Received March 31, 2021, accepted April 13, 2021, date of publication April 30, 2021, date of current version May 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3076919

Spatially-Distributed Missions With Heterogeneous Multi-Robot Teams

EDUARDO FEO-FLUSHING[1], LUCA MARIA GAMBARDELLA[2], AND GIANNI A. DI CARO[1]
[1]Department of Computer Science, Carnegie Mellon University Qatar, Doha CH-6962, Qatar
[2]Dalle Molle Institute for Artificial Intelligence (IDSIA USI-SUPSI), 6962 Lugano, Switzerland

Corresponding author: Eduardo Feo-Flushing (efeoflus@andrew.cmu.edu)

This work was made possible by Sinergia grant CRSI22_133059 (project SWARMIX) from the Swiss National Science Foundation (SNSF) and by National Priorities Research Program (NPRP) grant 10-0213-170458 from the Qatar National Research Fund (a member of Qatar Foundation). The findings herein reflect the work, and are solely the responsibility of the authors.

ABSTRACT This work is about mission planning in teams of mobile autonomous agents. We consider tasks that are spatially distributed, non-atomic, and provide an utility for integral and also partial task completion. Agents are heterogeneous, therefore showing different efficiency when dealing with the tasks. The goal is to define a system-level plan that assigns tasks to agents to maximize mission performance. We define the mission planning problem through a model including multiple sub-problems that are addressed jointly: task selection and allocation, task scheduling, task routing, control of agent proximity over time. The problem is proven to be NP-hard and is formalized as a mixed integer linear program (MILP). Two solution approaches are proposed: one heuristic and one exact method. Both combine a generic MILP solver and a genetic algorithm, resulting in efficient anytime algorithms. To support performance scalability and to allow the effective use of the model when online continual replanning is required, a decentralized and fully distributed architecture is defined top-down from the MILP model. Decentralization drastically reduces computational requirements and shows good scalability at the expenses of only moderate losses in performance. Lastly, we illustrate the application of the mission planning framework in two demonstrators. These implementations show how the framework can be successfully integrated with different platforms, including mobile robots (ground and aerial), wearable computers, and smart-phone devices.

INDEX TERMS Multi-robot systems, mobile robots, cooperative systems, planning, decision support systems, optimization methods, genetic algorithms, mathematical programming.

I. INTRODUCTION

Heterogeneous multi-agent teams, combining diverse types of physical agents[1] such as robots, humans, and animals, are becoming a viable solution to tackle complex, spatially-distributed real-world problem scenarios. Teaming-up physical agents with different cognitive and sensory-motor skills can naturally provide heterogeneity and redundancy of resources, parallelism, and distribution, making it highly suited for real-world missions in scenarios such as search and rescue, marine environmental monitoring, and surveillance.

In this work we consider *weakly cooperative systems* [40], where the system-level performance can be boosted by an underlying scheme of cooperation, but cooperation is not strictly necessary for mission achievement. Our high-level goal is to define mission plans such that agents get effectively coordinated and positive team-level synergies arise.

A typical mission can be conveniently represented in terms of a *set of tasks* where tasks can have dependencies, priorities, time windows, utilities, costs, and so on. Traditionally, the concept of task is related to the actions that agents execute, without attaching to it a notion of spatial locality. Instead, in this work we emphasize that in many mission scenarios of practical interest, tasks are related to specific spatial locations, where the task 'resides' and must be serviced. Therefore, we consider *missions composed by spatially distributed tasks*, where the completion of each task can deliver a given mission-level *utility*. We focus on the realistic case when the mission is constrained by *limited time budget*, such that some tasks may be left uncompleted, making it necessary to select which tasks to service during the available mission time with the goal of *maximizing the overall utility*.[2] Moreover, we consider tasks that are, in general, *not atomic*: they can be brought to completion incrementally, possibly by different

The associate editor coordinating the review of this manuscript and approving it for publication was Yilun Shang.

[1]Henceforth we use the term *agent* and *robot* interchangeably

[2]We also consider the complementary scenario when mission time is unconstrained. In this case, the goal is to minimize the *makespan*.

agents and at different times, where a partial task completion provides some positive utility.

Under these premises, we define *mission planning* as the assignment of subsets of spatially distributed tasks to *heterogeneous* agents with the aim of maximizing a system-level utility (assuming that task utilities can be composed additively). The agents are assumed to be single-task [23] and need to be *mobile* for switching from one task to another, with an associated traveling cost. The mission planning problem breaks down into three sub-problems that need to be addressed jointly: *task selection and allocation*, *task scheduling*, and *task routing*.

Task selection and allocation is about choosing and assigning a subset of tasks to each one of the agents while optimizing the matching between task and agent skills (sensory-motor, cognitive). Given the assumed spatially distributed nature of the tasks, task routing defines the execution sequence of the tasks assigned to each agent, which in turn defines the routes the agents use to travel from one task to the other. Additional space-time constraints can be taken into account when relations about the *proximity* among agents (e.g., be close to favor communications and collaborations, be far from each other to minimize interference) need to be enforced or promoted. Finally, task scheduling defines how much time an agent should spend on each task. E.g., the workload of a task can be divided among multiple agents, servicing it at different times, if it makes the overall mission plan more efficient.

In this work we tackle the team mission planning problem as described above, addressing the *joint* solution of task selection and allocation, scheduling, routing, and proximity relations. We provide a *formal definition* of the problem and prove that it is NP-hard even if all tasks are non-atomic. This is a central contribution since previous work has considered atomic tasks and, while problems with atomic tasks are usually NP-hard, a deeper understanding of the computational complexity is needed of problems with non-atomic tasks.

We introduce a mathematical formulation of the mission planning problem as a *mixed-integer linear program* (MILP). We present both *heuristic* and *exact* solution algorithms that are computationally efficient. We also introduce a *decentralized method* that allows to solve the problem in a fully distributed way and shows to be scalable and usable in an online replanning scheme (e.g., during mission execution, if unexpected events or deviation happens, it might be necessary to replan on the spot). A number of *computational experiments* are performed to study efficiency and efficacy of the proposed solution methods as well as to compare their performance against state-of-the-art methods. Simulations are used to study the properties of the decentralized implementation. Finally, we describe *two real-world deployments* that illustrate the application of the proposed model and methods and highlight their versatility.

More precisely, the contributions of this paper can be summarized as follows:

- It presents an original formal definition of the *spatial task allocation and scheduling problem in heterogeneous multi-robot teams* (STASP-HMR) and its formulation as a MILP.
- It extends the basic problem definition STASP-HMR with a set of linear formulations for representing *spatio-temporal proximity relations* (e.g., networking, interference) among the agents.
- It provides the formal proof for the *NP-hardness* of STASP-HMR.
- It proposes an original heuristic, in the form *matheuristic* [35], for tackling the computational and scalability challenges of STASP-HMR. The matheuristic features evolutionary operators that embed MILP formulations and solvers. In this way, it is able to accommodate multiple variants and time-requirements of the problem while keeping low the computational demands.
- It proposes an *anytime exact algorithm* that finds solutions with formal guarantees on optimality based on the synergistic combination of matheuristic solvers and MILP solvers that dynamically cooperate by exchanging incumbent solutions. Through empirical analysis, we show that this combination allows both components to speed up finding good quality solutions and outperforms a state-of-the-art method.
- It describes a *distributed and decentralized architecture* based on implicit coordination [28], which shows to be scalable and can be employed for *online replanning* at the expenses of moderate losses in performance.

The paper is organized as follows. Related work is discussed in the next section. The formal model of the STASP-HMR is introduced in Section III. The NP-hardness of the STASP-HMR is proved in Section IV. The STASP-HMR is mathematically formulated as a MILP in Section V. The spatio-temporal relations are formalized in Section VI. The solutions methods are presented in Section VII (matheuristic) and in Section VIII (anytime, exact algorithm). Centralized and decentralized architectures for online replanning are presented in Section IX and X, respectively. Computational experiments and results are discussed in Section XI. Real-world demonstrators are described in Section XII. Finally, Section XIII concludes the paper and highlights some possible future research directions.

II. RELATED WORK

In its most basic form, the problem of selecting and allocating sequences of tasks to a team of robots can be formulated as a *Vehicle Routing Problem* (VRP), a well-known family of combinatorial optimization problems. VRPs are widely studied in the field of Operations Research (OR), and relate to the transportation of goods between depots and customers by means of a fleet of vehicles. The solution of a VRP determines which customers are served by each vehicle and which routes the vehicles should follow in order to minimize transportation costs. In *VRPs with profits* [16] a profit or reward is associated with each customer and the aim is maximizing the total collected reward. A time limit is imposed to each route, such that some customers may be left unvisited. A solution to VRPs with profits thus also involves a selection of customers to be served. Models of VRPs with profits include the *Multiple Tour Maximum Collection Problem* [7] also known as the *Team Orienteering Problem* (TOP) [9].

The TOP has been the starting point of many planning models for complex multi-agent missions, such as surveillance [37], disaster relief with unmanned aerial vehicles [53], underwater inspection [49], and precision irrigation in agriculture [48]. Complex variants of TOP, inspired by real-world applications, have been proposed, such as grouping of customers [50], split service [2], curvature-constrained vehicles [41], time-dependent [51], and time-varying profits [34]. Some works combine aspects from scheduling problems [8], [25], [32], [43] such as temporal dependencies between customers/tasks.

Although our modeling approach leverages the works described above, we consider several aspects that are motivated by real-world applications, and which have not previously been studied in-depth. Among these we address the integration of scheduling decisions into the routing problem, rewards dependent upon decidable service time, and strong inter-dependencies between vehicles/agents. All these aspects represent new challenges for VRPs [15].

In the robotics domain, *multi-robot task allocation* (MRTA) problems address scenarios similar to those considered in VRPs. However, considered assumptions, goals, and constraints are usually slightly different, accounting for a number of dynamic and uncertain aspects. A seminal taxonomy of MRTA problems was defined in [23], showing the relations with *mathematical optimization models* such as vehicle routing, assignment, and set problems. The taxonomy was extended in [31] to characterize interrelated utilities and constraints, and in [39] to categorize the constraints between the schedules of the robots.

These categorizations have provided a way to formalize complex MRTA scenarios, and have also revealed the NP-hard nature of MRTA problems. As a result, finding computationally-affordable solution approaches providing performance guarantees is inherently challenging when the scenario is realistically complex. Nevertheless, the use of optimization formulations is a sound approach to the solution of MRTA problems when formal guarantees are expected. Notable examples include [30], where a MILP formulation for a multi-robot coordination problem with temporal inter-task constraints is provided and tackled using a generic solver. In [32] a similar problem has been modeled using a set partitioning formulation and solved using a branch-and-price. The drawback of these approaches is the lack of scalability, as well as the difficulty to formalize complex scenarios.

To overcome these limitations, the use of *agent-based* models and of *heuristics*, such as in [25], [41], [47], [52], can be of great help to tackle complex and large problems. Among these works, Gombolay *et al.* [25] developed *Tercio*, a centralized multi-agent task assignment and scheduling algorithm for coordination problems with temporospatial constraints. They combined a heuristic task sequencer with a MILP solver to compute multi-agent mission plans that satisfy precedence and temporal and spatial-proximity constraints. The most distinctive characteristic of Tercio is the decomposition of the problem into task assignment (tackled with a MILP solver) and task routing and scheduling (tackled with a heuristic). The authors empirically showed that the Tercio achieves results close to the optimal for real-world problems, providing better-quality solutions than prior state-of-the-art techniques. However, Tercio, as well as most of the aforementioned agent-based approaches, does not provide any formal guarantees for solution quality.

MILP-based formulations and solutions have been predominantly used in multi-robot task allocation problems [3], [5], [12], [32], [33], Although these models are computationally hard to solve to optimality, they are particularly attractive due the wide availability of powerful solvers with *anytime* property: solutions are progressively improved, and sub-optimal solutions with *quality guarantees* can be obtained at any point in time before the solver finds the optimal one [32]. Furthermore, *heuristics* and *meta-heuristics* can be coupled with MILP solvers to handle large problem instances and to improve the computational efficiency of the solution method. For instance, meta-heuristics have been proved to be successful at tackling problems that involve the routing of multiple agents such as Vehicle Routing problems [26]. In this work we follow this direction, and consider a mathematical optimization framework, precisely because it can enjoy *anytime* properties even when using standard solvers (based on various branch-and-bound variants). We also propose two solution algorithms: one *exact* method and one heuristic, in the form of a *matheuristic*. Both approaches deal with the computational complexity, improve the scalability of the solution approach, and can flexibly accommodate the addition / dropping of operational constraints. To the best of our knowledge, this is the first work that proposes matheuristics for multi-robot task allocation problems.

Most of previous MRTA works have been proposed under the restrictive assumptions that tasks are *atomic* procedures [11], [30], [42], [44], [45] and require uninterrupted agent effort. In this work we remove these assumptions to allow the computation of – possibly more efficient – solutions in which tasks are accomplished in an *incremental* manner over disjoint periods of time during which different agents can devote some effort into them. To the best of our knowledge this is the first work that effectively addresses the issue of *non-atomic tasks* and that deals with the problem of simultaneously defining *task allocation* and *task schedules with spatio-temporal dependencies*. Furthermore, we show the problem is NP-*hard* even if all tasks are non-atomic.

To tackle scalability issues, a number of MRTA works have proposed *distributed* approaches. Market-based methods [4], [14], [38] rely on negotiation techniques (e.g., auctions) and usually have both centralized and distributed elements, hence requiring *explicit coordination* among the agents (e.g., to simultaneously engage in the auctioning phase managed by an auctioneer agent). Alternatively, swarm intelligence methods rely on *self-organized* behaviors and are usually fully distributed and decentralized [6], hence requiring *no explicit coordination* or communication among the agents. However, this ease of design comes at the expenses of performance. In between explicit coordination and no coordination/communication is *implicit coordination*, which is a flexible way to deal with loosely-coupled missions [28]. It is achieved by having each agent, independently, computing a (global) solution utilizing a replica of a centralized planner. In this work we present both a *centralized* and a *decentralized* scheme for mission planning. The latter is designed after implicit coordination. It is aimed at minimizing computational and communication costs for individual agents and, at the same time, minimizing the impact of the loss of coordination due to the decentralization. We show that implicit coordination is effective in decentralizing the solution of the STASP-HMR and in providing scalability (of computations and communications) while incurring in a relatively small loss compared to a centralized architecture. Using the decentralized scheme we show that frequent *online replanning* to adapt to mission evolution is computationally feasible and effective.

III. A MODEL FOR COOPERATIVE MISSION PLANNING WITH HETEROGENEOUS TEAMS

A *mission* is defined as a set $\mathcal{T}$ of *spatially distributed, location-dependent tasks*. We assume that the tasks in the set $\mathcal{T}$ are the result of a given decomposition. A task corresponds to the execution of a *particular action* at a *specific location* (or portion) of the environment.[3] Tasks can be *non-atomic*, enabling the possibility of being completed by multiple agents, each providing individual contributions over time.

A set of resources A (*mobile agents*) is available for the mission. Each agent can perform the different tasks to some extent (including the case where an agent is not suited to deal with a task) and with an *agent-specific time efficiency*. An agent can only deal with one task at-a-time. Moving between tasks/locations usually incurs into a *traveling cost*. We look at tasks as *requests to be serviced*, and at the agents as the entities with the capabilities to service these requests. We assume that the mapping defining the time efficiency for each (agent, task) pair is available.

The two different mission planning objectives that we consider in this work are: (i) maximizing team performance given a limited time budget, and (ii) minimizing the time required to complete all the tasks. Henceforth, we will focus on the former because it represents more realistic scenarios. Moreover, in dynamic environments, it is suitable for implementing *mission replanning*, as described later in Section IX. Nevertheless, given that the latter mission objective is more common in the literature, we consider it to compare our solution approach against state-of-the-art mission planning algorithms in Section XI-F.

[3]Under this definition, tasks can also be seen as a pair (*action*, *location*)

In the time-budgeted problem, the team performance is defined in terms of the utility collected by the team during the *time budget* T. For each task $i \in \mathcal{T}$, we define a *reward* value R_i that quantifies the utility of servicing the task for the mission. The *mission utility* is defined as an additive, undiscounted function of these rewards. Note that, given the limited time budget, not necessarily all tasks can be dealt with (or brought to completion). Task rewards can also be seen as a way to prioritize the service provided by the agents.

The goal of *mission planning* consists in maximizing the mission utility by jointly solving the following sub-problems.

- *Task selection:* select a subset of tasks to perform given the time budget and the resources available in the team;
- *Agent task allocation*: assign the selected tasks to specific agents within the team, based on their current status and their efficiency dealing with the tasks;
- *Task routing:* for each agent, define the sequence for dealing with the assigned tasks and, therefore, for moving (with a cost) from one spatial location to another;
- *Time scheduling:* appoint the duration of the services provided to each one of the selected tasks by each agent.

In the rest of this section, we detail the concepts that compose this core model for mission planning and introduce the notation that is used in the rest of the paper.

A. TASK GRAPH AND TIME STEPS

Tasks in set $\mathcal{T}$ are not necessarily spatially disjoint, and we assume there are no temporal dependencies related to the tasks (e.g., execution time-windows, or deadlines). The spatial layout is captured by a *traversability graph* $G = (\mathcal{T}, E)$ where E contains an arc (i, j) if task j can be executed right after task i. The graph G imposes constraints over the sequences of tasks that can be executed. This is, for instance, the case when some tasks cannot be designated immediately after others, e.g., because they are very distant from each other, or when specific tasks must be serviced immediately before servicing others.

All aspects concerning *time* (i.e., traveling, task servicing) are discrete: the time range $[0, T]$ of the time budget is uniformly *discretized* into a sequence of intervals of length Δ_T. Therefore, agents only spend an integral number of time steps servicing a task or switching from one task to another.

B. TASK EFFICIENCY MODELS AND COMPLETION MAP

The diversity in the agent team A is captured through *task efficiency models* that, given $a \in A$ and $i \in \mathcal{T}$, define the efficiency (i.e., amount of work accomplished over time) with which agent a services task i. The intuition is that any progress in the completion of a task depends on the time spent on it. For a given task i, we say that agent a is more efficient than agent b if, devoting the same amount of time to task i, a would complete more of i's workload.

We assume that task efficiency models are known for each agent. We consider *linear* task efficiency models, represented by *performance functions* $\varphi_k : \mathcal{T} \mapsto [0, 1]$. The value of $\varphi_k(i)$ can be seen as the rate at which the workload of task $i \in \mathcal{T}$ decreases over time when agent $k \in A$ is executing it. These values are normalized with respect to the initial workload of the tasks. We also assume that the efficiency models are additive: when agents $A' \subseteq A$ work simultaneously on the same task i for t time steps, the decrease in workload of i is equal to $\sum_{k \in A'} \varphi_k(i)t$.

Given the non-atomicity of the tasks, the *completion map* $C : \mathcal{T} \mapsto [0, 1]$ is used to express the *fractional residual workload* of each task. C identifies the *current completion level* of tasks whose workload has been partially addressed so far. For instance, a value of $C(i) = 0$ indicates that task i has already been completed in the past, and therefore no further effort from the agents is required. If an agent attempts to further deal with a completed task, no additional reward is provided, which will amount to a waste of time and resources. Servicing $p \cdot 100$ percent of the workload of task i decreases its required completion $C(i)$ by p and provides a partial utility of pR_i.

C. SPATIAL TASK ALLOCATION AND SCHEDULING PROBLEM IN HETEROGENEOUS MULTI-ROBOT TEAMS (STASP-HMR)

Based on the above concepts and assumptions, the *spatial task allocation and scheduling problem in heterogeneous multi-robot teams*, or STASP-HMR in short, is formally stated as follows. Given a set A of heterogeneous agents, characterized by their task performance functions φ, a set of assignable tasks $\mathcal{T}$, a traversability graph G, a set of initially accessible tasks, and a given *limited time budget*, or mission time span, T, the STASP-HMR consists in determining a mission plan – joint plans for the activities of the agents in the environment – that maximizes a mission utility as the sum of all gathered rewards. Note that, as pointed out at the beginning of this section, a companion definition can be given for the case when the mission time is unconstrained, such that the goal is to minimize the time cost for completing all tasks. We will come back to this formulation in Section XI-F.

A solution to the STASP-HMR is a mission plan that is represented by a set of tasks a_k, task routes p_k, and schedules s_k for each agent k in team A. We denote a mission plan as $\mathcal{P} = \{\mathcal{P}_k \mid k \in A\}$, where $\mathcal{P}_k = < a_k, p_k, s_k >$ denotes the plan (i.e., assigned tasks, route, and schedule) corresponding to agent k. The sets $a_k \subseteq \mathcal{T}$ indicate the tasks that are assigned to agent k. Each route p_k is the sequencing of tasks a_k that is valid path in G. Schedules $s_k : a_k \mapsto \mathbb{N}$ are time assignments that define the total amount of time each of the selected tasks will receive. Note that due to the limited time budget, not all tasks may be performed: a mission plan implicitly defines a selection $\bigcup_{k \in A} a_k \subseteq \mathcal{T}$ among the tasks. All routes p_k must start with a task that belongs to the set $\mathcal{T}_0$ of initially accessible tasks (e.g., the locations of mission's control centers). We further assume that a *task cannot appear more than once on a single route*. This assumption corresponds to the sub-tour elimination constraint in VRPs, which is fully justified when traveling costs among tasks satisfy the triangle inequality.

TABLE 1. Summary of notation.

Element	Description
$\mathcal{T}$	Tasks composing a mission
$\mathcal{T}_0$	$\subseteq \mathcal{T}$ Initial tasks
R_i	Reward for task $i \in \mathcal{T}$
G	Traversability graph $(\mathcal{T}, E)$
A	Agents
$\mathcal{P}$	A mission plan
$\mathcal{P}_k$	Plan corresponding to agent $k \in A$
a_k	Tasks assigned to $\mathcal{P}_k$
p_k	Sequence of tasks belonging to $\mathcal{P}_k$
s_k	Schedules corresponding to $\mathcal{P}_k$
$C(i)$	Completion level of task $i \in \mathcal{T}$
Δ_T	Length of time interval
T	Mission time span
$\varphi_k(i)$	Performance of agent $k \in A$ in executing task $i \in \mathcal{T}$

Fig. 1 shows an example plan for 3 agents and 16 tasks located on a rectangular grid. Here tasks correspond to grid cells and we assume that agents can only move between adjacent cells (eight-connected grid) without incurring in traveling costs. Since in this example each task is associated to a unique cell, we denote tasks as τ_{ij}, where $0 \le i, j < 4$ indicate, respectively, the column and row of the cell of the task in the grid. Columns are numbered from left to right and rows from bottom to top. Hence τ_{00} is the bottom-left-corner cell. The set of initial tasks $\mathcal{T}_0$ contains τ_{00} and its adjacent tasks. A mission plan is defined for a time budget of $T = 5$ mission intervals with the following assignments: $a_1 = \{\tau_{00}, \tau_{01}, \tau_{11}, \tau_{21}, \tau_{30}\}$, $a_2 = \{\tau_{11}, \tau_{02}, \tau_{13}, \tau_{22}, \tau_{31}\}$, $a_3 = \{\tau_{10}, \tau_{21}, \tau_{32}, \tau_{33}\}$, and the following routes: $p_1 = \langle \tau_{00} \to \tau_{01} \to \tau_{11} \to \tau_{21} \to \tau_{30} \rangle$, $p_2 = \langle \tau_{11} \to \tau_{02} \to \tau_{13} \to \tau_{22} \to \tau_{31} \rangle$, $p_3 = \langle \tau_{10} \to \tau_{21} \to \tau_{32} \to \tau_{33} \rangle$, for agents 1, 2, and 3, respectively. Here the schedules of all agents allocate one time unit to all tasks with the sole exception of τ_{10}, which is assigned to agent 3 for two mission intervals. Initially, all tasks require full service: $C(\tau) = 1, \ \forall \tau$. The execution of the mission plan decreases the completion map as shown in the bottom of the figure. For instance, task τ_{00} is completed by agent 1, and task τ_{11} is completed after being serviced by agents 1 and 2. Note that some of the tasks do not receive any service (e.g., τ_{03}), while others are completed (e.g., τ_{00}), or partially completed (e.g., τ_{30}). Assuming uniform task rewards $R_\tau = 1$, the mission utility is equivalent to total decrease of workload of the entire set of tasks: $\sum_{\tau \in \mathcal{T}} (1 - C(\tau)) R_\tau = 10$.

IV. COMPUTATIONAL COMPLEXITY OF STASP-HMR

The stated STASP-HMR problem is NP-hard, even in the restricted case when all tasks are non-atomic. The formal proof of this core result is given in the rest of this section.

We only consider the single agent case, the complexity result of the multiple agent problem follows from this result. As an intermediate step in our proof, we first show that a variant of the STASP-HMR that considers all tasks as atomic

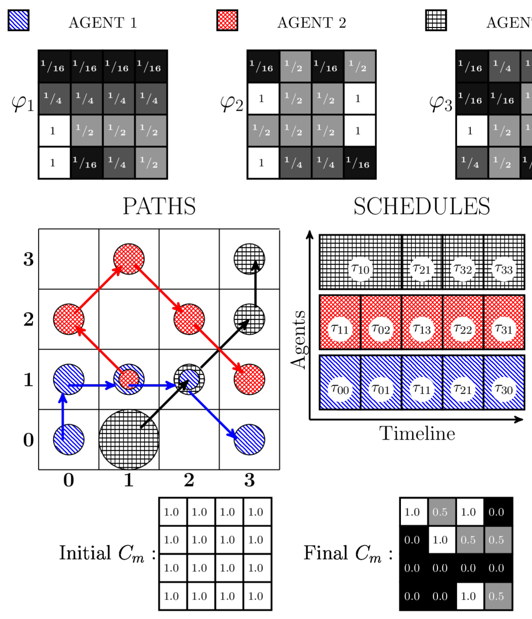


FIGURE 1. Example plan for 3 agents. The mission is composed of 16 tasks arranged on a rectangular grid. All agents start from the bottom left corner of the area and $T = 5$. On the top, the performance functions φ_k for each agent. Paths are depicted in the left of the middle row of the figure, while schedules are depicted as a timeline in the right. On the bottom, the initial (left), and final (right) completion maps.

procedures is NP-hard. Finally, this result is used to show that the STASP-HMR is at least as hard as its atomic variant.

Let $\mathbf{W}(\tau)$ be the amount of time it takes to complete a task τ estimated from the performance function $\varphi(\tau)$. The STASP-HMR with atomic tasks, STASP-HMR-AT in short, imposes the restriction the any plan that includes a task τ must assign a time equal to $\mathbf{W}(\tau)$.

To prove the NP-hardness of the STASP-HMR-AT we reduce from the Partition Problem [22], in particular the 2-PARTITION Problem. Given a set of positive integers $V = \{v_1, v_2, \ldots, v_n\}$, with $\sum_{v_i \in V} v_i = 2A$, the 2-PARTITION problem aims at finding a partition of V into two sets $V_1, V_2 \subset V$ such that $\sum_{v_i \in V_1} v_i = \sum_{v_j \in V_2} v_j = A$ or determining that such partition does not exists. From an instance of 2-PARTITION $\mathcal{J}$, we define an instance $\mathcal{J}'$ of the STASP-HMR-AT as follows. First, define two sets $\mathcal{T}_1, \mathcal{T}_2$, each consisting of n dummy tasks, where each task corresponds to one element in V. Intuitively, tasks in $\mathcal{T}_1 = \{\tau_{11}, \tau_{12}, \ldots, \tau_{1n}\}$ and $\mathcal{T}_2 = \{\tau_{21}, \tau_{22}, \ldots, \tau_{2n}\}$ are used to indicate whether an element in V belongs to the set V_1 or V_2. When τ_{1i} is included in a plan, then v_i belongs to V_1. Similarly for τ_{2i}. Let M be an integer greater than $2A$. We let $R_{\tau_{1i}} = M - v_i$, and $R_{\tau_{2i}} = M$, whereas completion times $\mathbf{W}(\tau_{1i}) = M$, and $\mathbf{W}(\tau_{2i}) = M + v_i$. The traversability graph G is acyclic and has a level structure, where each level i contains tasks τ_{1i} and τ_{2i}. The first level contains tasks τ_{11} and τ_{21}, which are source vertices of the graph and also constitute the set of initial tasks $\mathcal{T}_0$. We connect each pair of consecutive levels as shown in Fig. 2. Lastly, we set the mission time span $T = nM + A$. Before we prove that a solution to $\mathcal{J}'$ yields a solution to $\mathcal{J}$, we introduce and prove the following lemmas.

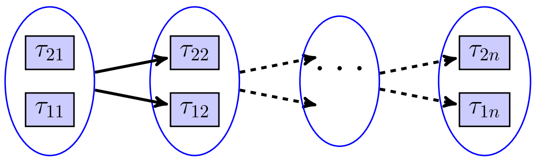


FIGURE 2. DAG construction to define the traversability graph that reduces a 2-PARTITION instance to an instance of the STASP-HMR-AT.

Lemma 1: Let $\mathcal{T}_1' \subseteq \mathcal{T}_1$ and $\mathcal{T}_2' \subseteq \mathcal{T}_2$ be the subset of tasks that are included in the optimal solution to $\mathcal{J}'$. Then $|\mathcal{T}_1'| + |\mathcal{T}_2'| = |V| = n$

Proof: We note that, given the way G is constructed and $M > 2A$, the total reward of any plan that reaches level l in G is always greater or equal to $(l-1)M$ and less or equal to lM. As a result, any plan of length l provides higher reward than any plan of length $l-1$. Now let us consider the plan $\mathcal{P}$ with $\mathcal{T}_1' = \mathcal{T}_1$ that reaches the n-th level. Clearly, $\mathcal{P}$ is always feasible for $T = nM + A$, and $|\mathcal{T}_1'| = n$ because its path reaches the last level of G. Furthermore, we note that the reward provided by $\mathcal{P}$ is equal to $nM - 2A$. Therefore, any plan with $|\mathcal{T}_1'| + |\mathcal{T}_2'| < n$ cannot be an optimal solution to $\mathcal{J}'$ because there always exists $\mathcal{P}$ with $|\mathcal{T}_1'| + |\mathcal{T}_2'| = n$ that is feasible, and has a greater value. □

Lemma 2: Let $\mathcal{T}_1' \subseteq \mathcal{T}_1$ and $\mathcal{T}_2' \subseteq \mathcal{T}_2$ be the subset of tasks that are included in the optimal solution to $\mathcal{J}'$. Then $\sum_{\tau_{1i} \in \mathcal{T}_1'} v_i + \sum_{\tau_{2i} \in \mathcal{T}_2'} v_i = 2A$.

Proof: It follows immediately from Lemma 1 that, given the way G is constructed, any optimal solution must reach the last level of G. As a result, the optimal solution to $\mathcal{J}'$ involves n tasks and must include either τ_{1i} or τ_{2i} for all v_i. □

Lemma 3: Let $\mathcal{T}_1' \subseteq \mathcal{T}_1$ and $\mathcal{T}_2' \subseteq \mathcal{T}_2$ be the subset of tasks that are included in an optimal solution to $\mathcal{J}'$. Then, $\sum_{\tau_{2i} \in \mathcal{T}_2'} v_i \leq A$.

Proof: Note that the time span of the optimal solution to $\mathcal{J}'$ is defined as $T(\mathcal{J}') = \sum_{\tau_{1i} \in \mathcal{T}_1'} \mathbf{W}(\tau_{1i}) + \sum_{\tau_{2i} \in \mathcal{T}_2'} \mathbf{W}(\tau_{2i})$. From Lemma 1 it follows that $T(\mathcal{J}') \leq nM + \sum_{\tau_{2i} \in \mathcal{T}_2'} v_i$. Since $T = nM + A$, it follows that $\sum_{\tau_{2i} \in \mathcal{T}_2'} v_i \leq A$ □

Lemma 4: Let $\mathcal{T}_1' \subseteq \mathcal{T}_1$ and $\mathcal{T}_2' \subseteq \mathcal{T}_2$ be the subset of tasks that are included in an optimal solution to $\mathcal{J}'$. Then, the optimal value is bounded by $nM - A$.

Proof: Note that the optimal value of $\mathcal{J}'$ can be expressed as $\mathbf{OPT}(\mathcal{J}') = \sum_{\tau_{1i} \in \mathcal{T}_1'} R_{\tau_{1i}} + \sum_{\tau_{2i} \in \mathcal{T}_2'} R_{\tau_{2i}}$. From Lemma 1 it follows that $\mathbf{OPT}(\mathcal{J}') = (|\mathcal{T}_1'| + |\mathcal{T}_2'|) M - \sum_{\tau_{1i} \in \mathcal{T}_1'} v_i$. From Lemma 2 and Lemma 3, it then follows that $\mathbf{OPT}(\mathcal{J}') \leq nM - A$. □

Using the previous lemmas, we now prove that the STASP-HMR-AT is NP-hard.

Theorem 1: The STASP-HMR-AT is NP-hard.

Proof: We show that the instance $\mathcal{J}'$ of the STASP-HMR-AT described above has a solution with value equal to $nM - A$ **iff** the instance $\mathcal{J}$ of 2-PARTITION has a solution.

It is apparent that if $\mathbf{OPT}(\mathcal{J}') = nM - A = (|\mathcal{T}_1'| + |\mathcal{T}_2'|) M - \sum_{\tau_{1i} \in \mathcal{T}_1'} v_i$ then, by Lemma 1,

$\sum_{\tau_{1i} \in \mathcal{T}_1'} v_i = A$. It follows from Lemma 2 that $\sum_{\tau_{2i} \in \mathcal{T}_2'} v_i = A$, and therefore we have found a perfect partition $V_1 = \mathcal{T}_1'$, $V_2 = \mathcal{T}_2'$.

Conversely, if there exists a perfect partition V_1, V_2, then a plan with $\mathcal{T}_1' = V_1$ and $\mathcal{T}_2' = V_2$, is feasible and has a value equal to $nM - A$, which is equal to the upper bound given by Lemma 4. Thus, that plan is also optimal. □

Theorem 2: The STASP-HMR is NP*-hard.*

Proof: For any problem instance $\mathcal{K}$ of the STASP-HMR-AT, we can create an instance $\mathcal{K}'$ of the STASP-HMR such that a solution to $\mathcal{K}$ can be derived from a solution to $\mathcal{K}'$.

Let $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_{|\mathcal{T}|}\}$ be the set of tasks of $\mathcal{K}$. Let $\mathcal{T}' = \bigcup_{i \in \mathcal{T}} \{\tau_{i1}, \ldots, \tau_{i\mathbf{W}\tau_i}\}$ be the set of tasks of $\mathcal{K}'$ that includes $\mathbf{W}(\tau_i)$ copies of each task $\tau_i \in \mathcal{T}$. Each τ_{ij} represents a fraction of τ_i that can be completed in a single time unit. For τ_{ij}, we set a reward equal to zero when $j < \mathbf{W}(\tau_i)$, and equal to R_{τ_i} when $j = \mathbf{W}(\tau_i)$. We set $\varphi(\tau_{ij}) = 1$ for all $\tau_{ij} \in \mathcal{T}'$. Let G be the traversability graph of the STASP-HMR-AT instance. We define G' for $\mathcal{K}'$ that enforces the agent to either skip task τ_i or be assigned $\mathbf{W}(\tau_i)$ time units to it. This is done by defining arcs $(\tau_{ij}, \tau_{i(j+1)})$, for all $1 \le j < \mathbf{W}(\tau)$. For all $(\tau_i, \tau_k) \in G$, we define an arc $(\tau_{i\mathbf{W}(\tau_i)}, \tau_{k1})$. Fig. 3 illustrates the idea.

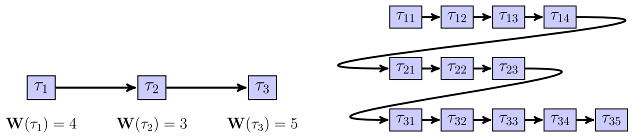

τ_1
τ_2
τ_3
$\mathbf{W}(\tau_1) = 4$
$\mathbf{W}(\tau_2) = 3$
$\mathbf{W}(\tau_3) = 5$
τ_{11} τ_{12} τ_{13} τ_{14}
τ_{21} τ_{22} τ_{23}
τ_{31} τ_{32} τ_{33} τ_{34} τ_{35}

FIGURE 3. Example of traversability graph that reduces STASP-HMR-AT to an instance of the STASP-HMR. Each of the tasks τ_i on the left instance is translated into $\mathbf{W}(\tau_i)$ tasks connected in series.

In order to prove that a solution to $\mathcal{K}'$ yields a solution to $\mathcal{K}$ we consider an injective function that maps the solution space of $\mathcal{K}'$ to the solution space of $\mathcal{K}$, First, let us distinguish two types of plans in $\mathcal{K}'$: (a) plans that include all the copies of all the tasks involved such that all tasks included in the plan are completed, and (b) plans in which the last task τ_{ij} is different from $\tau_{i\mathbf{W}(\tau_i)}$ – i.e., the last task in the plan's sequence is not completed. Plans of type (a) have an apparent equivalent plan in $\mathcal{K}$ due to the atomic condition. Any plan of type (b) is equivalent – in terms of reward – to a plan in which all the *fragments* of the last task are removed, and after this modification, we obtain a plan of type (a) that is contained in $\mathcal{K}$. Clearly, each $\mathcal{P}'$ in the solution space of $\mathcal{K}'$ provides the same reward and has the same time span as its mapped plan $\mathcal{P}$ in the solution space of $\mathcal{K}$. Furthermore, the range of the mapping function is the entire solution space of $\mathcal{K}$. This shows that the STASP-HMR is at least as hard as STASP-HMR-AT, and by Theorem 1, STASP-HMR is NP-hard. □

V. MIXED-INTEGER LINEAR (MILP) FORMULATION OF STASP-HMR

In this section, we formulate the STASP-HMR as a MILP. Such a formulation brings important advantages. First, the mission planning problem can be solved to optimality with formal guarantees using standard solvers. Second, by adding or removing constraints or modifying the objective the formulation can be easily adapted to deal with different scenarios. Furthermore, MILP solvers have *anytime* properties: solutions are progressively and monotonically improved over time and can be retrieved with formal error bounds on optimality based on the MIP gap. The formulation is as follows:

$$\text{maximize} \quad \sum_{i \in \mathcal{T}} R_i \Phi_i \tag{1}$$

$$\text{subject to} \quad \sum_{(0,j) \in E} x_{0jk} = 1 \quad \forall k \in A \tag{2}$$

$$\sum_{(i,0) \in E} x_{i0k} = 1 \quad \forall k \in A \tag{3}$$

$$\sum_{(i,j) \in E} x_{ijk} = \sum_{(j,i) \in E} x_{jik} = y_{jk} \quad \forall k \in A,\ j \in \mathcal{T} \tag{4}$$

$$t_{ik} + w_{ik} - t_{jk} \le (1 - x_{ijk})T \quad \forall k \in A, (i,j) \in E \quad i \neq 0,\ j \neq 0 \tag{5}$$

$$y_{ik} \le t_{ik}, w_{ik} \le T y_{ik} \quad \forall k \in A,\ i \in \mathcal{T} \tag{6}$$

$$\Phi_i \le \sum_{k \in A} w_{ik} \varphi_k(i) \quad \forall i \in \mathcal{T} \tag{7}$$

$$0 \le \Phi_i \le C(i) \quad \forall i \in \mathcal{T} \tag{8}$$

$$t_{ik},\ w_{ik} \in \mathbb{N},\ y_{ik} \in \{0,1\} \quad \forall k \in A,\ i \in \mathcal{T} \tag{9}$$

$$x_{ijk} \in \{0,1\} \quad \forall k \in A,\ (i,j) \in E \tag{10}$$

The objective function (1) represents the utility of a mission plan. It is computed as the sum of the rewards R_i provided by each task i weighted by its workload reduction Φ_i achieved by the plan. The continuous variables Φ_i range from 0 (task will be untouched) to C (task will be fully serviced). In order to simplify the formulation, we introduce a dummy task (denoted by 0) that represents the starting point and ending point of the agent paths. In this way, the problem of determining an elementary path p_k (i.e., a feasible sequence of tasks) translates into one of finding a *circuit*, starting and ending at 0. To this end, the traversability graph is extended with arcs from 0 to each of the initially accessible tasks in $\mathcal{T}_0$, and from all elements in $\mathcal{T}$ to the dummy task 0. Constraints (2-4) ensure path continuity. Constraints (5) are derived from the well-known Miller-Tucker-Zemlin sub-tour elimination constraints [36] that eliminate sub-tours and, together with (6), define the bounds on the variables t_{ik} and w_{ik} based on time budget T. Lastly, variables Φ_i denote the total time that agents commit to service task i. The value of Φ_i is equal to $\max(\sum_{k \in A} w_{ik} \varphi_k(i), C(i))$, that translates into constraints (7-8).

VI. MANAGEMENT OF SPATIO-TEMPORAL RELATIONS

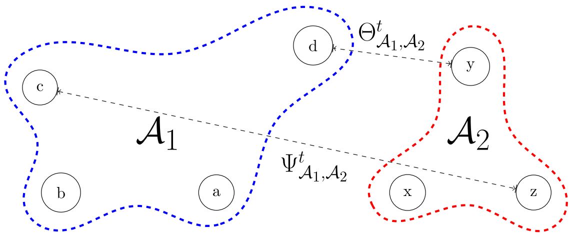
The MILP formulation presented in the previous section provides a basic model for STASP-HMR scenarios. However, in the spatially-distributed missions that we are considering, a mission plan defines the trajectories that the agents will follow and consequently has a direct effect over agent *proximity*. Depending on the application domain, physical proximity among agents during the mission can have a major impact on a real-world mission deployment. Agent proximity can be exploited in various ways or, in some situations, shall be avoided. For instance, in a search and rescue mission, agent proximity can play a core role in the following interaction scenarios:

- *Augmentation*: a human rescuer and a robot that are concurrently exploring close-by areas can share real-time video streams to augment each other's views.
- *Safety*: at night, human agents searching in the wilderness are safer if they stay relatively close to each other.
- *Interference*: air-scent dogs might get distracted by the nearby presence of other teammates or dogs.
- *Networking*: in areas not covered by infrastructure networking, data can be transmitted in a multi-hop way using teammates in radio range.

It is clear that explicitly accounting for these types of interactions during planning can promote or avoid the happening of the situations described above. To this end, we extend the STASP-HMR with a general strategic framework to consider *spatial proximity between groups of agents* during planning. More specifically, we introduce *spatio-temporal directives*, henceforth also referred to as directives, that can be used to impose minimum and maximum separation distances between groups of agents in the mission plans.

A. SPATIO-TEMPORAL DIRECTIVES: COALITION, INTERFERENCE, NETWORK, AND SPARSITY

Let $A_1, A_2 \subseteq A$ be two subsets of agents. With the spatio-temporal directives we are interested in controlling their distances. With $\Theta^t_{A_1,A_2}$ and $\Psi^t_{A_1,A_2}$ we indicate, respectively, the *minimum distance* and the *maximum distance* between the agents in A_1 and A_2 at a time t. These distances change over time depending on the location of the agents in the two sets. Fig. 4 illustrates Θ^t and Ψ^t given the deployment of two groups of agents A_1 and A_2, depicted in blue and red respectively, at an arbitrary point in time t.



d
c
b
a
y
x
z
$\mathcal{A}_1$
$\mathcal{A}_2$
$\Theta^t_{\mathcal{A}_1,\mathcal{A}_2}$
$\Psi^t_{\mathcal{A}_1,\mathcal{A}_2}$

FIGURE 4. Illustrative example of Θ^t and Ψ^t.

A *spatio-temporal directive* establishes a desired lower or upper bound to variables Θ^t or Ψ^t at a specific time step t. Multiple directives can be required simultaneously. A set of directives can be designed to enforce or promote proximity relations over a time interval, or at regular intervals. $\mathbb{T}$ denotes the set of time steps at which at least one directive is defined.

Setting an upper bound to Ψ^t, results in making the agents in A_1 and in A_2 staying within a bounded maximal distance. The lower the bound, the closer the groups will be. This is useful to promote *collaborative behaviors* between two groups of agents, or within one group of agents, by keeping them close to each other. Therefore, we refer to these directives as *coalition* directives, and are denoted as β_c. Instead, if we set an upper bound to Θ^t, there will be at least two agents, one per group, that are within the specified distance. For instance, this bound can represent the *communication range* of the wireless network used by the agents. Thus, these directives can enable data exchange when the agents communicate through a wireless ad hoc network. We refer to them as *network* directives, and denote them as β_n.

Regarding lower bounds, directives related to Θ^t aim at establishing a certain minimum separation distance between two groups of agents. In some domains, these directives can be useful to prevent the agents to interfere with each other while they execute their tasks, or to avoid dangerous situations, and undesirable events that are likely to occur when agents get too close to each other. These directives are called *interference-avoidance* directives and are denoted as β_i. Finally, the *sparsity* directives, are those where Ψ^t is required to be greater than a certain value. Their effect is that of spreading the groups over larger areas. These directives are denoted as β_s.

B. MILP FORMULATION OF DIRECTIVES

The formulation of the above directives and their inclusion in the MILP require three steps to adapt the basic formulation given in 1-10. First, we need to introduce *time-indexed variables* to represent the location of any agent at a particular time step. Next, based on the location of the agents, we define a set of variables that represent their distances. Finally, we define a set of binary variables – one per each directive – that take a value of one if the directive holds in the resulting mission plan, zero otherwise. In the following, we show how these variables are defined and included in the MILP to promote or enforce directives.

1) TIME-INDEXED MODEL

Let y^t_{ik} be binary time-indexed helper variables that take value 1 iff agent k is assigned to task $i \in \mathcal{T}$ at time step $t \in \mathbb{T}$. We want to enforce $y^t_{ik} = 1 \Leftrightarrow (t_{ik} \leq t < t_{ik} + w_{ik})$. This translates into the following set of linear constraints:

$$(T - t)\, y^t_{ik} + t_{ik} \leq T \quad \forall k \in A, i \in \mathcal{T}, t \in \mathbb{T}, \tag{11}$$

$$(t + T + 1)\, y^t_{ik} - T \leq t_{ik} + w_{ik} \quad \forall k \in A,\ i \in \mathcal{T},\ t \in \mathbb{T}, \tag{12}$$

$$\sum_{i \in \mathcal{T}} y^t_{ik} = 1 \quad \forall k \in A,\ t \in \mathbb{T}, \tag{13}$$

$$\sum_{t} y^t_{ik} \leq T y_{ik} \quad \forall k \in A, i \in \mathcal{T}, \tag{14}$$

$$y^t_{ik} \in \{0, 1\} \quad \forall k \in A, i \in \mathcal{T},\ t \in \mathbb{T}. \tag{15}$$

2) DISTANCE FORMULATIONS

We propose two alternative formulations to define real-valued variables d_{kl}^t that represent the *distance* between two agents k and l at time t, One formulation relies on user-supplied Cartesian positions of each task, that are used to approximate d_{kl}^t to the Euclidean distance between k and l. The other formulation assumes that distances between each pair of tasks are given as input parameters, without any assumptions or restrictions on how these distances are defined.

The two formulations differ significantly in the number of additional constraints that they involve. The number of constraints has a direct impact on their computational complexity as it will be shown later. Also note that the first formulation assumes that tasks are embedded in an Euclidean space while the second formulation relaxes this assumption.

a: EUCLIDEAN-BASED FORMULATION

In the Euclidean-based formulation, each task i is associated to a real-valued vector parameter $\mathbf{o}_i = [o_{i1}\ o_{i2}\ o_{i3}]$ that defines its 3D position in the Euclidean space. Using the Euclidean positions, variables d_{kl}^t are defined as an approximation of the Euclidean distance between the locations of the tasks that agents k and l are assigned to at time t.

The Euclidean-based formulation requires an additional set of helper vector variables $\mathbf{p}_k^t \in \mathbb{R}^3$, where $\mathbf{p}_k^t = \mathbf{o}_i$ iff node k is assigned to task i at time t. Variables $\mathbf{p}_k^t$ are defined by the linear constraints

$$\mathbf{p}_k^t = y_{ik}^t \mathbf{o}_i \quad \forall k \in A,\ i \in \mathcal{T},\ t \in \mathbb{T}. \tag{16}$$

Using variables $\mathbf{p}^t$, variables d_{kl}^t are defined:

$$d_{kl}^t = \left\| \mathbf{p}_k^t - \mathbf{p}_l^t \right\| \quad \forall k, l \in A,\ t \in \mathbb{T}. \tag{17}$$

Different methods can be used to *linearize* (17). For instance, [1] considers the use of a collection of tangential planes. Here we follow the method that we have developed in [18], where we use *linear regression* based on linear least-squares fitting. Following this method, we are able to model (17) using a number of linear constraints of order $|A|^2|\mathbb{T}|$.

b: PARAMETER-BASED FORMULATION

In the parameter-based formulation, the user inputs the distance between each pair of tasks using a real-valued parameters ψ_{ij} that indicate the distance between the location of each pair of tasks $i, j \in \mathcal{T}$. The formulation of d_{kl}^t is obtained using the following linear constraints:

$$\begin{aligned} \psi_{ij}\left(y_{ik}^t + y_{jl}^t\right) - \psi_{ij} \le d_{kl}^t \quad & \forall k, l \in A, \\ & i, j \in \mathcal{T}, t \in \mathbb{T}, \qquad (18) \\ 0 \le d_{kl}^t \quad & \forall k, l \in A,\ t \in \mathbb{T}. \qquad (19) \end{aligned}$$

Note that the parameter-based formulation involves a number of constraints of order $|A|^2|\mathcal{T}|^2|\mathbb{T}|$, that is considerably larger than the number of constraints induced by the Euclidean-based. Thus the increase of flexibility of the parameter-based formulation comes at the cost of significant computational overhead.

3) FORMULATION OF DIRECTIVES

Now that we have introduced variables to time index agent steps and to express distances, we can proceed to formulate the linear constrains that include spatio-temporal directives in the basic MILP. We need first to represent the distance variables $\Theta_{A_1A_2}^t$ and $\Psi_{A_1A_2}^t$, and then we need to set binary indicator variables to ensure the satisfaction of the desired bounds on the distances.

Variables $\Theta_{A_1A_2}^t$ and $\Psi_{A_1A_2}^t$ are defined using the following constraints:

$$\Theta_{A_1A_2}^t = \min_{k \in A_1, l \in A_2, k \neq l} d_{kl}^t \quad \forall t \in \mathbb{T}, \tag{20}$$

$$\Psi_{A_1A_2}^t = \max_{k \in A_1, l \in A_2, k \neq l} d_{kl}^t \quad \forall t \in \mathbb{T}. \tag{21}$$

Constraints (20-21), although not linear, can be easily linearized using auxiliary variables and linear constraints. Due to space limitations, these linear constraints are omitted and we refer the interested reader to [20].

The binary indicator variable associated to a spatio-temporal directive β with bound $\bar{d}$ is denoted with n_β. The variable takes value 1 if the directive β is satisfied in the current solution, and 0 otherwise.

We can now formulate the different types of directives using the introduced variables and differentiating among the directives based on the use of the distance variables.

The case of *coalition* directives is modeled through the following linear constraint:

$$\Psi_{A_1A_2}^t \le \mathcal{D}(1 - n_\beta) + \bar{d} \quad \forall \beta \in \beta_c, \tag{22}$$

where $\mathcal{D}$ is a large constant (e.g., maximum possible distance between any pair of agents) and $\bar{d}$ is the bound on the distance that is considered in the directive.

Similar constraints are used for other types of directives. For *network* directives:

$$\Theta_{A_1A_2}^t \le \mathcal{D}\left(1 - n_\beta\right) + \bar{d} \quad \forall \beta \in \beta_n; \tag{23}$$

for *interference-avoidance* directives:

$$-\mathcal{D}\left(1 - n_\beta\right) + \bar{d} \le \Theta_{A_1A_2}^t \quad \forall \beta \in \beta_i; \tag{24}$$

and *sparsity* directives

$$-\mathcal{D}\left(1 - n_\beta\right) + \bar{d} \le \Psi_{A_1A_2}^t \quad \forall \beta \in \beta_s. \tag{25}$$

4) USING THE DIRECTIVES

Using variables n_β, directives can be included in the form of *hard constraints*, such that a mission plan is feasible only if it is compliant with the directives, and a solution approach must exclude any plan that violates them. Alternatively, directives can be treated as *soft constraints* adding a penalty (or a reward) in the objective to avoid (or promote) plans that violate (or comply) with the directives.

One advantage of using hard constraints is that in some cases they can speed-up finding a solution since the size of the search space gets reduced. However, they can also lead to infeasible problems, or making the search for a feasible solution more challenging. Instead, when soft constraints are used, it is relatively easy to provide an initial feasible solution, and use this solution to find other –better– feasible solutions. Yet, the objective function of the MILP can become complex, and may require normalization of some terms (i.e., mission utility) and the definition of relative weights among the soft constraints [20].

VII. SOLUTION OF STASP-HMR: A GA MATHEURISTIC

In Section IV we have formally shown that, even in its basic form, the STASP-HMR is NP-hard. This poses inherent challenges in practical settings. Furthermore, in typical applications of the STASP-HMR, the time available for computing and issuing a mission plan is usually quite strict. Online replanning might also be needed during the actual development of the mission, in the face of unexpected events and issues. All these factors ask for an effective and scalable methodology for tackling the solution of STASP-HMR instances.

To cope with these challenges, in this section we propose a heuristic solution method based on a genetic algorithm embedding mathematical programming methods. Our algorithm is a combination of heuristic and MILP solvers, and it is categorized as a *matheuristic* in the literature [35]. To the best of our knowledge, our work presents the first application of matheuristics to multi-robot missions.

Genetic algorithms (GAs) are efficient stochastic search methods that simulate the adaptive evolution process of natural systems [24]. A GA starts from a *population* of individuals which encode feasible solutions of the problem. Each individual is associated to a *fitness value* that indicates its goodness compared to the rest of the solutions. From the initial population, a GA goes through a process of *evaluation, selection, crossover, mutation* and *replacement* leading to the next generation of individuals. The process is repeated for a number of generations during which the best features of parents are passed on to their offspring and thus individuals of progressively better quality can be obtained. In the following, we describe the design and the implementation of a GA matheuristic for the STASP-HMR, showing how the GA is designed and how the MILP formulation is used to solve specific sub-problems inside the genetic operators.

A. SOLUTION REPRESENTATION

In a GA, each individual solution is associated to a *chromosome*, the numeric representation of a set of features (genes) defining the individual (the *genotype*). A good encoding is fundamental in a GA in order to efficiently transmit genetic information from parents to offspring individuals. The more efficient the encoding, the better the performance.

One distinctive feature of our proposed GA is that it decomposes the STASP-HMR into two sub-problems: one of deciding the sequences of tasks in a mission plan (i.e., allocation and routing), and one of deciding how the time budget is spent among the selected tasks in the sequences (i.e., scheduling). Specifically, each member g of the population consists of a set of elementary paths in G (one for each agent). That is, $g = \{p_k \mid k \in A\}$. The genotype g can then be mapped onto a complete feasible (but not necessarily optimal) solution of the STASP-HMR by defining the amount of service time each task in the sequences receive, as long as the total time budget is being respected.

An assessment of the quality (fitness) of g requires decoding the genotype. In other words, for each g that is generated, we need to determine schedules s_k to decode the genotype and map it to a complete mission plan (i.e., a *phenotype*).

B. FITNESS EVALUATION

At the core of each GA is the definition of the *fitness* function, that quantifies how good or bad is a candidate solution in the population. In our case, the fitness of a genotype g corresponds to the maximal cumulative reward that can be obtained given that the agents follow the sequences of tasks specified by g. However, in order to compute this value we need to find the set of schedules $\{s_k|\ k \in A\}$ that maximizes the objective function of the STASP-HMR with the condition $s_k(i) > 0$ for $i \in a_k$ (tasks belonging to the paths p_k), and $s_k(i) = 0$ for $i \notin a_k$.

This is equivalent to the problem of finding an optimal service assignment for the fixed set of tasks a_k, which is an optimization problem that can be conveniently formulated as a restricted version of the STASP-HMR as follows:

$$
\begin{aligned}
\text{maximize}\quad & \sum_{i\in\mathcal{T}} R_i\Phi_i && (26)\\
\text{subject to}\quad & w_{ik} \geq 1 \forall k \in A,\ i \in a_k && (27)\\
& \sum_{k\in A} w_{ik} = T \quad \forall k \in A,\ i \in a_k && (28)\\
& w_{ik} = 0 \quad \forall k \in A,\ i \notin a_k && (29)\\
& \Phi_i \leq \sum_{k\in A} \varphi_k(i) w_{ik} \forall i \in \mathcal{T} && (30)\\
& 0 \leq \Phi_i \leq C_m(i) \quad \forall i \in \mathcal{T} && (31)\\
& w_{ik} \in \mathbb{N} \quad \forall k \in A,\ i \in \mathcal{T} && (32)
\end{aligned}
$$

Constraints (27) impose the condition $s_k(i) > 0$ for all $i \in a_k$, i.e., all tasks in the path must receive a minimal amount of service. For clarity reasons we consider all tasks $\mathcal{T}$ in the formulation, and enforce the condition that any task not belonging to any path is excluded from a solution by constraints (29). The remaining constraints are the same in the formulation for the STASP-HMR introduced in (V).

Using a MILP for fitness evaluation is convenient, not only because it can be solved using an out-of-the-shelf solver, but also because other model extensions such as the spatio-temporal directives proposed in Section VI can be considered here. The latter can be achieved by including additional linear constraints in the above MILP formulation. Note that this MILP is much smaller than the original one and, if rushing for time, the anytime capabilities of the solver can be exploited to solve the model up to a given guaranteed approximation based on the MIP gap value. This is a clear example of the versatility and the convenience of using the MILP framework to formulate the STASP-HMR.

C. GENETIC OPERATORS

The design of the genetic operators, namely the *mutation and crossover operators*, usually needs to be problem- and representation-specific to obtain an effective result. Here we propose genetic operators that are designed to achieve a good trade-off between exploration and exploitation. The mutation operator induces small perturbations to existing individuals, whereas the crossover operator takes two individuals (i.e., the parents) and *mixes* their paths to obtain a new individual.

1) MUTATION OPERATOR: BEST-SUBSEQUENCE

The mutation operator, indicated with *best-subsequence mutation*, aims at finding a better solution by removing or adding only a few tasks, without altering the order of the original ones. Intuitively, we aim at replacing each path p_k of g with another path of length greater than or equal to $|p_k|$ that includes a subsequence of p_k in a way to obtain an individual with better fitness. This is a form of *local search* since it is locally improving a given solution.

The best-subsequence operator is implemented using a variation of the MILP introduced in Section V with some restrictions on the use of certain arcs of G. Specifically, we include the following constraints:

$$\sum_{(i,j)\in E_k^1} x_{ijk} \leq M_1 \quad \forall k \in A \tag{33}$$

$$\sum_{(i,j)\in E\setminus p_k \cup E_k^1} x_{ijk} \leq M_2 \quad \forall k \in A \tag{34}$$

where E_k^1 is the set of arcs $(i,\ j)$ where either i or j, but not both, belong to p_k, and $M_1, M_2 \geq 0$ are user-defined parameters of the operator. Note that the greater the values of M_1 and M_2, the larger the deviations we can obtain through the operator.

The resulting MILP formulation of the best-subsequence mutation is composed of (1)-(10), (33)-(34). Similarly to the fitness evaluation, when the spatio-temporal directives proposed in Section VI are used, the corresponding linear constraints must also be included into the MILP.

The best-subsequence mutation can be computationally intensive, so its use must be regulated. We leverage the anytime property of the MILP solver and set a maximum computation time devoted to solving each MILP during mutation.

Lastly, we note that there exists a tradeoff between the values of M_1 and M_2 and the computational cost and quality of the mutations. Larger values can lead to better mutations, but increase the search space. When the corresponding MILP becomes too complex, the operator may produce low quality solutions given the limited computation time. Empirically we observed that $M_1 = 2, M_2 = 1$ gives a good performance.

2) CROSSOVER OPERATOR: MIXED-SUBSEQUENCES

Given two parent solutions, $g_d = \{p_k^d \mid k \in A\}$ and $g_m = \{p_k^m \mid k \in A\}$, the crossover operator (named *mixed-subsequences* crossover) works on a constrained version of the STASP-HMR in which every path p_k in the new offspring is composed by subsequences of the paths p_k^d and p_k^m. The genome of the offspring is determined by formulating and solving the MILP of Section V with the addition of the linear constraints below that restrict the paths to those that only contain arcs that are in parents' paths, g_d and g_m:

$$x_{ijk} \leq \begin{cases} 0 & \text{if } (i,j) \notin p_k^d \cup p_k^m \\ 1 & \text{otherwise} \end{cases} \quad \forall k \in A,\ (i,j) \in E \tag{35}$$

The offspring's genome is the one that satisfies these constraints and has the highest fitness.

Similar to the mutation operation, the time devoted to solve each MILP during crossover is limited by a certain threshold, after which the solver returns the best solution found.

VIII. SOLUTION OF STASP-HMR: AN ANYTIME BOUNDED OPTIMAL ALGORITHM BASED ON A SHARED INCUMBENT ENVIRONMENT

The GA-based matheuristic introduced in the previous section, being a heuristic, it is not (necessarily) optimal and does not provide any formal guarantees in the degree of optimality of the solution found. Nevertheless, it produces high-quality solutions in a very computationally-efficient manner and with low computational demands, specially in terms of memory (see Section XI). In other words, it is an effective and efficient approach to the heuristic solution of the STASP-HMR.

Yet in some scenarios it is crucial to understand how good the proposed solution is, for instance, by means of bounded optimal solutions. When computational resources for the solver are readily available (e.g., using dedicated cloud servers), effective bounded optimal solution approaches can be useful. In this section we build on and complement the previous approach with a novel *exact* method based on the combination of the GA-based matheuristic and a generic MILP solver for the STASP-HMR. The method is complete and yields bounded optimal solutions in an *anytime* fashion.

The solution approach consists in setting up a cooperative environment, namely a *shared incumbent environment* (SIE), that runs the GA-matheuristic proposed in the previous section and a generic MILP solver over the same problem, in parallel, interacting and continuously exchanging relevant information. Our implementation of the SIE method runs multiple instances of a generic MILP solver and the GA matheuristic. These instances run simultaneously, in parallel, solving the same problem instance. The SIE shares the best solutions found among all the solvers.

The objective of the SIE is to help each of the running solvers in overcoming their weaknesses, resulting in a speed-up of the overall solving process. The solution that is ultimately obtained is retrieved from the instance of the

MILP solver, together with its MIP gap. As a result, the SIE provides a solution with *formal guarantees in terms of the MIP gap and, eventually, the optimal solution*. The fundamental role of the SIE is to facilitate the exchange of solutions between the solvers.

The SIE inherits all the properties of the solvers from which it is composed. Since we are using an exact MILP solver within the SIE, the proposed SIE becomes an *exact solution method* that preserves the formal guarantees and anytime properties of exact solvers and, at the same time, it is also computationally-efficient.

Our implementation of the SIE consists of two independent processes – one running a generic MILP solver and the other one the GA-based matheuristic – over the same problem instance. These processes *communicate* with each other using mechanisms for inter-process communication. The cooperation scheme between both solution approaches consists in the continuous exchange of their best found solutions so far. From the MILP solver perspective, best solutions correspond to the best upper bounds – also known as the *incumbent solutions* – which are readily available during the branch-and-bound search. In the GA-based matheuristic, they represent the best individuals found during evolution.

At the MILP solver side, external feasible solutions (e.g., in the SIE, those received from the GA-matheuristic) are injected to the solver using the API. At the GA-matheuristic side, new solutions are introduced into the current population at each generation.

A. IMPLEMENTATION DETAILS

The SIE is implemented using Linux sockets for inter-process communication. The exchanged solutions consist of a list of tuples (*variable*, *value*), for each variable used in the MILP formulation. In the MILP solver, this list can be retrieved using the application programming interface (API) of the solver. In the specific case of the CPLEX software that was used, system callbacks are used to retrieve this list every time a new incumbent solution is found.

When a solution is received from the matheuristic, it is passed to the MILP-search using the API. With CPLEX, user-written callbacks are used to inject integer-feasible solutions during the branch-and-cut search. Such callback functions are called frequently during the search by CPLEX.

At the matheuristic side, all solutions received from the MILP solver are temporarily stored inside a FIFO queue. We explored two different ways to introduce these solutions into the current population: (a) through the genetic operators, and (b) replacing certain individuals after each generation. After experimenting with these two strategies, we noticed that forcefully replacing the worst genotypes with the exchanged solutions provided the best performance in terms of solution quality. We also discovered that it is necessary to limit the percentage of genotypes of the same generation that can be replaced to preserve population diversity. We empirically determined that replacing up to 10% provides a good performance.

In Section XI-B we demonstrate through extensive experiments the effectiveness of the SIE and show that a significant performance improvement can be obtained when the matheuristic and the generic MILP solver are combined, compared to their use as independent solvers.

IX. ONLINE ITERATIVE REPLANNING

Despite the efficient solution approaches proposed in the previous sections, two main challenges remain regarding the practical applications of the STASP-HMR. First, relatively large instances of the STASP-HMR present inherent computational challenges. In these cases, finding good approximate solutions may require a relatively long computation time (e.g., order of hours or even more). Secondly, although long computation times may still be affordable in some scenarios (e.g., when plans can be safely executed in *open-loop* modality), other scenarios need mission planning to be performed iteratively in *closed-loop with mission enrollment*. In these scenarios, which are indeed the most common in practical applications, a *replanning* scheme must be in place to trigger the computation of a new mission when facing new evidence or to accommodate unexpected contingencies during execution. For instance, during a search a rescue mission, a number of unforeseen events can naturally happen during mission execution, causing delays, deviations, and, in general, requiring to change or adapt the original plan on the spot.

We propose to tackle closed-loop scenarios by defining both a *centralized architecture* and a *decentralized architecture* for mission planning and control. In this section we describe the case of using a centralized architecture. In the next section, adopting a top-down design, the centralized architecture is used to derive a decentralized scheme, which brings additional advantages regarding computational requirements and resiliency to failures.

When using a *centralized architecture* for closed-loop mission enrollment, the centralized mission planner begins by computing a mission plan for all agents. Upon receiving the plans (e.g., through a wireless communication channel), the agents start the mission. During execution, mission updates are continually sent from the agents to the planner. In turn, either periodically or based on the received updates, the centralized planner triggers the computation of a new plan, which is sent out to the agents on the field. The process is iterated until the time budget for the mission is used fully.

In practice, the online iterative replanning is implemented as a *multi-stage, periodic, re-optimization procedure* [10] that solves a sequence of static problem instances over time with a varying horizon length. At any single planning stage, a horizon T_H defines how far to look ahead during planning. Longer horizons are expected to enable a better allocation of resources, but also increase the computational complexity as the size of the associated problems increase. In contrast, shorter horizons tend to be computationally affordable, but also to make agents to reason myopically resulting into lower-quality decisions. The chosen value T_H is used as the T parameter in the MILP associated to the current stage.

The described scheme is rather straightforward. In order to address the practical issue of long computation times for large instances, we exploit the anytime capability of our exact approach: a time limit t_{plan} is imposed for solving each staged instance of the MILP. t_{plan} is a strategically defined parameter: the larger the better, but at the same time it has to comply with mission's time constraints. In general, T_H and t_{plan} are strategic parameters that can affect both quality and feasibility of the mission.

The impact of and interplay between T_H and t_{plan} is studied in Section XI-E. The results show that in practice an online replanning with reasonably good performance is obtained with a centralized approach when the agent team is relatively small. However, the results also indicate that the scalability to large teams is still an issue. This justifies the need for the decentralized architecture described in the next section.

Another parameter that has an impact on mission quality and computational load is the *frequency of replanning*, that can follow a periodic scheme or can be adaptive. Without losing generality, we assume that the time between two consecutive replanning stages is randomly chosen from an interval $[t_a, t_b]$ with $1 \leq t_a \leq t_b \leq T_H - t_{plan}$. When $t_a = 1$, replanning must occur just after the agents execute the first action of the last plan computed. When $t_b = T_H - t_{plan}$, the next replanning stage can occur at latest just before the current mission plans expire (i.e., while agents execute their last task assigned). In Section XI-G the choice of $[t_a, t_b]$ and its impact on performance is investigated in a number of simulation experiments. We also refer the interested reader to [17] for additional discussions about the choice of horizon and replanning interval.

X. A DECENTRALIZED IMPLEMENTATION USING ITERATIVE REPLANNING AND IMPLICIT COORDINATION

In many applications of interest, a centralized architecture will present many shortcomings and will not be scalable. First, solving the MILP for the entire team might take too long to be used for a continual online replanning. Second, reaching out all the agents to / from the control center requires a reliable, high-bandwidth network infrastructure that not necessarily might be in place (e.g., after a disaster, or when performing the mission in remote areas). Third, the use of a control center brings an inherent weakness in terms of fault tolerance.

To overcome the issues related to a centralized architecture, we propose a fully *distributed and decentralized* architecture derived in a top-down modality from the centralized one. Each agent runs a replica of the mathematical model and use it to plan its own actions in closed-loop modality, based upon the knowledge acquired from information sharing with the other agents during the course of the mission. The designed decentralized architecture is based on the principle of *implicit coordination*: while deciding their own actions, independently, robots implicitly coordinate. Implicit coordination is regarded as a practical, flexible, and scalable way to tackle multi-agent missions [28]. The adopted MILP formalism provides an immediate way to decentralize the problem and the solution approach based on implicit coordination.

In the following two sub-sections we discuss how the online iterative replanning introduced in the previous section is implemented in a distributed and decentralized architecture. The implementation leverages the use of local agent data and the MILP formulation to compute plans that can implicitly coordinate the agents. We then discuss what and how *data are exchanged among the agents* in order to maximize their local perception of the mission. Henceforth, we assume that agents are equipped with a range-limited wireless interface that they can use to communicate with each other using a *mobile ad hoc network*. In other words, we do not assume the presence of a global network infrastructure and we emphasize some of the challenges that come with the lack of a global infrastructure, e.g., how the robots coordinate their plans with each other when a global network infrastructure is not available.

A. SHARED KNOWLEDGE REPRESENTATION

In addition to the static information about the scenario – which can be initially given to the agents – two types of data need to be revised and continuously updated to let each agent working in closed-loop interaction with mission enrollment and, therefore, perform online, iterative replanning. First, a local estimate of the completion map C in needed to track the amount of service each task has received and still requires. Second, each agent should be as aware as possible of the scheduled future actions (i.e., current plans) of other agents, in order to avoid overlapping actions.

To this end, agents share information about their completion map and their plans by means of *incremental update messages* that are broadcast. These messages, once received and added up all together, let other agents to maintain an up-to-date local view of the global status of the mission. In the case of the completion map, an incremental update message consists of a tuple $C^u =< k,\ i,\ t_{start}, t_{end}\ >$ that indicates that agent k has provided service to task i from time t_{start} to t_{end}. The updates are issued and sent out in the wireless network by the agent just after servicing a task. Once other agent receives a C^u, it uses it to update its local C by subtracting from it the amount of service provided by k to i based on the knowledge of k's efficiency function: $C(i) := C(i) - \varphi_k(i)(t_{end} - t_{start})/\Delta_T$.

Regarding the plans, a similar approach is followed. A plan update is a tuple $< k, t_{gen}, i, t_{start}, t_{end}, >$ that represents the intention of agent k to perform task i from t_{start} to t_{end}. Note that in this case a timestamp t_{gen} is added to indicate the time at which k expressed such intention. At any time instant t, the plan of an agent k can be (partially) reconstructed by considering all the received plan updates with $t \leq t_{start}$. The values of t_{gen} allow to resolve conflicts when two or more updates overlap. As for the completion map updates, each plan update is uniquely identified by a sequence number and stored in a local table.

During the mission, agents periodically broadcast network discovery messages to announce their presence to other agents. Using these messages, each agent maintains a list of agents from which it has received a message in a recent time period, henceforth called the agent's *neighborhood*. With certain frequency, the agent randomly chooses another agent in its neighborhood to initiate a synchronization their lists of mission updates. The synchronization procedure allows agents to merge their local knowledge and, in overall, improve their local perception of the mission.

B. ITERATIVE PLAN UPDATING USING THE MILP

In the distributed architecture, mission planning is performed iteratively, where each agent follows the scheme presented in Section IX in an *asynchronous* fashion. To do so, at each planning stage, the agent sets up a MILP using the locally estimated completion map, and current (partial) plans of other agents. This makes the MILP much smaller than in the centralized case. In practice, *only neighborhood-level MILPs are included and solved*.

The inclusion of the local completion map C in constraints (8) is straightforward whereas the *plan updates received from other agents* are handled in a more strategic way. It must be noted that, since data exchange can be sparse, an agent may become unaware of the activities of some members of the team. Therefore, it does not make sense to consider teammates for which no reliable information about location and current intentions are known. Instead, we let each agent only considering teammates from which it has got an information update recently, e.g., at least one update since the last replanning stage. Other agents are omitted from current MILP until they reappear in its neighborhood.

Furthermore, it is often the case where an agent has only *partial information* about plans of others. This is due to the fact that, as explained above, plans are computed up to a certain (rolling) time horizon T_H, such that the current plan of some of the neighbors may be due before the T_H being considered by the agent at any given planning stage. In these cases, the agent must either infer the incomplete part of the plan or let the solver optimizing it. In this work, we adopt the former strategy to reduce the search space and consequently the computational cost of planning. To this end, partial plans are completed using a *greedy heuristic* procedure as follows. In sequence, for each agent in the neighborhood, the planner performs a greedy assignment of tasks with a time budget of T_H: the next assigned task is the one that locally maximizes the neighbor's reward. After the greedy completion procedure, the plans of all neighbors are complete and used by the planner to fix constant values the variables t_{ik}, w_{ik}, and x_{ijk} for all k in the set of neighbors. As a result, these variables are no longer decision variables.

Restricting to the current neighborhood, and inferring and fixing variables corresponding to neighbor agents, drastically reduce the total number of decision variables of the MILP to those that only define the actions of the agent doing planning. In our tests (see Section XI-G), we noted that this leads to substantial computational savings and better performance. In fact, the gain that would potentially be obtained by solving the entire problem is diminished by the relative poor quality of the solutions that can be computed in the limited time budget allocated for planning.

XI. COMPUTATIONAL RESULTS

In this section we provide an experimental comparative evaluation of the different solution methods of the STASP-HMR, as well as an analysis of its centralized and decentralized implementations.

We start by defining a *benchmark* set that is generated to be representative of real-world instances of STASP-HMR. The relative performance of the solution methods is evaluated on the benchmark based on the *run-time distribution* (RTD) methodology, that serves to characterize the anytime behavior of stochastic search algorithms for combinatorial optimization [29]. Instead, for the decentralized implementation we quantify the loss with respect to the equivalent centralized implementation and we study the impact of different design parameters. All the numerical results were obtained with an AMD Opteron® Processor (2.0 Ghz) and the algorithms are implemented in C++.[4]

A. PROBLEM INSTANCES

The benchmark set consists of random problem instances where the locations of the tasks composing the mission are embedded in a 2D-grid, with a one to one correspondence between tasks and grid cells (e.g., see Fig. 1). We consider square grids of $L \times L$ cells, with $L \in \{5, 6, 8, 10, 20\}$. We adopt a dimensionless space, yet we consider that robots must move through adjacent cells: those that share at least one vertex on the grid.

Each instance includes from 4 up to 20 agents at random initial locations, and up to four different classes. Each agent class is characterized by the performance in accomplishing the tasks. Specifically, each instance defines four agent classes, each one with a randomly chosen performance function φ with five possible performance levels for each task: ranging from inefficient ($\varphi \triangleq 6.25\%$) up to highly efficient ($\varphi \triangleq 100\%$). This means that, depending on its class, a robot may require from 1 mission step up to 16 mission steps to complete a specific task. The total number of problem instances in the benchmark set is 3,060.

B. PERFORMANCE COMPARISON OF THE PROPOSED SOLUTION APPROACHES

We first compare the different solution approaches proposed in this work, namely a generic MILP solver, the GA-based matheuristic, and SIE. We use CPLEX® as generic MILP solver. We impose a *time limit* of one hour and a *memory limit* of 2 GB to optimize a mission plan for each of the instances using different planning horizons ($4 \leq T \leq 20$). Upon reaching the limits of these computational resources, all three

[4]Code and problem instances are available at https://github.com/EduardoFF/STASP-HMR

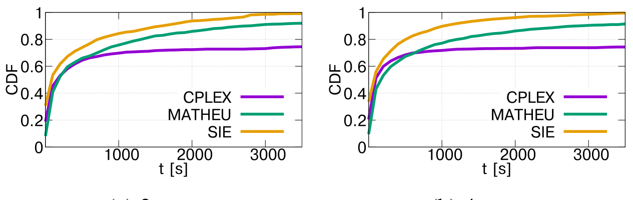
methods return the best solution found so far. Furthermore, during the computation of a solution we keep a record of the quality of the best solution found in terms of mission utility. When using SIE and CPLEX, we also retrieve the MIP gaps over time, that quantify the estimated distance of the current solution from the optimum. To account for the use of parallel computing, we run each method using 2 and 4 cores. In the SIE the number of cores is uniformly distributed over the CPLEX and the GA-matheuristic.

The GA is initialized with a randomly-generated population. The GA is a steady-state GA where only a percentage of the current population (5%) is replaced by the best solutions in the offspring. A population size of 200 is used. A tournament method is used for selecting the individuals with best fitness from the population using a linear scaling scheme. The selected genomes are used to generate new offspring by the crossover operation. The crossover probability is set to 0.9 and the mutation probability to 0.1.

We compare the algorithms using an RTD analysis as follows. Each solution method is used to solve all the benchmark instances. From the solutions obtained for the same instance, we take the best among them and refer to it as the *target solution*. To visualize and compare the anytime behavior of the solution methods, for each method we plot the RTD where the x-axis represents the run-time (in seconds), and the y-axis represents the proportion of the runs (normalized between 0 and 1) that provided a solution whose quality is equal to the target solution over all the benchmark. As a result, the RTD provides the empirical cumulative probability distribution of finding the best solution within a time bound for the algorithm.

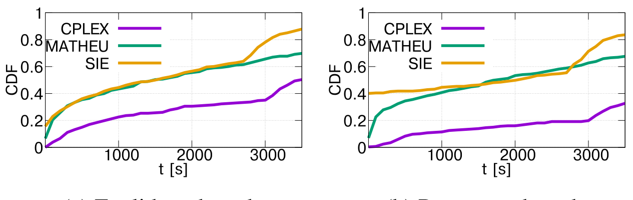
Numerical results are shown in Fig. 5. We note that SIE is by far the best performing algorithm, with the GA matheuristic being close in performance. Moreover, they both show a nice anytime growth in performance, while CPLEX alone shows a stagnation after a short while. The difference in performance among the algorithms is even more evident when using 4 cores. These results suggest that SIE is well able to exploit the synergies between the CPLEX solver and the GA matheuristic. We also note that, although we impose a memory limit of 2 GB, the GA matheuristic required less than 10 MB during the entire execution, while SIE exhausted the memory available in some of the instances.

We also consider problem instances that use spatio-temporal directives. To this end, we randomly generated sets of directives to be included as soft constraints. Furthermore, we consider separately each of the formulations of distance variables presented in Section VI-B2. In the benchmark that uses the spatio-temporal directives, the results are different, as shown in Fig. 6. This is due to the fact that the corresponding MILPs have also different (higher) complexity. In this case both SIE and the GA-based matheuristic have a similar behavior and significantly outperform CPLEX. The performance gap is more evident when using the parameter-based formulation, showing the different complexity incurred by the formulations.



(a) 2 cores (b) 4 cores

FIGURE 5. Run-time distribution on hard instances for GA-based matheuristic, CPLEX, and SIE using 2 and 4 cores. The x-axis represents the run-time (in seconds), and the y-axis represents the proportion of the runs (normalized between 0 and 1) that provided a solution whose quality is equal to the best solution found.



(a) Euclidean-based (b) Parameter-based

FIGURE 6. Comparison between two different MILP formulations of distance variables using problem instances with spatio-temporal directives. Each instance is solved using CPLEX, the GA-based matheuristic (MATHEU), and the SIE. The RTD shows the performance of the solvers using each formulation.

C. THE EFFECT OF NON-ATOMICITY AND PARTIAL FULFILLMENT OF TASKS

One aspect that differentiates our planning approach from most of previous work is that it accommodates plans described in terms of non-atomic tasks. This aspect is particularly relevant when a limited time budget is given. By considering non-atomic tasks, a valid solution may (i) leave some tasks *partially fulfilled* by the end of the planning horizon, and (ii) allow different agents to work on the same tasks over disjoint time intervals. These two aspects make the solution space larger than the case of atomic tasks, which in turn could produce more higher quality solutions.

We use the model and its MILP formulation to analyze the advantages of these two mechanisms. At this aim, leveraging the flexibility of the MILP formulation, we define two variants of the STASP-HMR. The first variant STASP-HMR-AT, that was already introduced in Section IV, enforces the atomicity of tasks. The second variant of the model (denoted by STASP-HMR-FF) relaxes the atomicity requirements but imposes the restriction that mission plans must fulfill the service requirements of any tasks included in the solution. In other words, if a task i is included in the mission plan, then i must be brought to completion at the end of the mission plan by the work of one or more agents. Both variants are formulated by means of additional constraints in the MILP.

Note that the atomicity condition also implies that the involved tasks are completely fulfilled. If we characterize the solution space of any given problem instance of the different variants of the STASP-HMR, the following relation holds:

$$\text{STASP-HMR-AT} \subseteq \text{STASP-HMR-FF} \subseteq \text{STASP-HMR}.$$

Fig. 7 shows the performance, in terms of mission completion, of optimal mission plans derived from the STASP-HMR, and each one of its variants, considering different number of agents and mission time spans. We consider a total of 100 problem instances in 7×7 grids. For all instances, we were able to obtain optimal solutions within one hour time.

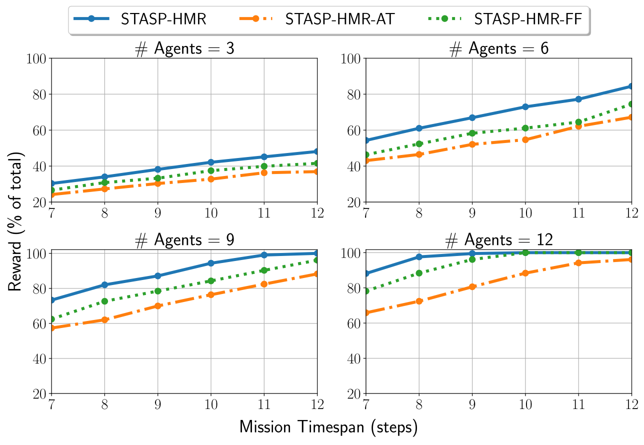


FIGURE 7. Effect of non-atomicity and partial fulfillment of tasks.

In all the scenarios the performance of plans using non-atomic tasks is significantly higher than the performance of those with atomic tasks. This suggests that the partial fulfillment condition enables the agents to better distribute their effort over a larger number of tasks, and as consequence, to achieve higher performance levels. These results encourage the use of non-atomic tasks to define mission plans whenever it is possible based on the application requirements.

D. THE VALUE OF TIME-EXTENDED TASK SCHEDULING

Next, we show the advantages of time-extended planning with respect to a more reactive approach in which we optimize the assignment of tasks using a *myopic* method. The myopic method (MYOP) solves a sequence of STASP-HMR instances with one-step look-ahead.

More specifically, for a time budget T, MYOP is equivalent to solve a sequence of problem instances $\langle P_1, \ldots, P_T\rangle$, each using a time horizon of 1 step. The solution of each instance P_i consists in a mission plan $\mathcal{P}_i$ that defines a single task assignment for time step i. After computing plan $\mathcal{P}_i$, the resulting agents' positions and coverage map serve as input to the computation of plan $\mathcal{P}_{i+1}$.

In Fig. 8 we show the performance of mission plans computed over the whole mission time-span (TIME-EXTENDED) and using the MYOP approach, for different mission timespan, and team sizes. We consider a total of 100 problem instances in 7×7 grids. For all instances, we were able to obtain optimal solutions within one hour time using the SIE. We can observe that time-extended plans exhibit better performance than the MYOP counterpart, and that the performance improvement is independent of the

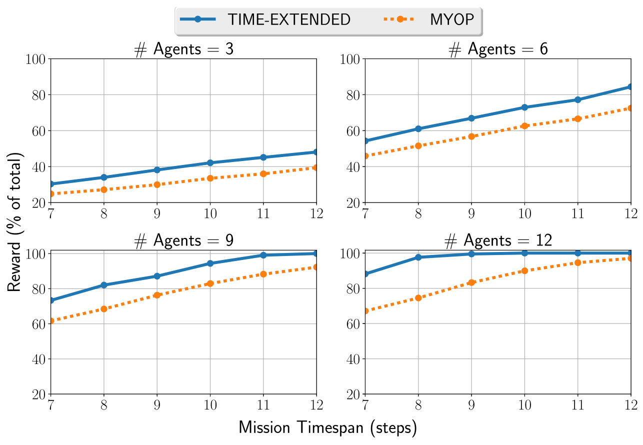


FIGURE 8. Time-extended vs. MYOP approach.

length of the mission timespan. We also notice a greater difference in performance for larger teams.

These experiments show the benefits of considering large horizons while coordinating a mission. They also illustrate how we can compute missions plans by solving a sequence of problem instances with a fixed look-ahead horizon length. We remark that the computational cost of time-extended planning increases with the length of the horizon.

E. ITERATIVE REPLANNING IN A CENTRALIZED SCHEME

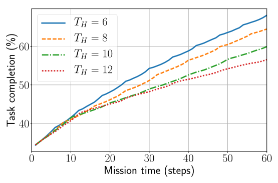
In this section we perform a computational study of the centralized implementation, focusing on the replanning strategy of Section IX. We analyze the impact on performance of time limits for the computation of mission plans, t_{plan}, and length of planning horizon T_H at each replanning stage.

We consider a mission duration of 60 time steps. Using a team of 6 agents, we use the centralized scheme, where the online iterative replanning is performed using different parameters. We consider 10 problem instances. For each mission, and at each replanning stage, we use the SIE to obtain solutions to the mathematical models associated to each stage. We restrict to t_{plan} the time allowed to the solution approach to compute a mission plan, after which the best mission plan found so far is retrieved and dispatched until a subsequent replanning is done.

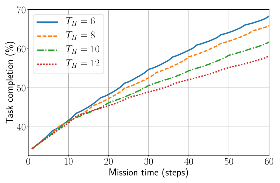
The simulation model that we use to execute the mission plans assumes that no deviations occur when agents execute their plans and a perfect communication channel between the planner and the agents. In this way, we only focus on the effect of the parameters of the replanning scheme over the performance of a mission under idealistic situations when plans are executed as expected, and perfect knowledge about the status of the mission is provided. Nonetheless, we note that in dynamic scenarios, replanning can be triggered at any time, by the user, to adapt the plans to deviation in the executions. Furthermore, different planning strategies can be used to support the communication between the planner and the agents. We refer the interested reader to [18], [20] for a comprehensive discussion about connectivity-aware mission planning.

We consider 4 values for the planning horizon $T_H \in \{6, 8, 10, 12\}$, and 3 values for the time limit (in seconds) imposed to the solver at each replanning stage $t_{plan} \in \{300, 600, 1200\}$. For each mission, we performed 5 independent runs.

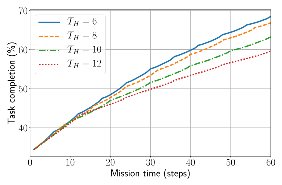
The results in Fig. 9 show the mission performance, in terms of task completion, over time. We note that longer planning horizons do not provide a better performance as expected. We also note that the performance gap increases when t_{plan} is reduced. Overall, the system seems relatively robust to the tuning of the parameters T_H and t_{plan}, with the relative differences being of order of 10-20%. However, it is evident that performance would not scale well to larger scenarios. It is in this perspective that we have developed the decentralized architecture described in Section X, which is evaluated in Section XI-G.



(a) $t_{plan} = 300$ s



(b) $t_{plan} = 600$ s



(c) $t_{plan} = 1200$ s

FIGURE 9. Mission performance using the online iterative replanning under centralized control.

F. COMPARISON WITH A STATE-OF-THE-ART APPROACH: TERCIO

In this section we evaluate the proposed anytime exact method against relevant state-of-the-art approaches. Since in this paper we are both proposing a new problem model, the STASP-HMR, and multiple solution methods for it, it was indeed challenging to find suitable competitors precisely because of the novelty of the problem.

In particular, one of the fundamental characteristics of the STASP-HMR is the non-atomicity of tasks, which requires an explicit decision on the amount of time that each agent devotes to each task. Instead, most of the existing approaches, both from the OR and multi-robot literature, are designed for atomic tasks. Therefore, to possibly make a fair comparison, instead of arbitrarily modifying existing methods (which could be an algorithmic challenge by itself), we have first converted our benchmark instances to equivalent instances with atomic tasks. In fact, our algorithms, while being designed to tackle non-atomic tasks, can work equally well with atomic tasks. The *conversion of the benchmark instances* was done as follows. Each task is split into a set of tasks, each one representing a fractional part of the original task. The split is done considering the least amount of workload an agent could do in a given time unit. Each of the resulting *fractional* subtasks are precedence-related atomic tasks with a reward equivalent to the fraction they represent.

We have looked at the literature discussed in Section II for state-of-the-art competitors for the benchmark set with atomic tasks. Given the similarity of the STASP-HMR with VRPs, and in particular with the maximization of reward collection under limited time budget in TOPs, we have first looked for competitors in this domain. Unfortunately, the methods that have been developed in the OR community to tackle these types of TOPs (always assuming atomic tasks) do not scale well to the size of our benchmark instances (once made atomic). For instance, in [27] the largest benchmark for TOP consists of up to 400 tasks, and 4 agents. Instead, after the conversion, the number of atomic tasks in our largest problem instance becomes greater than 6,000 (and we use up to 20 agents). Therefore, we omit OR methods for comparison since our benchmark instances seem not tractable for OR heuristics.

In Section II we have also remarked the relationship of the STASP-HMR with problems and methods for task allocation in multi-robot systems. However, in this domain rather than maximizing reward collection under a restricted time budget, it is more common to consider as objective the *minimization of the mission makespan*. In practice, it is tacitly assumed that the given tasks can or must be completed in the available time, such that an explicit restriction on the time is not necessary. We have already pointed out in Section III-C that the STASP-HMR can be equivalently defined for the case of makespan minimization and the MILP formulation can be immediately adapted for it, essentially by changing the objective and removing the time budget constraint.

Based on the above facts and remarks, we have selected and implemented *Tercio* [25] as a state-of-the-art baseline competitor for our proposed methods. Tercio is a recently proposed algorithm for task assignment, routing, and scheduling in multi-robot systems. It aims at minimizing the makespan, such that we had to consider the STASP-HMR with this objective and unrestricted time budget. Tercio was designed for and tested on atomic tasks, and explicitly considers the notion of precedence-related subtasks. It had been tested with instances of up to 1,000 subtasks. All these reasons make Tercio particularly suited for our evaluation.

The two solution methods, Tercio and ours, fundamentally differ in the way the problem is decomposed and in the treatment of non-atomic tasks. Tercio decomposes the problem into *allocations first*, *route and time assignment second* and does not explicitly consider non-atomic tasks. Our exact solution method, namely the anytime exact approach described in Section VIII, decomposes the problem into *allocate and sequence first*, *time assignment second* and it gives non-atomic tasks a proper treatment by explicitly splitting the workload of tasks among the agents.

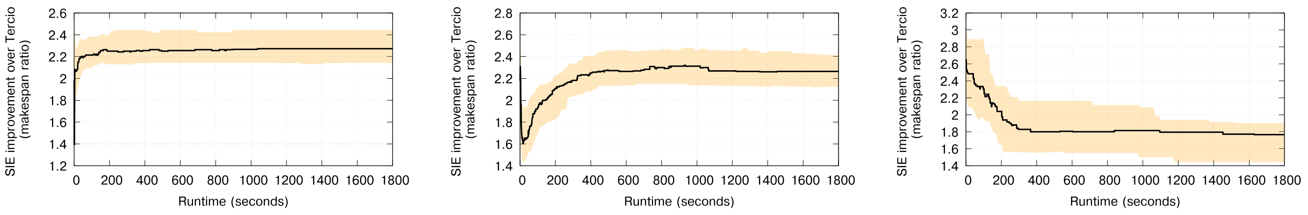


FIGURE 10. Tercio vs. our approach on instances with 5 × 5 (left), 10 × 10 (middle), and 20 × 20 (right) tasks: ratio between the makespans obtained by Tercio and the ones obtained by our solution approach over time.

In the evaluation, we consider instances of size $L \times L$, with $L \in \{5, 10, 20\}$. For each L, we consider 800 problem instances with teams from 4 up to 20 agents. Each problem instance is solved using Tercio and our approach. We ran each solution method for at most 30 minutes and record the progress of the makespan of the best solutions found. Note that both solution approaches are anytime algorithms, meaning that the solution is improved over time. In order to understand the anytime behavior of each algorithm and their relative performance, we compute the ratio between Tercio's makespan and our approach's makespan, for each instance, over the computation time. Fig. 10 shows the distribution of the ratios over time.

Results shows that, for this set of instances, the makespan of the solutions computed by Tercio after 30 minutes increases two-fold with respect to the makespan produced by the SIE. We also noted that our approach was able to find optimal solutions to 90% of the instances of size $L = 5$ and $L = 10$ within the allotted time, and within 300 seconds in most of the cases. The increasing ratios in the $L = 5$ and $L = 10$ plots suggest that the SIE progresses faster than Tercio in the first minutes of computation, and the performance of Tercio slows down over time. In the instances of size $L = 20$, which are considerably harder for both methods, we observe that the ratios tend to decrease fast in the first 200 seconds, but again remain almost constant after several minutes. These observations can be explained by the increasing complexity of computation that Tercio exhibits. Tercio iterates over feasible agent allocations, which are computed by means of a MILP model. After each iteration, a new constraint is added into the MILP to exclude the agent allocations tried previously. As a result, the MILP increases in size, slowing down the overall performance of the algorithm. This performance bottleneck is more evident in instances with a large number of (atomic) tasks, such as the ones considered here.

G. PERFORMANCE AND ANALYSIS OF THE DECENTRALIZED APPROACH

In this section we evaluate the decentralized implementation presented in Section X. We focus on the *loss* in mission performance with respect to an equivalent centralized implementation, and the *sensitivity* of the decentralized scheme on core parameters such as the replanning interval and the horizon length.

For this analysis, we consider 500 problem instances with a grid of 20 × 20, i.e., 400 tasks. The total number of robots in a mission is fixed to 12. All simulations have a duration of 48 steps. At the start of the mission, all robots are deployed at random positions inside the grid environment. They all compute an initial plan, and then execute the online replanning in an asynchronous way.

As seen in Section X-A, the decentralized scheme involves communication among the agents. In spatially distributed missions, task locality may impose limits on the communication. To evaluate the effect of constrained communication in the simulations, we let data exchange between two robots occur only within a fixed communication range, denoted as ψ_r. To this end, we embed the grid into a Cartesian plane, and define the distance between two tasks as the Euclidean norm between the center of the associated cells. Whenever two robots are within a distance less or equal to ψ_r we let the agents to exchange data and to synchronize their current knowledge using the mechanisms described in Section X-A. We consider two different classes of scenarios. In the first, agents are able to exchange data without restrictions ($\psi_r = \infty$). In the second class, robots can only exchange data with other robots located within a communication range $\psi_r = 3$. In this way we analyze the impact of incomplete information.

To measure the performance of the decentralized scheme, we compare it with a *centralized implementation* that has a full knowledge of the environment. These centralized solutions are used to derive bounds that define a baseline metric, which we use to evaluate the relative performance of the decentralized implementations with respect to the centralized one. More precisely, the centralized solution is obtained using the SIE which also provides a MIP gap. The MIP gap, in turn, defines an upper bound on the MILP's objective function. The ratio between the solution obtained by the decentralized implementation and this upper bound is then used to compute the relative gap G_{opt}^{dec}, which is reported in the plots. Note that this is a conservative estimate of the performance.

As pointed out in Section X, two of the most important parameters that affect the performance of the online replanning are the length of the look-ahead planning horizon (T_H) and the replanning frequency (specified by an interval $[t_a, t_b]$). We perform a sensitivity analysis of these parameters. We consider different replanning intervals and horizon lengths. First, we fix a relatively short interval for replanning and evaluate the effect of the increasing horizon.

Fig. 11 shows the effect of increasing the horizon from 6 up to 24 steps. We indicate the median value and 25% and 75%

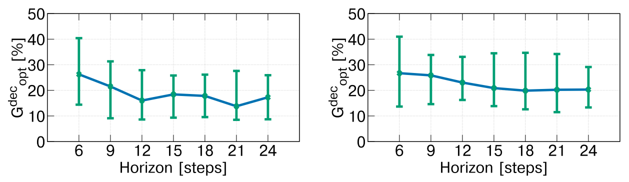


FIGURE 11. **Effect of the look-ahead planning horizon on the decentralized scheme with full (left), and constrained communication (right).**

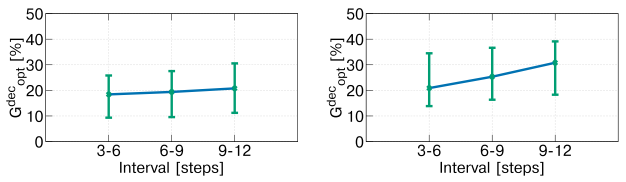


FIGURE 12. **Effect of the replanning frequency on the decentralized scheme with full (left), and constrained communication (right).**

error bars for both the communication-constrained (right) and unconstrained scenarios (left). Intuitively, one would expect that the longer is the horizon, the better are the plans that robots compute, which also results into better global mission plans. Indeed, we observe that the increasing horizon tends to improve the performance of the team. However, for longer horizons, the improvement stops and the performance stagnates. A similar observation was made in [13], although without an explanation. Regarding the relationship between these parameters and the amount of information exchanged, we note that information is better exploited with longer horizons. In the communication limited scenarios, longer horizons only provide a marginally better performance.

We conjectured that the performance stagnation effect described above is due to the fact that, for longer horizons, the corresponding MILPs become harder to solve. However, after an examination of the distribution of the MIP gap values over the increasing horizon length during the simulations ($\sim 70,000$), we noticed that the increase of problem complexity was not significant. In fact, most of the solutions remain within 2 % from the optimal solution. We conclude that this behavior is due to the asynchronicity of the online scheme and the effect of incomplete information regarding the planned actions of other agents. In other words, the whole system becomes highly dynamic and the benefits of longer planning horizons are greatly diminished by this dynamism.

Next, we evaluate the impact of the replanning frequency. We consider three replanning intervals, $[3, 6]$, $[6, 9]$, and $[9, 12]$, that are used to define the replanning frequency as described in Section IX. For a given scenario, the interval remains constant during the simulation. In the simulation, each agent randomly chooses the number of time steps between consecutive replanning stages from that interval. The goal is to analyze the case when agents perform planning in an asynchronous way, and have the freedom to replan in an adaptive way. Fig. 12 compares the performance that is obtained using the different replanning intervals. When the replanning occurs less frequently, the performance tends to deteriorate sharply when communication is limited. When communication is not an issue, the frequency of replanning is not central, but nevertheless compensates the small decrease in performance due to the inconsistent information.

XII. DEMONSTRATORS

This section describes two real-world deployments of the STASP-HMR model that we have implemented for cooperative mission planning. The first one relates to wilderness search and rescue operations (WiSAR in short) using a team of heterogeneous agents. The second one involves a team of mobile networked robots and illustrates the use of spatio-temporal constraints.

A. MISSION SUPPORT SYSTEM FOR WiSAR

In the WiSAR scenario, the location of a *stationary non-evasive target* (e.g., an injured hiker) must be identified within a relatively short time. To accomplish this mission, a group of searcher agents, each of them with possibly different sensory-motor characteristics, is ready to be deployed in the area. This scenario was implemented in the context of the Swiss-funded project SWARMIX (`www.swarmix.org`).

We can convey many different types of information into a model based on the STASP-HMR and use the resulting model to synthesize team-level search and rescue mission plans. In the WiSAR scenario, tasks consist in autonomous searching activities done by the agents inside well-defined sectors. A completed task means that an accurate search for the target has been performed in the entire task sector, returning yes/no about the presence of the target or of target's cues. The goal is to define mission plans maximizing the outcome of search activities within the time limits of the mission. A mission plan is an allocation of sectors and the definition of schedules that each agent should follow.

To address the practical challenges of a real-world mission deployment, the STASP-HMR model and the mission planning algorithms have been implemented as a part of a more comprehensive *mission support system* (MSS) for WiSAR. The system has been proposed as a practical tool that a mission commander can use to delineate mission plans and to monitor the execution of the mission [21]. Assuming the presence of an underlying communication infrastructure enabling the MSS and the WiSAR agents to continuously exchange mission-relevant information, the MSS supervises the activities of the agents and the execution of the mission plan. A graphical visualization and a simulation tool of the computed plans – both at the trajectory and exploration gain level – enable the commander to revise the current plan, trigger replanning, and dispatch new instructions to the searchers using the communication network.

In the SWARMIX project, the MSS was successfully integrated with different platforms, including aerial robots (both fixed-wing and quadcopters), an intelligent dog collar to track dogs' movement and to support the execution of directional commands, and smartphones that support human rescuers during the mission. A full demonstration was carried out in Budapest, Hungary, and featured four aerial robots, two rescue dogs, and three human rescuers carrying smartphone devices. All agents were connected through a mobile ad hoc wireless network. An overview of the SWARMIX project achievements was presented in [46] and is available at `https://youtu.be/WWUpqY3RoUw`. Screenshots of the MSS interface are shown in Fig. 13 and in the video attachment. We refer the interested reader to [21] for more details about the MSS and the WiSAR application.

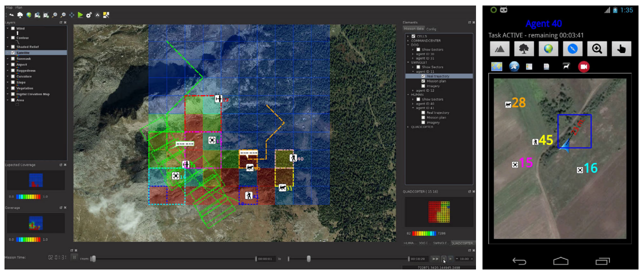


FIGURE 13. **(Left) A screenshot of the control center GUI that shows the completion map, routes, task assignments, and agents' trajectories. (Right) A screenshot of the smartphone application used by the rescuers to receive mission data and send status data to the MSS through a wireless network.**

B. MOBILE ROBOTIC TESTBED

The cooperative mission planning using the STASP-HMR was also demonstrated indoor, using a mobile robotic testbed. As experimental platform we have used the *foot-bot*: a small differential drive robot (about 15 cm wide and 20 cm high), developed in the SWARMANOID project (`www.swarmanoid.org`). We equipped each foot-bot with a wireless interface that operates at the 2.4 Ghz band and whose transmission power has been artificially constrained to enforce the use of multi-hop routes in our lab environment. Specifically, we have used Wi-Fi adapters with attenuators attached between the adapter and their external antennas. To set up a multi-hop wireless scenario we enforced a transmission power of 1 mW, a bit rate of 54 Mbps, and a signal attenuation of 20 dBm. This setup was sufficient to reduce the transmission range of the network to below 1.25 m.

In the *video attachment*, a team of robots executes one stage of the mission plan (replanning could be performed iteratively, similarly to the approach described in Section IX). A mission consists of *scouting a number of specified areas of interests*. Each area represents a task and we consider a total of 16 tasks embedded in a two-dimensional grid of dimension 4×4 cells. This setup emulates reconnaissance or search and rescue missions where multiple robots scout the environment for objects of interest. The demonstration also replicates the scenario where planning is performed using a *centralized controller* and spatio-temporal directives are used to balance task completion and wireless network connectivity in the computed plans. In this example application, wireless connectivity is related to the data exchange between the mobile agents and the centralized controller using a mobile multi-hop mobile wireless network.

We consider a total of 6 *mobile robots*, $A = \{a_1, \dots, a_6\}$. One static robot plays the role of the centralized controller, denoted as a_c, and it is located in the upper left corner of the area. Each mobile robot generates 1 Mbps of data towards the controller. A dynamic routing protocol is used to compute data routes based on the location of the robots [19]. Mission plans are computed following a *communication-aware strategy*, in the form of directives, that selects a subset of agents ($A_R \subset A$), called *relays*, to conform a chain, connected to the controller. Agents A_R are numbered $a_{r,1}, \dots, a_{r,|A_R|}$. Directives are classified into two groups: those that promote the formation of the chain, and those that promote the connectivity between the relays and the rest of the team $A \setminus A_R$.

The formation of the chain is guided by the first group of instant directives:

$$(\{a_{r,1}\}, \{a_c\}, t) \quad \forall t \tag{36}$$

$$(\{a_{r,i}\}, \{a_{r,i-1}\}, t) \quad \forall t,\ 2 \le i \le n. \tag{37}$$

where (A_1, A_2, t) denotes a network directive between group A_1 and A_2 at time t (see Section VI-A). The directives consider a bound $\bar{d} = 1.25$m which represents the transmission range of the wireless network.

Note that network directives specified by (36) link the controller to the first member of the chain, while those represented by (37) join the remaining elements of A_R to form a chain.

The *connectivity between the rest of the team and the chain* is promoted by directives

$$(\{a_k\}, \{a_c\} \cup A_R, t) \quad \forall a_k \in A \setminus A_R,\ t. \tag{38}$$

The directives are treated as soft constraints, and we include rewards in the objective function to promote their occurrence. We consider a ratio of 3 : 1 between the rewards of the directives corresponding to the formation of the chain ((36), and (commrelay:2)) and those corresponding to the connectivity of the rest of the agents to the chain ((38)). This decision is motivated by the fact that the formation of the chain is the key aspect of this strategy, and without it, the complete team can become permanently disconnected from the centralized controller. In the demonstrator, we observed an increase of received packets at the controller of up to 30% via the communication-aware strategy, when using two relays.

XIII. CONCLUSION

This work addressed the problem of mission planning in heterogeneous multi-agent/robot systems for missions that are composed of spatially distributed tasks. Tasks are considered as non-atomic and providing a utility for either a full or a partial completion. Team heterogeneity makes the agents showing different efficiency when dealing with the same task. The mission planning problem (named STASP-HMR) is defined as the joint problem of: selecting tasks, assign subset of tasks to each agent based on its task-solving efficiency, and for each subset define a schedule (time duration for dealing with the task) and routing (task sequence). The goal is to jointly solve all these sub-problems maximizing the overall utility gathered in a given time budget. The mission planning model is justified in many real-world scenarios that precisely present the characteristics considered here.

The mission planning problem is formalized as a MILP, a formulation that can enjoy the availability of effective standard solvers with performance guarantees, and that is flexible enough to easily accommodate the addition and removal of multiple constraints. An example of this versatility is the definition of a set of additional linear constraints that allow the time control of proximity relations among agent groups (e.g., for boosting communications and cooperation, or to avoid mutual interference). The mission planning model can be used to analyze certain properties of the system, such as the benefits of time-extended planning and the advantages of considering non-atomic tasks, that let the system to partially fulfill some tasks and better distribute the effort of the agents.

The STASP-HMR is proven to be NP-hard, a result that has urged us to find efficient solution methods. At this aim, we proposed two anytime algorithms: one matheuristic based on GAs, and one bounded optimal algorithm. Both algorithms significantly improve the performance of standard solvers, each offering specific advantages. We benchmarked our optimal solution approach against a state-of-the-art method for task allocation and scheduling, obtaining solutions that improve the performance of the team by a factor of 2.

The decentralized implementation, obtained top-down from the main MILP model, makes the STASP-HMR useful in practical scenarios where a continual online iterative replanning is necessary (e.g., to operate in closed-loop interaction with mission enrollment). Using simulations, we studied the impact of the different parameters of the online iterative replanning on team performance in both centralized and decentralized setups. We showed the excellent scalability of the decentralized algorithm, which dramatically drops down computation times, and only incurs in losses of order of 20% compared to a centralized solution.

Lastly, we described two real-world deployments of the STASP-HMR: an application in the search and rescue domain, and an implementation with networked mobile robots. The deployments demonstrate versatility, usability, and the practical effectiveness of the proposed models and algorithms.

Future work includes considering other types of task efficiency models (e.g., nonlinear, super-additive) and a comprehensive treatment of task dependencies, e.g., time-windows for task execution, multi-robot tasks that require certain combination of skills.

REFERENCES

[1] P. Abichandani, H. Y. Benson, and M. Kam, "Decentralized multi-vehicle path coordination under communication constraints," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2011, pp. 2306–2313.

[2] C. Archetti, N. Bianchessi, M. G. Speranza, and A. Hertz, "Incomplete service and split deliveries in a routing problem with profits," *Networks*, vol. 63, no. 2, pp. 135–145, Mar. 2014.

[3] B. A. Asfora, J. Banfi, and M. Campbell, "Mixed-integer linear programming models for multi-robot non-adversarial search," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6805–6812, Oct. 2020.

[4] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proc. IEEE*, vol. 94, no. 7, pp. 1257–1270, Jul. 2006.

[5] K. E. C. Booth, T. T. Tran, G. Nejat, and J. C. Beck, "Mixed-integer and constraint programming techniques for mobile robot task planning," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 500–507, Jan. 2016.

[6] A. Brutschy, G. Pini, C. Pinciroli, M. Birattari, and M. Dorigo, "Self-organized task allocation to sequentially interdependent tasks in swarm robotics," *Auto. Agents Multi-Agent Syst.*, vol. 28, no. 1, pp. 101–125, Jan. 2014.

[7] S. E. Butt and D. M. Ryan, "An optimal solution procedure for the multiple tour maximum collection problem using column generation," *Comput. Oper. Res.*, vol. 26, no. 4, pp. 427–441, Apr. 1999.

[8] J. A. Castillo-Salazar, D. Landa-Silva, and R. Qu, "Workforce scheduling and routing problems: Literature survey and computational study," *Ann. Oper. Res.*, vol. 239, no. 1, pp. 39–67, Apr. 2016.

[9] I.-M. Chao, B. L. Golden, and E. A. Wasil, "The team orienteering problem," *Eur. J. Oper. Res.*, vol. 88, no. 3, pp. 464–474, Feb. 1996.

[10] L. Chen and E. Miller-Hooks, "Optimal team deployment in urban search and rescue," *Transp. Res. B, Methodol.*, vol. 46, no. 8, pp. 984–999, 2012.

[11] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, Aug. 2009.

[12] Z. Chongjie and J. A. Shah, "Co-optimizating multi-agent placement with task assignment and scheduling," in *Proc. Int. Jt. Conf. Artif. Intell. (IJCAI)*, Jan. 2016, pp. 3308–3314.

[13] D. Claes, F. A. Oliehoek, K. Tuyls, and D. Hennes, "Effective approximations for multi-robot coordination in spatially distributed tasks," in *Proc. Intl. Conf. Auton. Agents Multiagent Syst. (AAMAS)*, 2015, pp. 881–890.

[14] M. D'Emidio and I. Khan, "Collision-free allocation of temporally constrained tasks in multi-robot systems," *Robot. Auton. Syst.*, vol. 119, pp. 151–172, Sep. 2019.

[15] M. Drexl, "Synchronization in vehicle routing—A survey of VRPs with multiple synchronization constraints," *Transp. Sci.*, vol. 46, no. 3, pp. 297–316, Aug. 2012.

[16] D. Feillet, P. Dejax, and M. Gendreau, "Traveling salesman problems with profits," *Transp. Sci.*, vol. 39, no. 2, pp. 188–205, May 2005.

[17] E. Feo Flushing, L. Gambardella, and G. A. Di Caro, "On decentralized coordination for spatial task allocation and scheduling in heterogeneous teams," in *Proc. Intl. Conf. Auton. Agents Multiagent Syst. (AAMAS)*, 2016, pp. 988–996.

[18] E. F. Flushing, L. M. Gambardella, and G. A. Di Caro, "Simultaneous task allocation, data routing, and transmission scheduling in mobile multi-robot teams," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 1861–1868.

[19] E. F. Flushing, L. M. Gambardella, and G. A. Di Caro, "On using mobile robotic relays for adaptive communication in search and rescue missions," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot. (SSRR)*, Oct. 2016, pp. 370–377.

[20] E. F. Flushing, M. Kudelski, L. M. Gambardella, and G. A. Di Caro, "Connectivity-aware planning of search and rescue missions," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot. (SSRR)*, Oct. 2013, pp. 1–8.

[21] E. Feo Flushing, L. Gambardella, and G. A. Di Caro, "GIS-based Mission Support System for Wilderness Search and Rescue with Heterogeneous Agents," in *Proc. 2nd IROS Workshop Robots Sensors Integr. Future Rescue Inf. Syst. (ROSIN)*, 2012, pp. 1–9.

[22] M. R. Garey and D. S. Johnson, *Computers Intractability; A Guide to Theory NP-Completeness*. New York, NY, USA: W. H. Freeman, 1990.

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

[24] D. E. Goldberg, *Genetic Algorithms Search, Optimization Machine Learning*, 1st ed. Reading, MA, USA: Addison-Wesley, 1989.

[25] M. C. Gombolay, R. J. Wilcox, and J. A. Shah, "Fast scheduling of robot teams performing tasks with temporospatial constraints," *IEEE Trans. Robot.*, vol. 34, no. 1, pp. 220–239, Feb. 2018.

[26] A. Gunawan, H. C. Lau, and P. Vansteenwegen, ‘‘Orienteering problem: A survey of recent variants, solution approaches and applications,’’ *Eur. J. Oper. Res.*, vol. 255, no. 2, pp. 315–332, Dec. 2016.

[27] F. Hammami, M. Rekik, and L. C. Coelho, ‘‘A hybrid adaptive large neighborhood search heuristic for the team orienteering problem,’’ *Comput. Oper. Res.*, vol. 123, Nov. 2020, Art. no. 105034.

[28] G. A. Hollinger and S. Singh, ‘‘Multirobot coordination with periodic connectivity: Theory and experiments,’’ *IEEE Trans. Robot.*, vol. 28, no. 4, pp. 967–973, Aug. 2012.

[29] H. Hoos and T. Stützle, *Stochastic Local Search: Foundation & Applications*. Burlington, MA, USA: Morgan Kaufmann, 2004.

[30] M. Koes, I. Nourbakhsh, and K. Sycara, ‘‘Constraint optimization coordination architecture for search and rescue robotics,’’ in *Proc. IEEE Int. Conf. Robot. Autom.*, Dec. 2006, pp. 3977–3982.

[31] 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, Oct. 2013.

[32] G. A. Korsah, B. Kannan, B. Browning, A. Stentz, and M. B. Dias, ‘‘XBots: An approach to generating and executing optimal multi-robot plans with cross-schedule dependencies,’’ in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 115–122.

[33] H. Ma and S. Koenig, ‘‘Optimal target assignment and path finding for teams of agents,’’ in *Proc. Intl. Conf. Auton. Agents Multiagent Syst. (AAMAS)*, 2016, pp. 1144–1152.

[34] Z. Ma, K. Yin, L. Liu, and G. S. Sukhatme, ‘‘A spatio-temporal representation for the orienteering problem with time-varying profits,’’ in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 6785–6792.

[35] V. Maniezzo, T. Stützle, and S. Voß, *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming* (Annals of Information Systems). New York, NY, USA: Springer, 2010.

[36] C. E. Miller, A. W. Tucker, and R. A. Zemlin, ‘‘Integer programming formulation of traveling salesman problems,’’ *J. ACM*, vol. 7, no. 4, pp. 326–329, Oct. 1960.

[37] M. D. Moskal and R. Batta, ‘‘Adaptive unmanned aerial vehicle surveillance using a prize-collecting vertex routing model,’’ *Mil. Oper. Res.*, vol. 24, no. 4, pp. 5–22, 2019.

[38] E. Nunes and M. Gini, ‘‘Multi-robot auctions for allocation of tasks with temporal constraints,’’ in *Proc. 29th AAAI Conf. Artif. Intell. (AAAI)*, 2015, pp. 2110–2116.

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

[40] L. E. Parker, D. Rus, and G. S. Sukhatme, ‘‘Multiple mobile robot systems,’’ in *Springer Handbook Robotics*. Berlin, Germany: Springer, 2016, pp. 1335–1384.

[41] R. Pěnička, J. Faigl, M. Saska, and P. Váňa, ‘‘Data collection planning with non-zero sensing distance for a budget and curvature constrained unmanned aerial vehicle,’’ *Autom. Robots*, vol. 1, pp. 1–20, Feb. 2019.

[42] S. Ramchurn and M. Polukarov, ‘‘Coalition formation with spatial and temporal constraints,’’ in *Proc. Intl. Conf. Auton. Agents Multiagent Syst. (AAMAS)*, 2010, pp. 1181–1188.

[43] M. S. Rasmussen, T. Justesen, A. Dohn, and J. Larsen, ‘‘The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies,’’ *Eur. J. Oper. Res.*, vol. 219, no. 3, pp. 598–610, 2012.

[44] S. Sariel-Talay, T. R. Balch, and N. Erdogan, ‘‘A generic framework for distributed multirobot cooperation,’’ *J. Intell. Robot. Syst.*, vol. 63, no. 2, pp. 323–358, 2011.

[45] P. Schillinger, M. Bürger, and D. V. Dimarogonas, ‘‘Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems,’’ *Int. J. Robot. Res.*, vol. 37, no. 7, pp. 818–838, Jun. 2018.

[46] A. Swarmix, ‘‘Finding Linda—A search and rescue mission by SWARMIX,’’ in *10th AAAI Video Competition*, 2016.

[47] T. C. Thayer, S. Vougioukas, K. Goldberg, and S. Carpin, ‘‘Multi-robot routing algorithms for robots operating in vineyards,’’ in *Proc. IEEE 14th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2018, pp. 14–21.

[48] T. C. Thayer, S. Vougioukas, K. Goldberg, and S. Carpin, ‘‘Bi-objective routing for robotic irrigation and sampling in vineyards,’’ in *Proc. IEEE 15th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2019, pp. 1481–1488.

[49] F. Thompson and R. Galeazzi, ‘‘Robust mission planning for autonomous marine vehicle fleets,’’ *Robot. Auto. Syst.*, vol. 124, Feb. 2020, Art. no. 103404.

[50] A.-E. Yahiaoui, A. Moukrim, and M. Serairi, ‘‘The clustered team orienteering problem,’’ *Comput. Oper. Res.*, vol. 111, pp. 386–399, Nov. 2019.

[51] V. F. Yu, P. Jewpanya, S.-W. Lin, and A. A. N. P. Redi, ‘‘Team orienteering problem with time windows and time-dependent scores,’’ *Comput. Ind. Eng.*, vol. 127, pp. 213–224, Jan. 2019.

[52] J. Zhou, X. Zhao, X. Zhang, D. Zhao, and H. Li, ‘‘Task allocation for multi-agent systems based on distributed many-objective evolutionary algorithm and greedy algorithm,’’ *IEEE Access*, vol. 8, pp. 19306–19318, 2020.

[53] M. Zhu, X. Du, X. Zhang, H. Luo, and G. Wang, ‘‘Multi-UAV rapid-assessment task-assignment problem in a post-earthquake scenario,’’ *IEEE Access*, vol. 7, pp. 74542–74557, 2019.

EDUARDO FEO-FLUSHING received the M.Sc. degree in informatics from the University of Trento, Italy, in 2010, and the M.Sc. degree in software system engineering from RWTH-Aachen University, Germany, in 2010, and the Ph.D. degree from the University of Lugano, Switzerland, in 2017. Between 2009 and 2011, he was an EU Erasmus Mundus Masters Scholar. From 2011 to 2017, he was with the Dalle Molle Institute for Artificial Intelligence (IDSIA), Switzerland. He is currently a Postdoctoral Associate with the Department of Computer Science, Carnegie Mellon University, Qatar. His research interest includes cooperation in heterogeneous networked multi-agent systems, which encompasses topics in optimization, wireless networking, and multi-robot systems.

LUCA MARIA GAMBARDELLA is currently an Internationally Recognized AI Expert, with 35 years of experience in the field. He is also a Full Professor with the Faculty of Informatics, USI, Lugano, and a Professor with the IDSIA USI-SUPSI, which he directed for 25 years. He is the Co-Founder, CTO, and the Head of applied AI at artificialy in Lugano, Switzerland. Scientifically, he is one of the pioneers of the ant colonies optimization metaheuristics. In his career, he has successfully negotiated, conducted, and delivered many AI and ML projects with Swiss and international research agencies and companies. He has published more than 300 scientific articles, with H-index 70, and more than 55000 citations.

GIANNI A. DI CARO received the degree *(magna cum laude)* in physics from the University of Bologna, Italy, in 1992, and the Ph.D. degree (Hons.) in applied sciences from the Université Libre de Bruxelles (ULB), Belgium, in 2004. He was a Senior Researcher with the IDSIA, Lugano, Switzerland, from 2003 to 2016, a Postdoctoral Marie Curie Fellow with the IRIDIA/ULB, from 1996 to 1999 and from 2001 to 2003, and a EC S&T in Japan Fellow at the Advanced Telecommunications Research (ATR), Japan, from 1999 to 2001. Since 2016, he has been an Associate Teaching Professor with the Department of Computer Science, Carnegie Mellon University (CMU), Doha, Qatar. His research and teaching are the domains of autonomous robotics, multi-robot systems, swarm intelligence, artificial intelligence, machine learning, and optimization, networking. In these fields, he has coauthored more than 170 scientific works that, according to Google Scholar, have received more than 34,000 citations with H-index 42.