP. A binary decision variable $y_{ij} \in \{0, 1\}$ is introduced to model PCs between the tasks

$$y_{ij} = \begin{cases} 1, & \text{if city } i \prec j \text{ , } \pi_{ij} = 1 \\ 0, & \text{otherwise} \end{cases}$$

and to introduce subtour elimination constraints (SECs) in the solution of the ECTSP

$$y_{ij} \ge x_{ijs} \qquad \forall i, j \in \mathcal{V} \quad \forall s \in \mathcal{S} \tag{11}$$

$$y_{ij} + y_{ji} = 1 \qquad \forall i, j \in \mathcal{V} \tag{12}$$

$$y_{ik} + y_{kj} + y_{ji} \le 2 \qquad \forall i, j, k \in \mathcal{V}. \tag{13}$$

In particular, (11) states that even though x_{ijs} is 0, there might be a path connecting cities i and j. If y_{ij} is 0, it means no salesperson can access city j from city i (with any possible path). Equation (12) is introduced in order to remove possible precedence loops, that is, if city i precedes city j, then city j cannot precede city i. The transitive property of the graph $\mathcal{G}$ is ensured by (13), that is, it ensures that there are no precedence cycles. We also introduce an additional binary decision variable $z_{ijs} \in \{0, 1\}$

$$z_{ijs} = \begin{cases} 1, & \text{if } i \prec j, \text{ not necessarily immediately} \\ & \text{in the tour of salesmen } s \\ 0, & \text{otherwise} \end{cases}$$

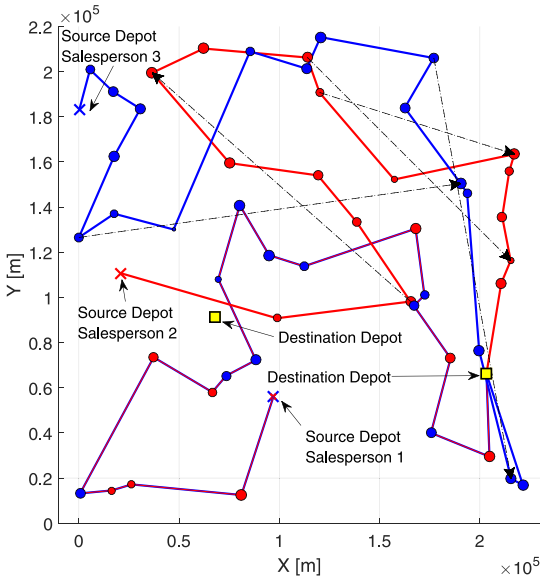

Fig. 1. Example of a solution for Instance 3.

to enforce the PCs on the tour of the specific salespersons $s \in \mathcal{S}$

$$2 \cdot z_{ijs} = \sum_{k\in V} x_{iks} + \sum_{k\in V} x_{kjs}, \qquad i \prec j \quad \forall i, j \in \mathcal{V} \quad \forall s \in \mathcal{S} \tag{14}$$

$$y_{ij} = \sum_{s\in S} z_{ijs} \qquad i \prec j \quad \forall i, j \in \mathcal{V}. \tag{15}$$

In particular, if city i precedes city j, (14) states that the number of edges exiting from city i and entering city j must be equal to $2 \cdot z_{ijs}$ ensuring that cities i and j are connected and visited by the same salesperson s. In other words, cities with mutual precedence relations must be visited by the same salesperson. As a consequence, the final plan is dead-lock free. In order to make sure only one salesperson makes a tour from city i to city j, when $i \prec j$, (15) is introduced.

Finally, we can formulate the optimization problem as

$$\begin{aligned} &\underset{x,y,z,Q}{\text{minimize}} \quad J \\ &\text{subject to} \quad \text{Constraints (2)–(15)} \end{aligned}$$

where $x = \{x_{ijs}\}$, $y = \{y_{ij}\}$, $z = \{z_{ijs}\}$, and $Q \in \mathbb{R}^{\ge 0}$.

Since TSP, which is an NP-hard problem [38], is a special case of ECTSP, we can conclude that ECTSP is at least as hard as TSP, thus at least NP-hard. Scalability is, therefore, the main issue when dealing with this kind of problems. This article analyzes the performance evaluation of different solvers, in terms of convergence rate and optimality.

Example of a Solution: In order to visualize and provide more insight into the problem itself, an example of a solved instance is shown in Fig. 1. The instance that is selected for visualization is Instance 3 from the proposed problem set.[2] This instance has a configuration that is complicated enough to show the full potential of ECTSP, and at the same time,

[2]ECTSP library with ten instances is available for download at https://github.com/mdh-planner/ECTSP.

it is simple enough to be visualized. In the figure, the three source depots, marked with *X*, with one salesperson in each of them are shown. Each salesperson, and its tour, has a specific color code: Salesperson 1 has both blue and red, Salesperson 2 is red, and Salesperson 3 is blue. Note that the tour of Salesperson 1 is drawn in both blue and red, meaning that Salesperson 1 can visit both blue and red cities. The two destination depots are marked with yellow squares. In this solution, all salespersons end their tours at one destination depot. Cities are shown as circles with different coloring. Circles are also different in size, denoting the weight of the city (e.g., duration of the task). Finally, dashed black lines with arrows at one end show the PCs between cities.

IV. Mapping Between ECTSP and Multirobot Mission Planning

The main objective of the ECTSP model presented herein is to model multiagent mission planning problems. The mapping from the problem domain to search domain is quite straightforward in this application. Cities and salespersons map into tasks and agents, respectively, while colors correspond to either equipment required by a task for its fulfillment, or equipment that a salesperson has. Another difference between ECTSP and TSP is due to the fact that in ECTSP, tasks are durative (ξ), and edge weights (ω) depend on the agents they are assigned to (e.g., agents may have different speeds). In addition, some tasks' duration may depend on the specific agent (e.g., survey of an area depends on the max speed of the agent) performing that specific task. Tasks may have PCs, that is, some tasks may need to be performed before some other tasks can be executed.

The applicability of the proposed ECTSP model has been verified in several scenarios belonging to two different application domains. The first application domain assumed multiagent underwater missions. These scenarios included the use of AUVs, ROVs, and surface vessels. The second application domain focuses on the multiagent missions (using drones and ground vehicles) for precision farming scenarios in crops, vineyards, and outdoor livestock activities. The precision farming scenarios aim primarily at data collection and monitoring.

Although the scenarios seem vastly different, also within each application domain, the same ECTSP model has been used in both projects, as the model itself is domain independent. The terms *task* and *agent* are generic enough to be applied in different domains. Nevertheless, task types, requirements, and the morphology and capabilities of agents have been different in both scenarios. This neither affects the model nor the solvers, since the proposed model is used for high-level planning; the differences in the scenarios mainly affect some lower levels of planning abstraction that are out of the scope of this work.

In order to use the proposed model, the mission has to be first defined and then translated into the ECTSP model. To ease the process of defining and monitoring the mission progress and processing of collected data, a mission management tool (MMT) [39] is used. This tool allows a human operator an intuitive way of creating complex missions involving different types of agents with different abilities, for example, due to sensory modalities, configurations, and task requirements.

When the operator had completed preparing a mission through the MMT's graphical user interface (GUI), the mission-related data are translated into the formal model (see Section III), which is forwarded to the solvers (see Section V). The operator manually chooses the target solver for the mission to be sent to. Consequently, the solver is invoked and the optimization process begins. After its completion, the result of the planning process is presented to the operator on the map, in addition to a Gantt chart (Fig. 2). When the final plan has been inspected and verified by the operator, it is forwarded to the dedicated robotic agents for the execution.

The mission shown in Fig. 2 consists of two robots, and five *Inspect* (point) tasks and five *Survey* (area) tasks. From the ECTSP model perspective, it makes no difference if the task is represented as a point or an area, since an area can be approximated to a point with the duration that it would take to survey an area. The Gantt chart shows the duration of each task. The tasks require two different capabilities to take pictures or to use a thermal camera for surveying. These capabilities correlate to colors in the ECTSP model. Finally, each robot finishes its tour at the predefined destination depot.

V. Solvers

There are two approaches for solving the MILP formulation. The first approach makes use of an exact method to obtain an optimal solution or suboptimal with a guaranteed distance from the optimal solution, that is, the gap. The gap usually represents the difference between the best lower bound found with a continuous relaxation of the MILP problem and the current solution. The second approach is based on metaheuristics. While exact methods can guarantee finding the optimal solution (if such exists), metaheuristics solve problems differently, which sometimes leads to suboptimal solutions without guaranteeing optimality. A common approach used in exact methods is to map the problem into a tree, or a graph, and search through the nodes, prune unfeasible branches, and backtrack from dead ends [40], [41]. However, as the search space increases, exact methods may fail to find even a suboptimal plan, within a reasonable time, and suffer from scalability. On the other hand, population-based metaheuristics methods work by seeding candidate solutions across the search space and iteratively improving them with variation operators.

In this work, the objective is to compare the performance, as well as the behavior, of the two approaches on the presented MILP formulation.

A. Genetic Mission Planner

One of the most popular approaches in solving TSP and its variants is by employing metaheuristics, for example, GAs. GA and its variations have been used to solve similar optimization problems, such as VRP [42], job shop scheduling [43], and resource-constrained problems [44].

Fig. 2. Screenshot of the GUI with a generated mission plan and a plan outline in a form of a Gantt chart.

The structure of the solver in this article extends the approach presented in [22], by introducing greedy search (GS) as a local search method. The immediate consequence of the addition of a local search method reduced the need of mutation to occur within an agent's plan. More specifically, swap mutation has been changed to only swap tasks between agents, and not tasks within the same agent as it was the case in the previous version of the solver. In addition, the new solver [GMP(GA+GS)][3] was implemented as a multithreaded application, thus reducing the overall running time. In order to have a valid comparison, the old solver [GMP(GA)] was redone work as a multithreaded application, too. The GMP works as described in the following.

1) Initial Population: The initial population is created at random with respect to given constraints. In the beginning, the minimum number of salespersons with an appropriate color is determined. This is done by creating a unique list of all colors cities have, and finding the minimum number of agents that together possess all the colors from that list. In the next stage, a number of salespersons in the chromosome are randomly picked in the range of the necessary minimum and the absolute maximum number of available salespersons. For example, if all the salespersons possess the same color, then the number of salespersons in a chromosome would be in the range of 1 to m. After this step, for each salesperson, a list of accessible cities is created from the values of $\overline{a}_{ijs}$. The number of tasks per salesperson is randomly determined based on the previously created list. Two or more salespersons may be able to visit the same city. If this is the case, a city is randomly assigned to one of the valid salespersons. The city assignment procedure is repeated until there are no more cities left to assign, and the whole process is repeated for every individual in the population. After the initial population is created, it undergoes the process of PC checking and reparation (see Section V-A4). In this way, it is ensured that all of the candidate solutions in the initial population represent feasible solutions to the problem.

2) Local Greedy Search: The GS is implemented, as a local refinement method, in order to try and improve the candidate solutions produced by GA. While the GA performs allocation of cities to salespersons, the GS only reorders cities within the salesperson's plan based on the nearest neighbor heuristic. In other words, GS performs exploitation of the candidate solution, by reordering the list of cities, governed by nearest-neighbor heuristics. There are no guarantees that GS will be able to improve the candidate solution. Moreover, GS might even produce infeasible solutions, since during the refinement process, PCs are not taken into account. For this reason, refined candidate solutions are evaluated first and if the newly produced candidate solution is better than the original one, it is inserted into the population; otherwise, it is discarded.

3) GMP Workflow: The workflow of GMP is shown in Fig. 3. After the creation of the initial population (Section V-A1), individuals, which are candidate solutions, are evaluated. The cost of each individual is calculated as described in (1), and individuals are then sorted in ascending order based on the costs. After the evaluation and sorting of the initial population, the inner loop of the algorithm starts. The thick arrows in Fig. 3 show the flow within the loop, and blue shaded boxes show the operations done in the inner loop. The algorithm leaves the inner loop when the stopping condition is reached. The number of the generations is fixed and is defined during the initiation phase of the GMP algorithm. Consequently, if the stop condition is not met, the inner loop will execute until the end generation is reached. In each

[3]Referred as GMP in the following for simplicity.

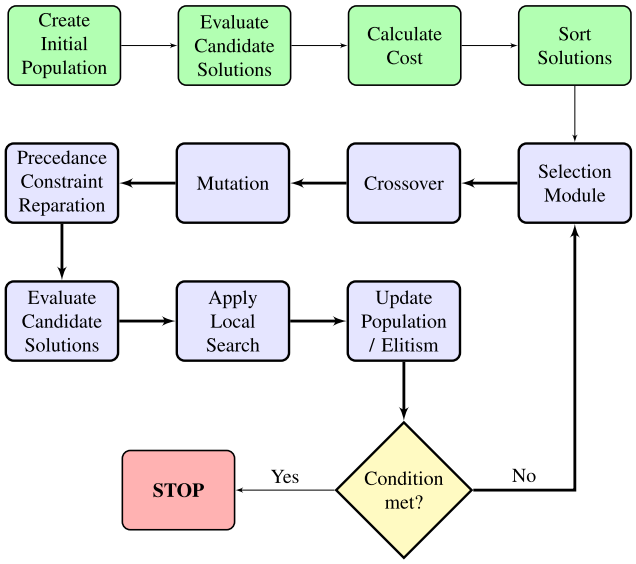

Fig. 3. Flowchart of the GMP algorithm.

generation, the rank selection algorithm is used to select pairs of individuals for mating through the two variation operators: 1) crossover and 2) mutation. Edge recombination crossover (ERX) is used to generate two new offspring from their two parents. The rank selection is used to select parents. The offspring are later subject to mutation, which is the second variation operator. Mutation operators have been specifically tailored for ECTSP [22]. The newly modified population is evaluated again and a local search (GS) is applied (as described in Section V-A2). Finally, the elitism is applied by replacing 5% of the worst performing solutions with the best 5% from the previous generation. When all updates on the population are done, the stopping condition is re-evaluated.

4) Precedence Reparation Process: This operation is applied in every generation, after the variation operators. Since neither mutation nor crossover respect given PCs, it is necessary to perform this step. Two different PC violation cases can be identified here. The first case where the ordering of tasks is reverse, but tasks are still assigned to the same agent. In the second case, tasks are not assigned to the same agent. The second case is in violation of PCs, no matter the actual start time of the tasks involved, since the presented problem formulation supports only intraschedule ordering constraints. The algorithm starts by iterating through each of the candidate solutions. When a task with PC is identified, the corresponding constraints are checked. In case of PC violation, it is necessary to categorize the conflict. If it belongs to the first case, the violation is fixed by randomly determining which task should be removed and inserted into a new valid position in the plan. The new position of insertion is also determined randomly. In the second case, tasks are first reallocated to the correct agent and then the procedure from the first case is performed.

B. CPLEX

CPLEX is a state-of-the-art optimization tool capable of solving various MIP problems. As a tool that has been in development for decades, it incorporates a vast selection of algorithms with tunable parameters.

1) Parameter Settings: In order to perform a fair comparison in this work, nonstandard parameters for CPLEX to speed-up the convergence time have been selected. For the tests presented in this work, the `probing` parameter was set to zero, which means that CPLEX would automatically decide an appropriate level of probing. Presolve is performed at the beginning of the optimization (during preprocessing) and applied a second time to the root relaxation. Heuristic options in CPLEX are used unchanged from the default settings, that is, it was up to CPLEX to decide when and how to use heuristics. The parallel mode was set to opportunistic, using up to 36 threads. The search method was set to default, that is, it was up to CPLEX to determine whether to use traditional branch-and-cut or dynamic search. The rest of the settings is also used with their default values unless otherwise specified.

2) Model Implementation: For convenience, we will call the original ECTSP formulation 2CFN. The ECTSP formulation presented in this article will be referred to as *RMTZ*. With respect to the implementation aspects, there is a difference between the two MILP formulations. RMTZ is implemented as a two-step lazy constraint callback. Note, however, that (13) is not directly implemented in the model, since the number of constraints depends on the number of cities [as $O(n^3)$], direct implementation of this constraint would lead to a significant increase in the overall number of constraints. Thus, this constraint is implemented as a lazy constraint in a callback function. However, (13) in conjunction with (11) and (12) forms the SECs. Hence, when (13) is removed from the model, there is nothing to prevent CPLEX from obtaining a solution with subtours. This is why the Dantzig–Fulkerson–Johnson (DFJ) [13] subtour elimination formulation has been extended and adapted to be applied to multiple salespersons as

$$\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{M}} x_{ijs} \leq |\mathcal{M}| - 1 \quad \forall \mathcal{M} \subseteq \mathcal{V}, \mathcal{M} \neq \emptyset \quad \forall s \in \mathcal{S}. \quad (16)$$

Thus, to eliminate the subtours, it is required that for each nonempty subset $\mathcal{M} \subseteq \mathcal{V}$, the number of edges between the nodes of $\mathcal{M}$ must be at most $|\mathcal{M}| - 1$. Equation (16) is implemented as a lazy constraint. More specifically, it is implemented in the first step of the lazy callback in order to remove subtours. For this reason, the lazy constraint has two steps, which are further explained in Section V-B3. As a consequence of a lazy constraint callback, the dynamic search had to be disabled. On the other hand, the 2CFN model was implemented without any lazy callback function.

3) CPLEX Workflow: The workflow of the CPLEX algorithm is shown in Fig. 4. The green boxes represent operations done before the search process starts. These operations include parsing of mission data and the creation of the CPLEX model according to the MILP formulation proposed in this article. When the search process finds a candidate solution, that is, a solution that satisfies the given constraints, a lazy constraint callback function is invoked. This function consists of two steps. The first step is to check whether the SECs (16) have

TABLE I
BENCHMARK RESULTS OF GMP(GA) AND GMP ON TEN INSTANCES

Inst.	GMP (GA)					GMP				
	best	std.	median	t/run [s]	tot. time [s]	best	std.	median	t/run [s]	tot. time [s]
1	**0.79**	**0**	**0.79**	**19.2**	**576.7**	**0.79**	**0**	**0.79**	20.3	607.9
2	**0.98**	**0.03**	1.02	**41.8**	**1253**	**0.98**	**0.03**	**0.99**	48.1	1442.9
3	**0.92**	0.05	1.00	**58.5**	**1753.8**	**0.92**	**0.03**	**0.99**	71.0	2130.1
4	1.36	**0.02**	1.40	**80.0**	**2401.2**	**1.35**	**0.02**	**1.39**	102.1	3063.2
5	1.11	0.05	1.20	**110.8**	**3322.9**	**1.08**	**0.04**	**1.14**	143.2	4295.2
6	**1.35**	0.06	1.46	**173.5**	**5205.6**	**1.35**	**0.05**	**1.44**	232.7	6979.2
7	2.64	0.06	2.74	**226.4**	**6791.7**	**2.62**	**0.05**	**2.70**	345.4	10362
8	2.40	**0.09**	2.54	**381.0**	**11431**	**2.20**	**0.09**	**2.38**	620.8	18625
9	2.69	**0.05**	2.77	**599.9**	**17996**	**2.25**	0.09	**2.47**	990.0	29701
10	2.95	**0.08**	3.08	**752.8**	**22584**	**2.19**	0.12	**2.37**	1444.1	43322

*(std. - standard deviation; best - best solution found; t/run - average time for one run; best, std., and median values multiplied by 10^5)

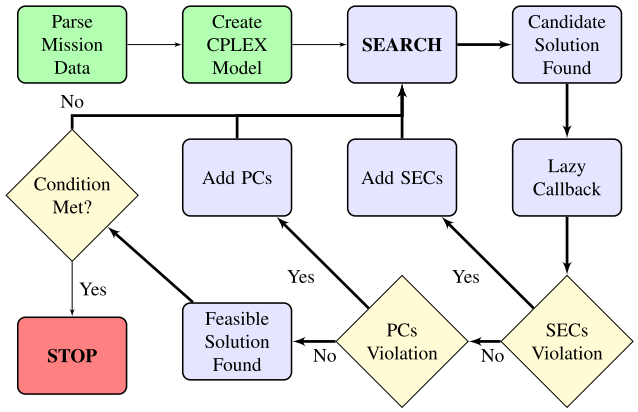

Fig. 4. Flowchart of the CPLEX algorithm.

been violated. In case they have been violated, a set of constraints is added to the model that should eliminate detected subtours. The second step is only reached if there are no subtours in the candidate solution, and this step is used to check for PCs violation. If violations of PCs do exist, constraints following from (13) are added to the model. Otherwise, the candidate solution is considered to be feasible. Finally, it is checked whether the stopping criterion is reached. In this article, the algorithm is stopped if one of two events occur: 1) the optimal solution is reached or 2) the given time limit is exceeded. In the absence of these events, the search process continues.

VI. EXPERIMENTAL RESULTS

This section presents the evaluation of the proposed formulation. After describing the used experimental setup, we present a benchmark of different problem instances used for the evaluation of the proposed method. The evaluation is organized in three parts. The first part compares the GMP(GA) and the GMP solvers developed (Section VI-C), showing the advantages of the introduction of the local search. Second, the implementation in CPLEX of the proposed MILP formulations (2CFN and RMTZ) are compared (Section VI-D) to analyze which one has the best performance in terms of quality of the computed solution, when a timeout to the computation is set. Finally, the GMP formulation is compared with the two MILP formulations, showing its advantages over the considered benchmark (Section VI-E).

A. Experimental Setup

The experimental platform is an i9-9980XE @3.8 GHz (18 cores) CPU with 128 GB of DDR4 RAM. The focus is on the creation of the initial mission plan, and not on the replanning. The assumption is that the time is limited, although, not as critical as in the case of the replanning, which is done during the mission. For this reason, an upper limit for the algorithm to find the initial plan of 24 h has been set. User-defined weights in the objective function (1) are set to $w_1 = 1$ and $w_2 = 0.1$ to make both parts of the objective function comparable. The results consist of three different comparisons.

B. Benchmark

The benchmark for the experimental evaluation of the proposed approaches consists of ten problem instances with a gradually increasing complexity in terms of the number of tasks to be completed, agents involved, and the number of PCs. The cities can have one of three different colors. The salespersons are randomly created having up to three different colors. The problem instances assume from 10 to 500 cities, and from 1 to 10 salespersons. The amount of PCs per instance is randomly generated, and is in the range of 5%–20% of the number of cities in that instance.

C. GMP Versus GMP(GA)

The improvements done in the GMP reflect on its overall performance on the tested benchmark. Both solvers are implemented in C++, as a multithreaded application. The performance comparison between the improved GMP and GMP(GA) is shown below (Table I, better values are in bold). Every instance is run 30 times. The results show that in almost every instance, GMP outperforms GMP(GA) in terms of the best solution found and the median value. Naturally, the time per run and total time are in favor of GMP(GA) since GMP also performs a local search. The local search is done at the end of every generation on every candidate solution. The overall runtime of GMP might be reduced if instead of running a local search on every candidate solution in every generation, a smarter mechanism is used to select solutions for

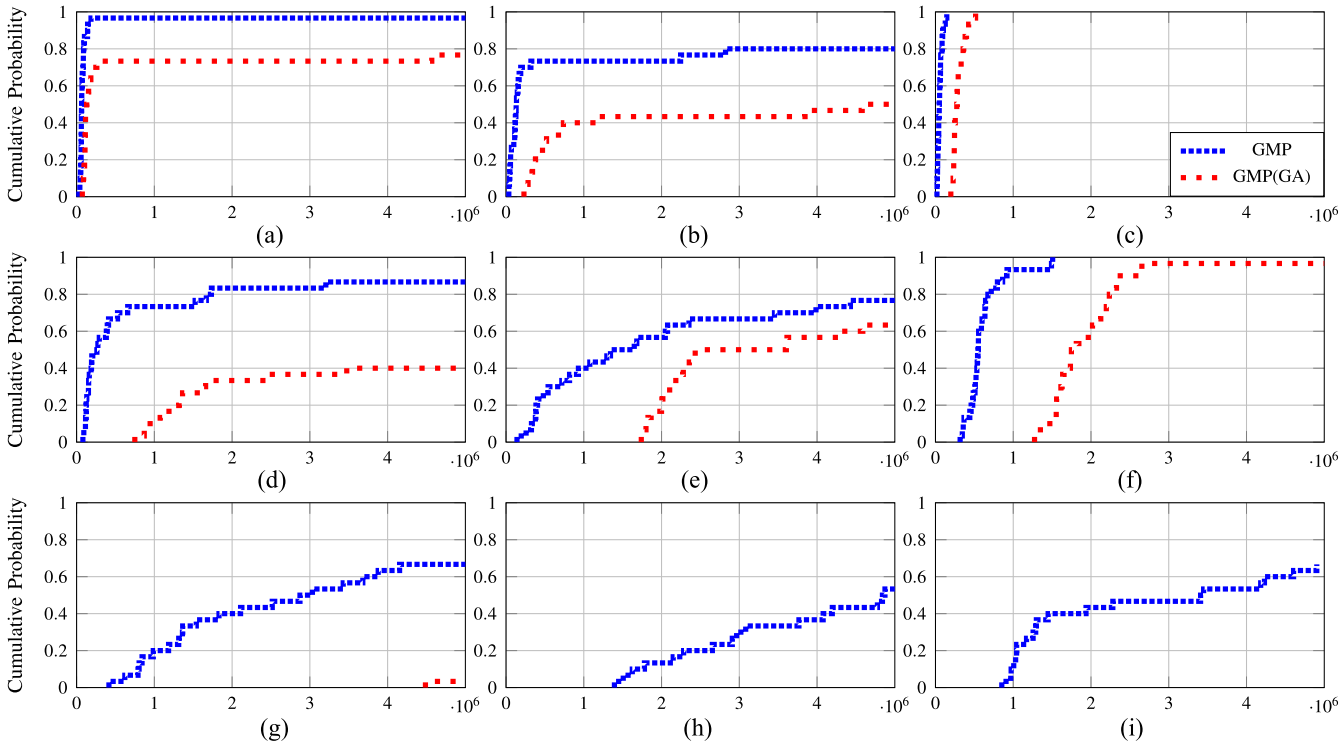

Fig. 5. Empirical CDF for Instances 2–10. The target value is within the 10% of the best known solution for each instance.

the local refinement. The median value of the solution distribution of GMP is as good as GMP(GA) for the first two instances, and the gap widens in favor of GMP toward more complex instances. On the contrary, the standard deviation seems to increase in GMP in Instances 8–10 compared with that of GMP(GA). However, GMP is clearly able to overcome some of the local minima where GMP(GA) gets stuck. Consequently, GMP found equal or better solutions in each instance. In order to provide a better understanding of the underlying performance of the two compared algorithms, an empirical cumulative distribution function (eCDF) is shown in Fig. 5. Instance 1 is omitted from Fig. 5, since it is too trivial to be solved, and the initial population generator was able to find the optimal solution immediately, making the rest of the search process obsolete.

For the target value of eCDF, the best-known solution for that instance is selected. The progress of both algorithms in closing the gap to less than 10% between each of the runs and the target value is shown in Fig. 5. GMP's cumulative probability is represented with a densely dotted blue line. It can be noted that for every instance, it has the cumulative probability higher than that of GMP(GA) (loosely dotted red line, Fig. 5). In addition, GMP converges, within the 10% of the best solution, with high cumulative probability (higher than 60%), except in Instance 9, where it is a little above 50%. GMP(GA), on the other hand, failed to find a solution within 10% from the best one for Instances 9 and 10. This figure also shows how some instances are harder than others, and that it is not strictly related to the number of cities/salespersons. For example, Instance 4 [Fig. 5(c)] is solved easier than Instance 3, judging from the number of function evaluations to reach the value of 100% as cumulative probability.

D. Comparison of the MILP Models

The first comparison is done between the two different ECTSP MILP formulations. Both of these formulations are implemented in CPLEX Concert Technology 12.8 and are tested in four different cases. In all cases, the search process is stopped after 24 h (or before if the optimal solution was found).

In case 1, both ECTSP formulations started without any warm start feature, that is, no initial solution was provided. The search emphasis was put on balancing between feasibility and optimality. In case 2, the warm start option was used. The initial solution provided was generated by the same algorithm that generates an initial population for the GMP (Section V-A1). It is important to emphasize that all of the provided warm start solutions were feasible. In case 3, the warm start was provided; however, this time it was the best solution found by GMP (Table I). In cases 2 and 3, the search emphasis was the same as in case 1, that is, balancing optimality and feasibility. In case 4, however, the search emphasis was changed to moving the best bound. Since in case 3, CPLEX failed to improve any of the initially provided solutions, the idea being to see if the bound can be further moved in order to minimize the gap. The gap is defined as

$$\text{gap} = \frac{(\text{bs} - \text{bb})}{\left(\text{bs} \cdot 10^{-2}\right)} \tag{17}$$

where bs is the best solution found, and bb is the best bound.

The tests are done on the first six instances of the previously described benchmark (Section VI-B) and the results can be seen in Fig. 6. The blue colored bars represent the best bound, whereas the green bars illustrate the gap (17) between best bound and feasible solution found. The yellow bars show the

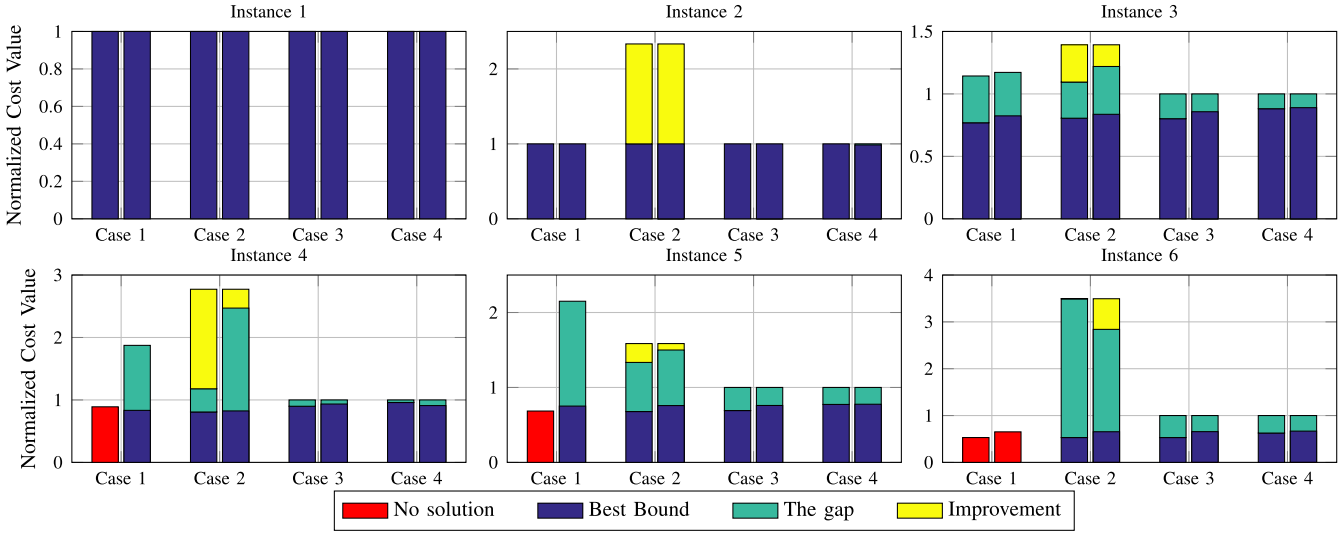

Fig. 6. Visualization of the comparison of 2CFN (left bar) and RMTZ (right bar) formulations, on four cases, implemented in CPLEX and tested on six instances.

improvement done by CPLEX on the provided initial solution. Finally, the red bars represent the best bound for the instances where no feasible solution is found. While the x-axis shows each of the four cases, the y-axis represents normalized cost value (Fig. 6). All the cost values are normalized with respect to the best solution obtained for each instance.

Instance 1 seems to be easily solvable by CPLEX, regardless of the formulation or the test case, and the optimal solution was reached within a second. Instance 2 is also solved to optimality in every case; however, the time taken varies between the MILP formulations. Instance 3 is not solved to optimality in a given time span as it can be noted in Fig. 6. The MILP formulation proposed in this article gives a better bound, while 2CFN found a slightly better solution.

For 2CFN, CPLEX is unable to find a feasible solution for Instance 4. However, it produced a better bound than RMTZ, although RMTZ managed to find a feasible solution. 2CFN produced a better bound in case 4 as well. These are the only two cases throughout these tests for which CPLEX found better bound for 2CFN than RMTZ formulation.

Similar to the previous instance, in Instance 5, 2CFN failed to produce a feasible solution; however, this time it founds worse bound as well. However, when CPLEX was provided with a warm start solution, 2CFN produced a better solution.

The RMTZ formulation outperforms 2CFN in instance 6 in all categories (Fig. 6). In case 1, both models failed to find a feasible solution; however, the RMTZ formulation provided better lower bound. In case 2, the provided initial solution was marginally improved; however, RMTZ provided better lower bound and improved initial solution. In case 3, neither of the formulations was able to improve the given initial solution. Case 4 was similar to case 3, with lower bound marginally improved; however, the difference between lower bounds was smaller, although still in favor of RMTZ formulation.

Since CPLEX failed to find a feasible solution for Instance 6, with either formulation, we decided not to continue the benchmark on the remaining four instances in the ECTSP library. Based on the presented results, it can be concluded that RMTZ formulation, on average, provides better lower bound and was even able to provide a feasible solution for instances where 2CFN failed to do so. On the other hand, 2CFN was more successful in improving bad initial solutions.

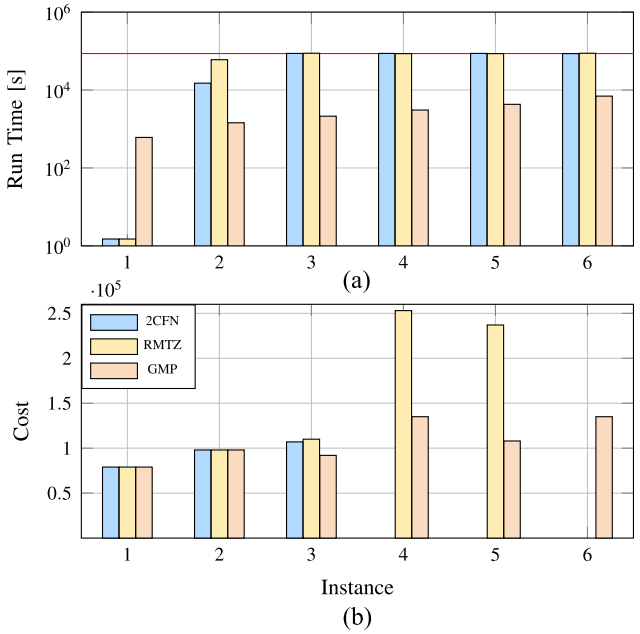

Fig. 7. (a) Comparison between the runtimes of 2CFN, RMTZ, (for case 1) and GMP on six test instances. (b) Cost of the best found solution for each of the approaches (lower is better). If no solution is found, no bar is plotted.

E. CPLEX Versus GMP

Finally, we compare results generated by both GMP and CPLEX. Fig. 7 shows a comparison of the two MILP formulations, and the GMP approach, for the different instances in terms of computation time [Fig. 7(a)] and of best computed cost [Fig. 7(b)]. Notice that in Fig. 7(a), the y axis is

logarithmic, and that the timeout of 24 h set for finding the best solution is indicated with a red solid line. The total running time of GMP is significantly lower than the running time of CPLEX, except in Instance 1 [Fig. 7(a)]. However, GMP settings are excessive for this instance, since the runs were fixed to the same values for all instances. More specifically, to 10 000 generations and a population size of 500, with 70% crossover, 10% mutation, and 5% elitism. This is also notable in the standard deviation, which is zero. In the case of the remaining five instances, not only does GMP produce equal (Instance 2) or better results (Instances 3–5) but also does it in a significantly shorter time (15–40 times faster). In the case of Instance 6, both formulations are unable to find a feasible solution, and therefore are no bar is plotted in Fig. 7(b). In addition, it failed to improve on the provided solution from GMP and only managed to prove that the provided solution is, in the worst case, 33.26% away from the optimal one. It can be concluded that for instances smaller than Instance 2, CPLEX can be a viable solution. For even smaller instances, CPLEX can be used not only for initial planning but online replanning as well. On the other hand, GMP is obviously more suitable for larger problems if there is no need for a guaranteed optimal solution, and the time is somewhat limited. It is also worth noticing that GMP is an anytime algorithm that always produces feasible solutions.

In addition, GMP could be further improved with the use of more sophisticated local search than GS. On the other hand, CPLEX has 76 tunable parameters as stated by Hutter *et al.* [45] in their paper on automated configurations for MIP solvers. They managed to improve proving optimality by the speedup factor of up to 52, and to improve gap minimization with a factor of up to 42, on some MIP problems. However, parameter tuning is a time-consuming process on its own, so it should be used only when there is a clear benefit.

VII. Conclusion

In this article, an alternative MILP formulation of ECTSP, called RMTZ, is presented. It has been compared to the previous ECTSP formulation (2CFN) by implementing both models in CPLEX, and performing a benchmark on six problem instances, which all represent problems in the domain of mission planning with heterogeneous multiagents with a gradually increasing complexity with respect to problem size. The results show that the RMTZ formulation yielded better lower bound in most cases. In addition, in some cases, the RMTZ formulation provided a feasible solution where 2CFN could not. On the other hand, 2CFN was more successful in improving suboptimal initial solutions, which were also distant from the optimal solution. However, CPLEX could not find a solution for the sixth test instance with either of the formulations. In contrast, this was not the issue for the presented GMP based on the GA+GS for local search. First, GMP was compared to the previous version of GMP(GA) and it was shown that the addition of local search in the form of GS helped the algorithm converge faster, and also produced better solutions. Second, GMP performed at least equally good or in some cases even outperformed CPLEX, regardless of the used model, in every test instance in terms of cost, while only in Instance 1 was it slower than the CPLEX. However, it can be attributed to the fixed parameter settings of GMP, which were overly complex for the simple problem represented in Instance 1. To conclude, these results show that for planning of complex missions, which include a large number of tasks and heterogeneous robots, GMP has the upper hand compared to other options presented in this article. However, in application domains that require planning of less complex missions in terms of the number of tasks and robots, or if the planning time is not critical, the CPLEX approach may be considered.

For future work, further parameter tuning in CPLEX in order to improve runtime and/or cost, making CPLEX approach more practical for larger scale problems, even in the replanning phase, is to be considered. Similarly, a heuristics-based problem-specific local search can be incorporated in GMP in order to improve the quality of the solution. This would increase the GMP algorithm's usability and improve its scalability in real-world applications.

References

[1] F. Thompson and D. Guihen, "Review of mission planning for autonomous marine vehicle fleets," *J. Field Robot.*, vol. 36, no. 2, pp. 333–354, 2019.

[2] L. Ru, L. Ya-Fei, and H. Zhong-Xi, "A model of mission planning for cooperative UAVs," in *Proc. Chin. Control Decis. Conf.*, 2015, pp. 67–72.

[3] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[4] N. Tsiogkas and D. M. Lane, "An evolutionary algorithm for online, resource-constrained, multivehicle sensing mission planning," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 1199–1206, Apr. 2018.

[5] H. Nakhost, J. Hoffmann, and M. Müller, "Resource-constrained planning: A Monte Carlo random walk approach," in *Proc. Int. Conf. Autom. Plan. Schedul. (ICAPS)*, 2012, pp. 181–189.

[6] C. Ramirez-Atencia, G. Bello-Orgaz, M. D. R-Moreno, and D. Camacho, "Solving complex multi-UAV mission planning problems using multi-objective genetic algorithms," *Soft Comput.*, vol. 21, pp. 4883–4900, Oct. 2016.

[7] J. Bi, H. Yuan, S. Duanmu, M. C. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 3774–3785, Mar. 2021.

[8] J. Wang, M. Zhou, X. Guo, and L. Qi, "Multiperiod asset allocation considering dynamic loss aversion behavior of investors," *IEEE Trans. Comput. Soc. Syst.*, vol. 6, no. 1, pp. 73–81, Feb. 2019.

[9] J. Li, M. Zhou, Q. Sun, X. Dai, and X. Yu, "Colored traveling salesman problem," *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2390–2401, Nov. 2015.

[10] L. Gouveia and J. M. Pires, "The asymmetric travelling salesman problem and a reformulation of the Miller–Tucker–Zemlin constraints," *Eur. J. Oper. Res.*, vol. 112, no. 1, pp. 134–146, 1999.

[11] G. Finke, A. Clauss, and E. A. Gunn, "A two-commodity network flow approach to the traveling salesman problem," *Congressus Numerantium*, vol. 41, no. 1, pp. 167–178, Jan. 1984.

[12] B. Miloradović, B. Cürüklü, M. Ekström, and A. Papadopoulos, "Extended colored traveling salesperson for modeling multi-agent mission planning problems," in *Proc. Int. Conf. Oper. Res. Enterprise Syst.*, vol. 1, 2019, pp. 237–244.

[13] G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson, "Solution of a large-scale traveling-salesman problem," *J. Oper. Res. Soc. Amer.*, vol. 2, no. 4, pp. 393–410, 1954.

[14] T. Bektas, "The multiple traveling salesman problem: An overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[15] M. Kubo and H. Kasugai, "The precedence constrained traveling salesman problem," *J. Oper. Res. Soc. Jpn.*, vol. 34, pp. 152–172, Jun. 1991.

[16] T. Öncan, İ. K. Altınel, and G. Laporte, "A comparative analysis of several asymmetric traveling salesman problem formulations," *Comput. Oper. Res.*, vol. 36, no. 3, pp. 637–654, 2009.

[17] R. Roberti and P. Toth, "Models and algorithms for the asymmetric traveling salesman problem: An experimental comparison," *EURO J. Transp. Logist.*, vol. 1, nos. 1–2, pp. 113–133, 2012.

[18] X. Meng, J. Li, X. Dai, and J. Dou, "Variable neighborhood search for a colored traveling salesman problem," *IEEE Trans. Int. Transp. Syst.*, vol. 19, no. 4, pp. 1018–1026, Apr. 2018.

[19] J. Li, X. Meng, M. Zhou, and X. Dai, "A two-stage approach to path planning and collision avoidance of multibridge machining systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 7, pp. 1039–1049, Jul. 2017.

[20] J. Li, X. Meng, and X. Dai, "Collision-free scheduling of multi-bridge machining systems: A colored traveling salesman problem-based approach," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 1, pp. 139–147, Jan. 2018.

[21] X. Xu, J. Li, and M. Zhou, "Delaunay-triangulation-based variable neighborhood search to solve large-scale general colored traveling salesman problems," *IEEE Trans. Int. Transp. Syst.*, vol. 22, no. 3, pp. 1583–1593, Mar. 2021.

[22] B. Miloradović, B. Çürüklü, M. Ekström, and A. V. Papadopoulos, "A genetic algorithm approach to multi-agent mission planning problems," in *Operations Research and Enterprise Systems*. Cham, Switzerland: Springer Int., 2020, pp. 109–134.

[23] S. Karaman, T. Shima, and E. Frazzoli, "A process algebra genetic algorithm," *IEEE Trans. Evol. Comput.*, vol. 16, no. 4, pp. 489–503, Aug. 2012.

[24] W. Zhao, Q. Meng, and P. W. H. Chung, "A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario," *IEEE Trans. Cybern.*, vol. 46, no. 4, pp. 902–915, Apr. 2016.

[25] L. Wang and J. Lu, "A memetic algorithm with competition for the capacitated green vehicle routing problem," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 2, pp. 516–526, Mar. 2019.

[26] Z. Zhang, H. Qin, and Y. Li, "Multi-objective optimization for the vehicle routing problem with outsourcing and profit balancing," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 5, pp. 1987–2001, May 2020.

[27] N. Ascheuer, M. Jünger, and G. Reinelt, "A branch & cut algorithm for the asymmetric traveling salesman problem with precedence constraints," *Comput. Optim. Appl.*, vol. 17, no. 1, pp. 61–84, 2000.

[28] T. Saradatta and P. Pongchairerks, "A time-dependent atsp with time window and precedence constraints in air travel," *J. Telecommun. Elect. Comput. Eng.*, vol. 9, nos. 2–3, pp. 149–153, 2017.

[29] R. Salman, J. S. Carlson, F. Ekstedt, D. Spensieri, J. Torstensson, and R. Söderberg, "An industrially validated CMM inspection process with sequence constraints," *Proc. CIRP*, vol. 44, pp. 138–143, May 2016.

[30] A. Fügenschuh, D. Müllenstedt, and J. Schmidt, "Mission planning for unmanned aerial vehicles," Brandenburgische Technische Universität Cottbus-Senftenberg, Cottbus, Germany, Rep. 6, 2019.

[31] Z. Yuan, A. Fügenschuh, and A. Martin, "Solving real-world vehicle routing problems using MILP and pgreedy heuristics," Ph.D. dissertation, Diplomarbeit, Technische Universität Darmstadt, Darmstadt, Germany, 2007.

[32] E. F. Flushing, L. M. Gambardella, and G. A. Di Caro, "A mathematical programming approach to collaborative missions with heterogeneous teams," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 396–403.

[33] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, 2004.

[34] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *Int. J. Robot. Res.*, vol. 32, no. 12, pp. 1495–1512, 2013.

[35] E. Nunes, M. Manner, H. Mitiche, and M. Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robot. Auton. Syst.*, vol. 90, pp. 55–70, Apr. 2017.

[36] B. Miloradović, B. Curuklu, M. Ekström, and A. V. Papadopoulos, "TAMER: Task allocation in multi-robot systems through an entity-relationship model," in *Proc. Int. Conf. Principles Practice Multiagent Syst.*, 2019, pp. 478–486.

[37] S. C. Sarin, H. D. Sherali, and A. Bhootra, "New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints," *Oper. Res. Lett.*, vol. 33, no. 1, pp. 62–70, 2005.

[38] G. Gutin and A. P. Punnen, *The Traveling Salesman Problem and Its Variations*, vol. 12. New York, NY, USA: Springer, 2006.

[39] A. Ameri, B. Curuklu, B. Miloradovic, and M. Ekström, "Planning and supervising autonomous underwater vehicles through the mission management tool," in *Proc. Global OCEANS*, 2020, pp. 1–7.

[40] J. E. Mitchell, "Branch-and-cut algorithms for combinatorial optimization problems," *Handbook of Applied Optimization*, vol. 1. Heidelberg, Germany: Springer, 2002, pp. 65–77.

[41] J. Clausen, "Branch and bound algorithms-principles and examples," Dept. Comput. Sci., Univ. Copenhagen, Copenhagen, Denmark, Rep., 1999.

[42] B. M. Baker and M. A. Ayechew, "A genetic algorithm for the vehicle routing problem," *Comput. Oper. Res.*, vol. 30, no. 5, pp. 787–800, 2003.

[43] B. Nisha and C. Nathi Ram, "Genetic algorithm applications on job shop scheduling problem: A review," in *Proc. Int. Conf. Soft Comput. Techn. Implement.*, 2015, pp. 7–14.

[44] A. H. Brie and P. Morignot, "Genetic planning using variable length chromosomes," in *Proc. Int. Conf. Autom. Plan. Schedul.*, 2005, pp. 320–329.

[45] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Automated configuration of mixed integer programming solvers," in *Proc. Int. Conf. Integr. Constrct. Progr. AI Oper. Res.*, 2010, pp. 186–202.

Branko Miloradović (Student Member, IEEE) received the B.Sc. and M.Sc. degrees in mechanical engineering from the University of Belgrade, Belgrade, Serbia, in 2008 and 2010, respectively. He is currently pursuing the Ph.D. degree in computer science with Mälardalen University, Västerås, Sweden.

His research interests include robotics, multirobot mission planning, combinatorial optimization, and evolutionary algorithms.

Baran Çürüklü received the M.Sc. and Ph.D. degrees in computer science from Mälardalen University, Västerås, Sweden, in 1998 and 2005, respectively.

He is a Senior Lecturer of Artificial Intelligence with Mälardalen University. His current research interests include knowledge representation in cortical structures, collaboration in heterogenous teams of artificial and human agents, and evolution in complex systems.

Mikael Ekström (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in physics from Uppsala University, Uppsala, Sweden, in 1993 and 1999, respectively.

He is a Professor of Robotics with Mälardalen University, Västerås, Sweden, where he is the Head of the Robotics Research Group. His research interests include robotics, autonomous vehicles, sensors, and communication.

Alessandro Vittorio Papadopoulos (Senior Member, IEEE) received the B.Sc. and M.Sc. (*summa cum laude*) degrees in computer engineering and the Ph.D. degree (Hons.) in information technology from the Politecnico di Milano, Milan, Italy, in 2008, 2010, and 2013, respectively.

He is an Associate Professor of Computer Science with Mälardalen University, Västerås, Sweden. His research interests include robotics, control theory, real-time systems, and autonomic computing.