On Decentralized Coordination for Spatial Task Allocation and Scheduling in Heterogeneous Teams

Eduardo Feo Flushing, Luca M. Gambardella, Gianni A. Di Caro
Dalle Molle Institute for Artificial Intelligence (IDSIA), USI/SUPSI
Lugano, Switzerland
{eduardo, luca, gianni}@idsia.ch

ABSTRACT

In the context of coordination and planning in collaborative multi-robot/agent systems, we consider a general reference problem that includes tasks that are spatially localized and have an associated service time, and accounts for the use of a heterogeneous team, in which different robots may have a different performance on the same task. A mixed integer linear formulation is introduced and used to solve the problem model in a centralized iterative manner: in closed-loop, team-level plans are adaptively computed and sent out. Unfortunately, a centralized scheme can suffer from computational and communication shortcomings. Therefore, we introduce a top-down recipe for decentralization, aiming to balance the tradeoff among implementation costs, computational requirements, and quality of coordination. The decentralized architecture depends on various aspects that we study through an empirical sensitivity analysis. Results show that the impact and the relationships among the different aspects are far from being obvious or intuitive. A number of practical lessons are learned, that could apply to other similar problems and/or decentralized architectures derived in the same top-down modality.

Keywords

Multi-Robot Systems; Decentralized Coordination and Planning; Collaborative Teams; Task Allocation

1. INTRODUCTION

It is well understood that, in general, the support of a *coordinated planning* scheme is necessary to optimize the performance of a *multi-agent team*. Coordination is required to avoid conflicts, unnecessary overlapping, and incoherent executions, while, at the same time, to boost cooperation and synergies when tackling a common mission. This is even more true when the team is heterogeneous, due to the fact that different agents might perform differently for the various sub-tasks composing the overall mission.

In this work, we focus on mission scenarios that require the use of embedded physical agents, a mobile *multi-robot team* in particular, and that are defined through a set of *spatially distributed tasks*, each with its own characteristics and demand levels. The class of problems that we consider generalizes the classical multi-robot task allocation problems. It includes the notion of spatial locality [22, 10] and service time, which enrich the mission model with a routing and scheduling component. We refer to the problems in this class as the *spatial task allocation and scheduling problem in heterogeneous multi-robot teams*, or STASP-HMR in short.

In practice, our modeling approach is derived from complex *search and rescue* scenarios, that require to coordinate a heterogeneous team for searching over an area, transporting items, mapping, detecting dangerous events, and so on. In these cases, due to the inherent uncertainties of the environment to act upon, agent planning cannot be usually done in one shot, but instead needs to be performed iteratively, in *closed-loop with mission enrollment*, to account for new evidence, deviations, and unexpected issues.

In previous work [13], we tackled the heterogeneous team-level planning through the use of a *linear mixed integer mathematical formulation*, which was solved using the combination of standard solvers and ad hoc heuristics that was able to retain *anytime* properties and formal guarantees on solution quality while speeding up computations. The planner operates in a *centralized, closed-loop modality*: a joint plan is issued, then updates are continually sent from the executing agents to the planner, and in turn this triggers the computation of new plans, iteratively. This way of proceeding has obvious limitations, especially in the search and rescue scenarios of reference, where communication infrastructures are hardly available, many unexpected events can happen on the field requiring continual replanning, and the criticality of the situation requires a very rapid adaptation and response.

To overcome these well-known issues, in this paper we propose a *decentralized approach* for STASP-HMR, where each agent runs a replica of the mathematical model, based on local data and limited information sharing. Apart from the characterization of the above class of problems, our *contribution* here is two-fold. First, we introduce a decentralized coordination and planning algorithm that meets computational and communication requirements, and at the same time incurs in limited performance losses compared to the centralized solution. Second, we propose a *general top-down recipe* for deriving an effective decentralized scheme from a centralized mathematical model. Moreover, a *third contribution* consists in the identification of a set of interrelated design aspects that have a critical impact on the performance of a decentralized team, and in performing a *sensitivity analysis* (in simulation) with respect to these aspects.

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

In particular, we consider team heterogeneity, amount of information sharing, planning look-ahead, and replanning frequency. A number of not trivial results have been observed and practical lessons have been learned. The claim is that these same results could apply to other classes of decentralized coordination systems obtained following a top-down approach from a centralized mathematical model.

2. BACKGROUND

According to the level of coordination that they demand, multi-robot missions can be categorized into tightly-coupled and loosely-coupled missions [15, 5, 31, 18, 2, 25]. Tightly-coupled missions are composed of tasks with strong coordination requirements. These include coalition formation scenarios in which robots must complement their skills and coordinate their actions to act, as a team, upon certain tasks [28, 30, 26], and scenarios that impose ordering constraints between tasks that require negotiation and coordination among the agents so as to prevent conflicts and deadlocks [18, 7, 11, 21, 20, 14]. Other forms of tightly-coupled missions are inter-schedule dependencies [19] that restrict the set of actions of a single robot depending on the tasks being carried out by other robots, e.g., for preserving network connectivity [6, 23], or coalitions [31]. In general, these tightly-coupled domains involve inter-task or inter-agent dependencies that affect the feasibility of the mission, therefore they require the use of strong coordination mechanisms.

On the contrary, the degree of coordination in loosely-coupled missions may have an impact on their performance but does not compromise their feasibility. In practice, these missions could rely on local decisions made by the agents without any form of agreement among them. However, the lack of coordination in these missions may result, for instance, into a duplication of effort and a waste of resources [4], therefore having a negative effect over team performance. Thus, coordination is desirable but not strictly necessary. Among these type of missions we find scenarios that are modeled as foraging tasks, with the objective of minimizing completion time or maximizing some notion of reward, e.g., search, surveillance, coverage, patrolling [1, 17].

Decentralized approaches surge as a response to the drawbacks and limitations of their centralized counterparts. They eliminate the need for a central entity by transferring the control authority to the agents that execute the mission. As a result, the agents become autonomous decision-making elements in charge of planning their own actions. This transfer of control provides fault tolerance, resiliency, and scalability.

Several decentralization strategies exist in the literature, tailored to fit the different coordination requirements of the missions that they tackle. Among them, implicit coordination is a common and flexible way to deal with loosely-coupled missions [17]. It is achieved by having each agent, independently, computing a (global) solution utilizing a centralized planner. From this solution, each single robot retrieves its own plan. The basis of this method is that, when all robots possess a similar knowledge about the environment, they would compute similar solutions. As a result, they implicitly coordinate and plan their own actions.

In this work, we analyze the tradeoff between the choice of coordinated actions and the performance of a decentralized scheme based on the implicit coordination principle, in the context of a loosely-coupled scenario. This tradeoff is not well understood in the literature, and it is also unclear how to quantify the performance degradation of reduced coordination in these domains [29, 3]. Interestingly, we can empirically study these properties by comparing the performance of a centralized planner, which computes optimal mission plans using full knowledge, against the one of a decentralized solution. In this work we follow this approach to investigate the effect of several design choices and their different parameters in implicitly coordinated missions.

3. SPATIAL TASK ALLOCATION & SCHEDULING IN HETEROGENEOUS TEAMS

In this section, we define the class of problems that we address in this work, namely the STASP-HMR. We first introduce the concepts that underlie the model of the mission. Then, we provide a concise statement of the STASP-HMR.

3.1 Mission representation

Let $\mathcal{A}$ be a team of heterogeneous mobile agents that are available to perform a *joint mission* in an environment of a specified dimension. We assume that the mission has been decomposed into a set $\mathcal{T}$ of *spatially distributed, location-dependent tasks*. These tasks can be *non-atomic*, incrementally providing a reward proportional to the progress achieved in their completion, and can be eventually brought to an end.

The spatial layout of tasks is captured by a *traversability graph* that defines how agents can move between tasks. In this respect, we define a directed traversability graph $G = (\mathcal{T}, E)$ where E contains an arc (i, j) if task j can be scheduled right after task i. In the general case, graph G is complete (i.e., $E = \mathcal{T} \times \mathcal{T}$), and, in this way, we are assuming that all tasks are independent from each other. However, G can be used to accommodate constraints over the sequences of tasks that can be executed. For instance, when some tasks cannot be designated immediately after others, e.g., due to mobility constraints, or when specific tasks must be serviced immediately before servicing others, e.g., unblocking a road to reach other parts of the area.

From the point of view of the mission, the complete execution of any task $\tau \in \mathcal{T}$ provides an overall *utility*, or *reward*, indicated with R_τ. The values assigned to R_τ can be used to indicate how important is the task for the mission, e.g., in a search and rescue mission when some locations are more important to search accurately than others based on some a priori knowledge.

3.2 Task efficiency model and completion

For a given mission, we can look at tasks as requests to be possibly serviced, and at the agents as the entities with the capabilities to service these requests. When we deal with heterogeneous teams, different agents may demonstrate different levels of performance accomplishing the same task.

In order to accommodate team heterogeneity we propose the use of *task efficiency models* that allow, as their name suggest, to specify the efficiency with which an agent services a specific task. The intuition behind these models is that any progress on the completion of a task is proportional to the overall time devoted to it. Agent a is more efficient than agent b on the same task τ if in the same amount of time a pushes τ closer to its completion.

The definition of the task efficiency models should be based on the specific skills of the agents in relation to the

specific local characteristics of each task. For instance, the time effort required for pushing a box across a room is dependent upon how much force a robot is able to apply to the box and on well it can detect possible obstacles along the way. We assume that the efficiency models, for each single agent, are known and defined as $\varphi : \mathcal{A} \times \mathcal{T} \mapsto \mathbb{R}$. $\varphi_k(\tau),\ k \in \mathcal{A}, \tau \in \mathcal{T}$ relates the time an agent spends doing a task to the progress achieved towards its completion. The value of $\varphi_k(\tau)$ is inversely proportional to the amount of service time that k requires for the completion of τ.

In the following, for the sake of simplifying the formulation, we assume that time is discretized into *mission intervals* of equal length Δ_T. In addition, with the purpose of avoiding the challenges posed by the solution of non-linear models, we assume that task efficiency models follow a linear law. More precisely, when an agent $k \in \mathcal{A}$ performs a task $\tau \in \mathcal{T}$ for t mission intervals, it contributes with $100 \times \varphi_k(\tau) \cdot t$ percent to the *task completion*.

To express the residual service needed by each task, we use the *completion map* $C_m : \mathcal{T} \mapsto [0, 1]$. The values in the completion map identify the *current completion level* of tasks whose workload has been partially addressed in previous planning stages. For instance, a value of 0 for $C_m(\tau)$ indicates that task τ has been already completed in the past, and therefore no further efforts from the agents are required. If an agent attempts to further deal with that task it will receive no additional reward, which will amount a waste of time and resources. Servicing a fraction p of the workload of task τ decreases its required completion $C_m(\tau)$ by p and provides a partial utility of pR_τ.

3.3 Definition of STASP-HMR

Based on the concepts introduced above, the STASP-HMR can be stated as follows. Given a set of heterogeneous agents, each characterized by its task efficiency model, and a *limited time budget* T, the STASP-HMR consists in determining joint plans for the activities of the agents in the environment which enable the timely selection of the tasks to perform, and aiming to maximize the overall mission utility (i.e., the sum of all gathered rewards).

A *solution to the STASP-HMR* (i.e., a mission plan) consists of time-constrained sequences of tasks, one for each agent, that can be represented as paths in G. In addition, plans also define how much effort (i.e., devoted time) each one of the selected tasks will receive. Due to the limited time budget, not all tasks may be performed: a plan implicitly defines a selection among all tasks in $\mathcal{T}$. The goal is to assign the right agent to deal with the right task, at the right time. Here "right" means the agent who is the most efficient for the task among the agents in the vicinity.

In real-world scenarios, the available information concerning the environment is usually incomplete or imperfect, and the environment itself is often dynamic. Moreover, deviations in the agents' response to plans may occur as a consequence of unexpected sensing/actuation problems. For these reasons, a solution approach to the STASP-HMR in real-world scenarios should adapt the issued plans to these dynamic aspects. This calls for a *closed-loop* approach: while plans are executed, mission information is gathered from the agents, based on this information new plans are adaptively issued, and the process is iterated until mission completion. This is precisely the approach we follow, as described in Section 4.1.

4. CENTRALIZED SOLUTION APPROACH

We formulate the STASP-HMR as stated above by means of a *mixed-integer linear program* (MIP). An optimal solution to the MIP defines plans for each one of the agents with the goal of maximizing the total mission reward. For the sake of completeness, we present the MIP, and refer the interested reader to [13] for a detailed description.

$$\text{maximize} \quad \sum_{\tau \in \mathcal{T}} R_\tau \Phi_\tau \tag{1}$$

subject to

$$\sum_{(0,j) \in E} x^k_{0j} = 1 \qquad k \in \mathcal{A} \tag{2}$$

$$\sum_{(i,0) \in E} x^k_{i0} = 1 \qquad k \in \mathcal{A} \tag{3}$$

$$\sum_{(i,j) \in E} x^k_{ij} = \sum_{(j,i) \in E} x^k_{ji} = y^k_j \qquad k \in \mathcal{A},\ j \in \mathcal{T} \tag{4}$$

$$t^k_i + w^k_i - t^k_j \leq (1 - x^k_{ij})\, T \qquad k \in \mathcal{A}, (i,j) \in E, i, j \neq 0 \tag{5}$$

$$y^k_i \leq t^k_i, w^k_i \leq T y^k_i \qquad k \in \mathcal{A},\ i \in \mathcal{T} \tag{6}$$

$$\Phi_\tau \leq \sum_{k \in \mathcal{A}} \varphi_k(\tau) w^k_i \qquad \tau \in \mathcal{T} \tag{7}$$

$$0 \leq \Phi_\tau \leq C_m(\tau) \qquad \tau \in \mathcal{T} \tag{8}$$

$$\Phi_\tau \in \mathbb{R} \qquad \tau \in \mathcal{T} \tag{9}$$

$$t^k_i,\ w^k_i \in \mathbb{N} \qquad k \in \mathcal{A}, i \in \mathcal{T} \tag{10}$$

$$x^k_{ij},\ y^k_j \in \{0, 1\} \qquad k \in \mathcal{A},\ i, j \in \mathcal{T} \tag{11}$$

We use the following *decision variables* to build the MIP model for the STASP-HMR presented in (1)-(11):
x^k_{ij}: binary, equals 1 if agent k traverses arc $(i, j) \in E$;
y^k_i: binary, equals 1 if agent k is assigned to task $i \in \mathcal{T}$;
Φ_τ: service provided to task $\tau \in \mathcal{T}$ by all agents;
t^k_i: starting time of execution of task $i \in \mathcal{T}$ by agent k;
w^k_i: time assigned to task $i \in \mathcal{T}$ for agent k.

The objective function (1) defines the quality of a mission plan in terms of its utility, quantifying the expected effect of agents' activities over the current state of the completion map C_m. A dummy vertex (denoted by 0) represents the starting point and ending point of the agents' paths. Graph G is extended with arcs from 0 to each one of the tasks that are initially accessible. Constraints (2-4) ensure path continuity. Constraints (5) eliminate sub-tours [27] and, together with (6), they define the bounds on the variables t and w. The completion levels of each task are bounded by constraints (7-8). These bounds ensure that each task τ provides a maximum reward equal to $R_\tau C_m(\tau)$, and that the utility of a plan is contributed with R_τ scaled by the completion of τ (i.e., Φ_τ). Finally, constraints (9-11) set the real, integer, and binary requirements on the model variables.

In general, the above MIP can be computationally hard to solve and therefore finding an optimal mission plan may take a significant amount of time. However and conveniently, standard MIP solvers have *anytime* properties: solutions are progressively and monotonically improved over time, and can be retrieved with formal error bounds on optimality. In our case, to exert a proper control of the time spent computing a plan, the planner sets a maximum time t_{plan} to the MIP solver, after which the solver stops the optimization and returns the best solution found.

4.1 Online iterative replanning

For a given mission, the MIP formulation proposed above could be used to perform a *one-shot* planning. Yet, as mentioned in Section 3.3, the offline computation of mission plans is not practical since it cannot accommodate the dynamic aspects of real-world environments.

To this end, we consider an online, iterative replanning approach based on the continual acquisition of new information about the environment and the current status of the agents. The iterative replanning is implemented as a *multi-stage, periodic, re-optimization procedure* [16, 8] that consists in solving, over time, a sequence of static problem instances with varying horizon length. During the entire mission, the planner continuously gathers new information about the state of the environment, as well as the current status of the agents and of the tasks they are executing. This information is taken into account in the next replanning iteration, so as to make the new plans adapting to the changing situation.

More in detail, the iterative replanning works as follows. At any single planning stage, the planner sets a horizon T_H that defines how far to look ahead during planning. Longer horizons should enable a better allocation of resources, but also increase the computational complexity as the size of the associated problems increases [14, 22]. In contrast, shorter horizons tend to be computationally affordable, but also to make agents to reason myopically, resulting into lower-quality decisions. This tradeoff is discussed in Section 6. After the value T_H has been chosen, it is used as T in the MIP model associated to the current stage.

The selected scheme for online replanning must guarantee that, at any time, an agent has a plan to follow. Recall that a given mission plan has a limited time span that depends on the value of the last horizon T_H used. Beyond this time span, the agents do not have any planned actions. Therefore, at the planner, the computation of the next plan needs to be triggered before the current plan expires, so as to extend the plan and let the agents to continue with the mission.

Without losing generality, we assume that replanning can be triggered within an interval $[t_a, t_b]$ with $1 \leq t_a \leq t_b \leq T_H - t_{plan}$ measured from the time when the last plan was issued. When $t_a = 1$, replanning can 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 the latest, just before the current mission plan expires (i.e., while agents execute their last task assigned). In Section 6 we extend the discussion about the choice of $[t_a, t_b]$ and the impact this parameter has on the performance of iterative replanning.

5. TOP-DOWN DECENTRALIZATION

The centralized iterative replanning scheme presented above provides several advantages: allows to place much of the computational load to the central controller, therefore reducing the cost and the complexity of the team [9], makes the whole operation relatively easier to implement, and, if the controller has an accurate, global view of the mission, then it is possible to compute optimal mission plans, that take into account global constraints. On the other side, this scheme suffers from scalability issues, meaning that, for large team sizes, the computational complexity and communication requirements prohibit its use [24]. These issues are even more evident when the replanning rate becomes higher.

In the following of this section we describe the top-down approach for solving the STASP-HMR in a decentralized manner. It addresses the drawbacks and limitations of its centralized counterpart, by providing fault tolerance, resiliency, and scalability. However, these appealing characteristics may often lead to sub-optimal, lower quality solutions The approach is based upon implicit coordination and replicates the centralized scheme on each agent. As a result a decentralized decision making is achieved by letting each agent planning its own actions.

In order to achieve decentralized coordination, agents must make their decisions on the basis of a common ground. Therefore, we first enumerate what and how data should be exchanged among the agents in order to maximize their local perception of the mission. Lastly, we discuss how the local data are exploited for the efficient computation of plans and for the implicit coordination the mission.

5.1 Shared knowledge representation

Apart from the static information about the problem, which can be initially distributed among the agents, there are two main types of information that need to be revised, and updated, in order to replicate the iterative replanning scheme at each agent: the completion map C_m and the planned actions of the other agents.

During the execution of a plan, the executed tasks receive certain amount of service that decrease the initial workload required for them. In a centralized setting, this information would be gathered at the central controller and used to keep an up-to-date global completion map C_m. However, this is not the case in a decentralized scheme, and agents should keep a local estimation of the C_m. The more accurate these estimates are, the better plans an agent can compute.

In addition, it is also useful to know the scheduled future actions of other agents. For instance, let us assume that an agent a is about to compute a local plan and, according to its local perception of C_m, there is a nearby task τ that requires full service. Even if such estimate is accurate and indeed τ has not been attended yet by any other agent, the inclusion of this task in a's current plan could be a bad decision if there exists another agent b that has already included τ in its plan. These situations can become frequent and, unless some explicit fallback strategy is executed (e.g., detecting ad preventing the overlapping of actions), they degrade the performance of the mission.

Both types of data, the completion map and the plans, are described in terms of *incremental updates* that, once added up all together, allow to reconstruct a global shared representation of the mission. In the case of the completion map, an incremental update is a tuple $C_m^u = < k, \tau, t_{start}, t_{end} >$ that indicates that agent k has provided service to task τ from time t_{start} to t_{end}. These updates are issued by the same agent that performs the service. Each agent uses the C_m^u's that are issued by other agents to update its own local C_m. This is done by computing the amount of service provided to each task using the efficiency models φ_k. Note that it is sufficient to aggregate the updates, independently of the time at which they were generated, in order to approximate C_m to its true value. An important remark is that agents should not apply twice the same update. To this end, each C_m^u is uniquely identified by a sequence number, assigned by the agent that generates the update.

Regarding the mission plans, a similar approach is followed. A plan update is a tuple $< k, t_{gen}, \tau, t_{start}, t_{end}, >$ that indicates the intention of agent k to perform task τ from t_{start} to t_{end}. Note that in this case we also use an additional time stamp t_{gen} to specify the time at which k expressed such intention. At any time instant t, we can (partially) reconstruct the mission plan of an agent k by considering all the plan updates with $t \leq t_{start}$. The values of t_{gen} allow us 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 table.

5.2 Information exchange

In a decentralized operation, communication among the agents enables the acquisition of relevant information regarding the past, current, and planned activities. In real-world deployments, communication is usually supported by a wireless network. However, when networked teams operate in large-scale environments, data exchange can become sparse, because it is restricted by the transmission range of the network.

Based on the shared representation, the final goal is to exploit at most the communication interactions among the agents. To this end, a gossip-based mechanism[12] helps to spread the information about the mission among the team. This mechanism is implemented as follows.

During the mission, all agents broadcast periodic network discovery messages that serve to announce their presence to other agents. Using these messages, each single agent keeps a list of all other agents from which it has received a message recently. From this list, and with certain frequency, the agent chooses another agent to initiate a synchronization procedure. This procedure enables the agents to merge and synchronize their local knowledge and possibly improve their local perception of the mission.

5.3 Adaptive replanning using the MIP

To perform the iterative replanning, each agent uses its local estimate of the the completion map, and of the current (partial) plans of other agents to set up the MIP.

The completion map C_m is used to define constraints (8). The plans of other agents are used to fix some of the routing and scheduling variables t_i^k, x_{ij}^k, and w_i^k. Since the agents can spread out over the area, data exchange can be sparse. As a result, agents may become unaware of the activities of some members of the team. Therefore, when building the MIP, each agent only considers the teammates from which it has recently gotten some information update. The general idea is to rely only on fresh information, that can be considered as reliable.

Each agent uses the MIP to primarily compute its own plan. However, the MIP is intended to optimize the global joint mission plan. This means that, when an agent a only has partial information about the plans of the other agents, a must either infer the missing parts of the plans or let the solver compute them. In this work, we adopt the former approach. Partial plans are completed by an agent a using a heuristic procedure, which works as follows. First, the tasks in the vicinity of a that still need to be completed are listed. Then, in sequence, for each other agent from which a has reliable information and an incomplete plan, a greedy task assignment is performed: the next task from the list that maximizes the agent reward is assigned to it in the MIP formulation. As a result, the set of decision variables of the MIP is drastically reduced to the variables that only define the actions of the agent a. The decision variables regarding the other agents are fixed and treated as constants. In our tests, we note that this way of proceeding leads to substantial computational savings and better performance compared to solving the entire MIP. 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 effectively computed in the limited time budget allocated for planning.

6. SENSITIVITY ANALYSIS

In this section we present a sensitivity analysis of the proposed decentralized scheme. The analysis is based on simulations. In order to obtain solutions to the mathematical programs we use CPLEX® as MIP solver, combined with a simple greedy heuristic that provides an initial feasible solution. In the following, first, we describe the instances of the STASP-HMR that constitute the benchmark of the analysis. Next, we use these instances to study the computational performance of the centralized implementation and to derive bounds on mission performance. These bounds are used as a performance baseline for the analysis of the decentralized system. In the remaining part of this section we focus the sensitivity analysis on the following aspects: team heterogeneity and spatial correlation among the tasks, amount of information sharing, and length of the planning horizon and frequency of replanning.

6.1 Experimental setup

We consider problem instances where the locations of the tasks composing the mission are obtained through a grid decomposition of the area with a one to one correspondence between tasks and grid cells. The grid size is 20×20, for a total of four hundred tasks. We adopt a dimensionless space, yet we consider that robots must move through adjacent cells in the Moore neighborhood (i.e., if they share at least one vertex on the grid). This condition is translated into the traversability graph and imposes routing constraints on the set of feasible sequences of tasks.

We let the task locality to impose limits on the communication. In some instances, data exchange between two robots occur only within a limited communication range ψ_{r}. To simulate the communication constraints 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} they are able to exchange data and to synchronize their current knowledge using the mechanisms described in Section 5.2.

We consider four types of agents, each characterized by the efficiency in accomplishing the tasks. As introduced in Section 3, the value of φ precisely relates each robot to the efficiency in doing each task. Numerically, φ defines the fraction of workload of a task that a robot can tackle during a single time step. For each single task, we consider that any robot type exhibits one out of five different efficiency levels, ranging from completely inefficient ($\varphi \triangleq 3\%$) up to highly efficient ($\varphi \triangleq 100\%$). These models are assumed to be known by all agents, and to remain constant during the mission. We setup a mission by defining the number

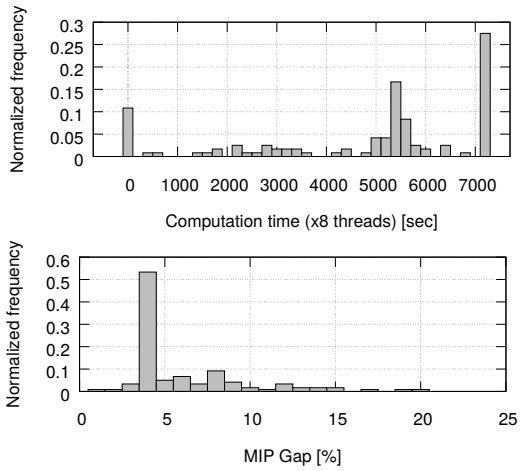

Figure 1: Computational results of the centralized planning over 120 STASP-HMR instances with 12 agents.

of agents of each type (which could be zero), together with their initial deployment. The total number of agents in a mission is 12 unless otherwise specified.

All simulated missions have a duration of 48 steps. At the start, the robots are deployed at random positions inside the grid. They all compute an initial plan, and then execute the online replanning in an asynchronous way, as described in Section 5. At each replanning stage, each agent allows a maximum of $t_{plan} = 150$ seconds to compute a next plan using the MIP solver. This time limit has been chosen to emulate a real-world scenario where plans meant to be computed in near real-time on embedded systems with potentially low computing power (e.g., small flying robots).

6.2 The complexity of centralized planning

As an initial step in our analysis we use the solver to compute global solutions under a centralized setting and with full knowledge of the environment. The goal of the experiments reported here is twofold. First, to observe the complexity of the centralized planning. Second, to use the computed solutions to derive a bound on the best achievable performance of a decentralized execution in any single problem instance.

We compute the baseline solutions using the mathematical solver, imposing a time limit of two hours and a memory limit of 8 GB, and using up to 8 parallel threads, each one allocated to an individual core in a computing cluster. Upon reaching the limits of computational resources, the solver returns the best solution found $Best^{cent}$. It also provides a measure of the solution quality in terms of a *relative optimality gap* (G_{opt}^{cent}), also called MIP gap. This gap is related to a lower bound (LB^{cent}) that is derived from the linear programming relaxation. The value of LB^{cent} is computed and improved by the solver during the solution process by relaxing the integrality constraints through the branch and bound search and by automatically generating cutting planes. We let the solver stop the computation when G_{opt}^{cent} becomes less than 5%, meaning that we obtained a satisfactory solution.

Figure 1 shows the distribution of the computation time (top) and MIP gaps (bottom) for a set of 120 problem instances using a centralized planner. Results show that most of the instances satisfied the relative optimality gap of 5%. In addition, besides a small subset of instances, the computation time taken by the solver was in general above 5,000 seconds, which multiplied by 8 threads/cores, it is equivalent to over 11 hours of computation. From these results we can appreciate the complexity of centralized planning. It is precisely this computational complexity one of the aspects that we expect to overcome by using a decentralized approach.

To evaluate the relative performance of the decentralized implementations with respect to the centralized one, we use the solutions obtained above to define a baseline metric. More precisely, for each problem instance, we compute the relative gap G_{opt}^{dec} between the decentralized solution and the centralized bound LB^{cent}. We consider this metric instead of using directly $Best^{cent}$ because, as noted in Figure 1, in many cases it is computationally unfeasible to reduce G_{opt}^{cent} to zero given the limited computational resources available.

6.3 Team diversity and tasks' correlation

A core feature of our solution approach to STASP-HMR is that we optimize the match between tasks and agents allocated to them. In this respect, heterogeneity could become an advantage when different tasks can benefit from different skills. However, in order to exploit heterogeneity, the plans for the individual agents need to be well coordinated among each other. This, in turn, depends on the spatial correlation among the tasks with respect to the task efficiency models of the agents.

When the task efficiency models are not spatially correlated, we can expect that the efficiency of an agent changes abruptly over neighbor tasks. On the contrary, when the models exhibit certain degree of spatial structure, changes are smooth, which also means that once an agent is performing a task for which is good for, we expect that it would be also well suited for the surrounding tasks.

The spatial correlation aspects also affect the way conflicts among the agents arise and are solved by local coordination. In the uncorrelated case, conflicts are sparse, and more difficult to solve due to the lack of smoothness of the models. For instance, when an agent finds itself competing with others for some task of high value for it (e.g., it is well suited for the task), it becomes more difficult to coordinate and maybe sacrifice its individual objective (for the sake of the mission) and select other tasks in the vicinity that have a similar value. However, when the efficiency models are spatially correlated, there is a good chance of solving the conflict with slight local changes in the plans.

With respect to team diversity, it remains to understand how it affects the performance of the decentralized scheme, and what is the role of coordination in this context. Perhaps more interestingly is to study the interplay between the choice of a heterogeneous team and the degree of spatial structure of the task efficiency models.

In this section, we consider two classes of missions. In the first class, tasks efficiency models follow a discrete uniform random distribution. In the second class, we use random fractals to obtain spatially structured surfaces from which we generate models that exhibit some degree of spatial correlation. Figure 2 depicts one instance of each class to illustrate the difference between them.

Discussion: Figure 3 shows the performance of the decentralized scheme for both classes of missions. The plot shows the median value and 25% and 75% error bars of the performance of the decentralized schemes, measured in terms of G_{opt}^{dec}. The performance for each class is analyzed

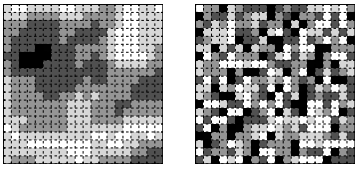

Figure 2: Visual representation of two efficiency models with (left) and without (right) spatial correlation. Each color represents an efficiency level.

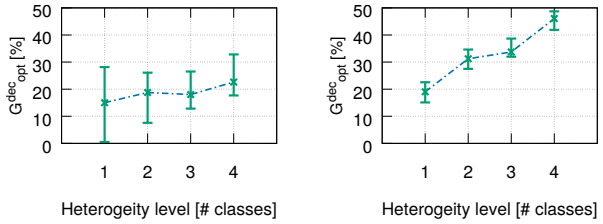

Figure 3: Effect of team diversity on instances with spatially correlated models (left) and with random uniform task models (right).

with respect to the level of heterogeneity (i.e., increasing the number of agent classes). From the results, we immediately observe that, in both cases, the performance with respect to the centralized model degrades when the heterogeneity level is increased. Surprisingly, these effects are stronger for uncorrelated models

The lesson learned from this discussion is the following. If we employ an heterogeneous team with efficiency models that exhibit low spatial correlation, we should be aware that the decentralized solution can show a performance (relative to the centralized solution) significantly lower than the one obtained with a homogeneous team. At last we point out that, at the global system level, heterogeneity is beneficial. For instance, the following table shows the percentage of performance improvement, with respect to homogeneous teams, for the centralized model.

# agent classes	2	3	4
uniform random	20	34	32
spatially correlated	24	31	34

These numbers indicate that well-coordinated heterogeneous teams perform better than homogeneous teams.

6.4 Local vs. global information sharing

In Section 5 we propose a shared representation of the mission on top of which the decentralized scheme is implemented. In this section we analyze the value that each piece of information (i.e., completion map, and partial plans) brings to the performance of the decentralized approach. In the presence of communication constraints, we also analyze the effect of inconsistent knowledge among the agents.

We consider two different classes of scenarios. In the first, robots are allowed 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.

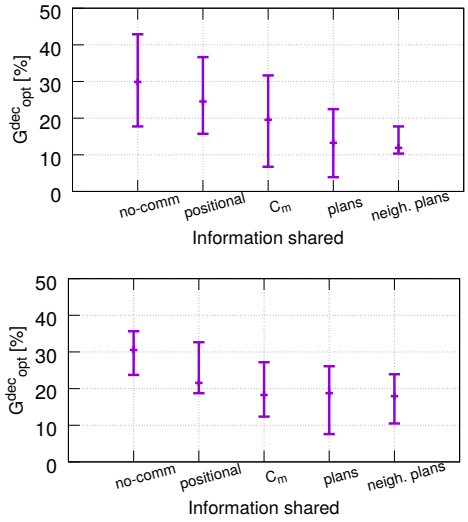

Figure 4: Effect of information sharing with full (top) and constrained communication (bottom).

We simulated 40 missions with spatially correlated models and using a team of 12 robots with two classes of agents. We distinguish 4 types of scenarios that add, incrementally, the following data to the list of data that are shared among the agents: (1) positional information; (2) completion map C_m updates; (3) partial mission plans; (4) plans computed for other agents. In the latter case, agents also share, with a subset of their neighbors, the plans that they complete for them. We believe that this strategy can help save computational resources since the agents utilize plans that have already been computed by another agents to extend their current plans, without resorting to the MIP. However, it is unclear if this provides some gain in performance.

Discussion: Figure 4 summarizes the value of each piece of information in both full and constrained communication scenarios. The data labeled as "no-comm" correspond to the case where agents do not exchange any information. An important observation is that sharing the completion map is sufficient to guarantee a performance within 20% from the centralized solution. This suggest that this piece of information is the most valuable one for the implicit coordination scheme. The rest of the information only makes a difference, albeit relatively small, if it can be spread out among the agents. Lastly, we remark that the strategy of sharing computed plans with the neighbors does not improve the performance in a significant way. However, it can be used to reduce the amount of planning iterations that agents must execute and, in overall, to reduce the cost of planning, while at the same time providing similar performance with respect to the case where this information is not shared.

6.5 Planning horizon and replanning frequency

Two of the most important parameters of the online replanning are the length of the lookahead planning horizon (T_H) and the replanning frequency (specified by an interval $[t_a, t_b]$). Here we examine the single and correlated effects 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. The short interval is chosen to ensure that the results primarily depend on the length of the horizon.

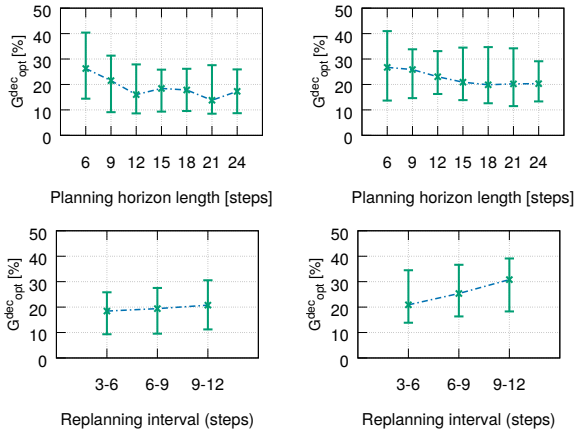

Figure 5: Effect of the lookahead planning horizon (top) and the replanning frequency (bottom) on the decentralized scheme with full (left), and constrained communication (right).

Figure 5 (top) shows the effect of increasing the horizon from 6 up to 24 steps. We indicate the median value and 25% and 75% error bars among the benchmark instances, for both the communication-constrained (top right) and unconstrained scenarios (top 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 is made by [10], although they do not provide any explanation of it.

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, the use of longer horizons only provide a marginally better performance.

Discussion: One possible cause for the behavior described above is the fact that, for longer horizons, the corresponding MIPs become harder to solve. However, after an examination of the distribution of the MIP gap values over the increasing horizon length for all the MIPs solved during the simulations ($\sim 70,000$), we noticed that the increase of complexity was not significant. In fact, most of the MIP solutions remain within 2 % from the optimal solution. We conjecture that this behavior may be due to the asynchronicity of the online schemes and the effect of incomplete information regarding the planned actions of other agents.

Next, we evaluate the impact of the replanning frequency. We consider three replanning intervals, $[3, 6]$, $[6, 9]$, and $[9, 12]$. The interval is set at the beginning of the missions and the agents decide, randomly, when to trigger the next replanning stage based on the 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.

Figures 5 (bottom) 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.

7. CONCLUSIONS AND FUTURE WORK

Building on previous research, we present a decentralized solution to the spatial task allocation and scheduling problem with heterogeneous mobile teams. The work was motivated by the computation and communication constraints that the real-world imposes on a centralized approach. In the process of developing a decision support system for search and rescue missions using multiple heterogeneous robots, we directly faced the challenges that limit the use of a centralized system. These are mainly due to the lack of a network infrastructure (which is not usually present in these scenarios) and the need for rapid recomputation of plans to adapt to the dynamic aspects of the mission.

We have presented a recipe for the top-down decentralization of a mathematical programming model for the coordinated planning of multi-robot systems. The recipe aims at minimizing the computational and communication costs for the single agents while, at the same time, minimizing the loss of coordination due to the decentralization. The experimental results from extensive simulations have shown that we could balance the trade off between computation vs. coordination with the decentralized system in general performing within 10-30% with respect to a centralized implementation.

We also performed an empirical sensitivity analysis of the decentralized system since in general its performance depends on the interplay of a number of aspects such as team diversity, amount of information exchanged, and parameters regulating the computation of plans. Unfortunately, these aspects have correlated effects in presence of multiple agent interactions which are difficult to predict. Our study reveals some counter-intuitive effects and suggests that care should be adopted when designing a distributed/decentralized system. A number of practical lessons have been learned and that could be applied to similar problems and/or to other decentralized architectures derived using the same top-down recipe. In the future we will integrate the empirical results with formal tools so as to perform a sensitivity analysis in a more systematic and general way. We will also soon test our results with real robots in the context of search and rescue scenarios. Preliminary results in this context confirm that the system works but more systematic and comprehensive experiments are needed.

Acknowledgments

This research was supported by the Swiss National Science Foundation (SNSF) through the National Centre of Competence in Research (NCCR) Robotics (www.nccr-robotics.ch) and the Sinergia project SWARMIX, project number CRSI22_133059.

REFERENCES

[1] M. Alighanbari and J. How. Decentralized Task Assignment for Unmanned Aerial Vehicles. In *IEEE Conference on Decision and Control*, pages 5668–5673, 2005.

[2] C. Amato, G. Konidaris, G. Cruz, C. A. Maynor, J. P. How, and L. P. Kaelbling. Planning for decentralized control of multiple robots under uncertainty. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1241–1248, 2015.

[3] F. Amigoni, N. Basilico, and A. Quattrini Li. How Much Worth Is Coordination of Mobile Robots for Exploration in Search and Rescue? In *RoboCup 2012: Robot Soccer World Cup XVI*, volume 7500 LNAI, pages 106–117. Springer Berlin Heidelberg, 2013.

[4] M. Anderson and N. Papanikolopoulos. Implicit cooperation strategies for multi-robot search of unknown areas. *Journal of Intelligent and Robotic Systems*, 53(4):381–397, 2008.

[5] M. Bernardine Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proc. of the IEEE*, 94(7):1257–1270, 2006.

[6] N. Bezzo, B. Griffin, P. Cruz, J. Donahue, R. Fierro, and J. Wood. A Cooperative Heterogeneous Mobile Wireless Mechatronic System. *IEEE/ASME Transactions on Mechatronics*, 19(1):20–31, 2014.

[7] A. Brutschy, G. Pini, C. Pinciroli, M. Birattari, and M. Dorigo. Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 28(1):101–125, 2014.

[8] L. Chen and E. Miller-Hooks. Optimal team deployment in urban search and rescue. *Transportation Research Part B: Methodological*, 46(8):984–999, 2012.

[9] H. L. Choi, L. Brunet, and J. P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25(4):912–926, 2009.

[10] D. Claes, F. A. Oliehoek, K. Tuyls, and D. Hennes. Effective Approximations for Multi-Robot Coordination in Spatially Distributed Tasks. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 881–890, 2015.

[11] D. Di Paola, A. Gasparri, D. Naso, G. Ulivi, and F. L. Lewis. Decentralized task sequencing and multiple mission control for heterogeneous robotic networks. In *IEEE International Conference on Robotics and Automation*, pages 4467–4473. IEEE, 2011.

[12] J. W. Durham, R. Carli, P. Frasca, and F. Bullo. Discrete partitioning and coverage control for gossiping robots. *IEEE Transactions on Robotics*, 28(2):364–378, 2012.

[13] E. Feo Flushing, L. M. Gambardella, and G. A. Di Caro. A mathematical programming approach to collaborative missions with heterogeneous teams. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 396–403, 2014.

[14] S. K. Gan, R. Fitch, and S. Sukkarieh. Online decentralized information gathering with spatial-temporal constraints. *Autonomous Robots*, 37(1):1–25, 2014.

[15] B. P. Gerkey and M. J. Matarić. Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, 2002.

[16] G. Ghiani, F. Guerriero, G. Laporte, and R. Musmanno. Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *European Journal of Operational Research*, 151(1):1–11, 2003.

[17] G. A. Hollinger and S. Singh. Multirobot coordination with periodic connectivity: Theory and experiments. *IEEE Transactions on Robotics*, 28(4):967–973, 2012.

[18] E. Jones, B. Browning, M. Dias, B. Argall, M. Veloso, and A. Stentz. Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, pages 570–575, 2006.

[19] G. A. Korsah, A. Stentz, and M. B. Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12):1495–1512, 2013.

[20] T. Lemaire, R. Alami, and S. Lacroix. A distributed tasks allocation scheme in multi-UAV context. *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, 4:3622–3627, 2004.

[21] L. Luo, N. Chakraborty, and K. Sycara. Multi-robot assignment algorithm for tasks with set precedence constraints. In *IEEE International Conference on Robotics and Automation*, pages 2526–2533, 2011.

[22] J. Parker, E. Nunes, J. Godoy, and M. Gini. Exploiting spatial locality and heterogeneity of agents for search and rescue teamwork. *Journal of Field Robotics*, 7, 2015.

[23] S. Ponda, J. Redding, Han-Lim Choi, J. P. How, M. Vavrina, and J. Vian. Decentralized planning for complex missions with dynamic communication constraints. In *Proc. of American Control Conference*, pages 3998–4003. IEEE, 2010.

[24] S. S. Ponda, L. B. Johnson, A. N. Kopeikin, H.-L. Choi, and J. P. How. Distributed Planning Strategies to Ensure Network Connectivity for Dynamic Heterogeneous Teams. *IEEE Journal on Selected Areas in Communications*, 30(5):861–869, 2012.

[25] S. Sariel-Talay, T. R. Balch, and N. Erdogan. A generic framework for distributed multirobot cooperation. *Journal of Intelligent and Robotic Systems*, 63(2):323–358, 2011.

[26] P. M. Shiroma and M. F. M. Campos. CoMutaR: A framework for multi-robot coordination and task allocation. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4817–4824, 2009.

[27] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.

[28] L. Vig and J. A. Adams. Market-Based Multi-robot Coalition Formation. In *Distributed Autonomous Robotic Systems 7*, pages 227–236. Springer, 2006.

[29] F. Wu, S. Zilberstein, and X. Chen. Online planning for multi-agent systems with bounded communication. *Artificial Intelligence*, 175(2):487–511, 2011.

[30] Y. Zhang and L. E. Parker. IQ-ASyMTRe: Forming Executable Coalitions for Tightly Coupled Multirobot Tasks. *IEEE Transactions on Robotics*, 29(2):400–416, 2013.

[31] Y. Zhang, L. E. Parker, and S. Kambhampati. Coalition coordination for tightly coupled multirobot tasks with sensor constraints. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1090–1097, 2014.