Distributed Task Assignment in a Swarm of UAVs

Luigi Bobbio[1*], Jörg Fliege[1] and Antonio Martinez-Sykora[2]

[1]Mathematical Sciences, University of Southampton, University Road, Southampton, SO17 1BJ, United Kingdom.
[2]Business School, University of Southampton, University Road, Southampton, SO17 1BJ, United Kingdom.

*Corresponding author(s). E-mail(s): l.bobbio@soton.ac.uk, luigi.bobbio@hotmail.it;
Contributing authors: j.fliege@soton.ac.uk; a.martinez-sykora@soton.ac.uk;

Abstract

We consider the problem of distributed task assignment in a swarm of Unmanned Aerial Vehicles (UAVs), where heterogeneous agents that can have different capabilities need to work in teams to execute tasks. We consider the case where communication between UAVs is costly or dangerous and should be limited or avoided, while individual UAVs have uncertain and incomplete information at hand and new tasks can appear during the mission time. For this setting, we develop a distributed computing framework that allows for optimal task assignment under quite general conditions. At each time step of the scheme, each UAV can solve a local version of an optimisation problem that encodes the optimal task assignment for all UAVs. This optimisation problem takes the form of a mixed-integer linear programming problem (MILP) that can be solved readily with state-of-the-art solvers. Theoretical results ensuring the soundness of the proposed approach are reported and numerical results showing its efficacy are investigated.

Keywords: Distributed, Optimal Task Assignment, UAV, Swarm, MIP, MILP

TABLES of SET, PARAMETERS, VARIABLES

Table 1 Overview of the sets used in the model.

Symbol	Index	Definition
$\{1,\dots,n\}$	i	Set of UAVs
$\{1,\dots,m\}$	j	Set of Tasks
$\mathcal{L}(j)$	-	Set of teams that can be allocated to task j
$\Lambda_{\cdot}(j)$	l	Team of UAVs that can be allocated to task j
$\{1,\dots,\Lambda\}$	λ	Set of all Teams, with repetition

Table 2 Overview of the parameters used in the model.

Symbol	Domain	Definition
p_j	$\mathbb{R}^+$	Priority weight for task j
l_j	$\mathbb{R}$	Minimum start time for task j
d_j	$\mathbb{R}^+$	Duration of task j
u_j	$\mathbb{R}$	Maximum completion time for task j
$f_{j,k}$	$\mathbb{R}^+$	Flight time distance between task j and k
L_i	$\mathbb{R}$	Maximum number of tasks that can be assigned to UAV i
$M_{\cdot}$	$\mathbb{R}^+$	A big number

Table 3 Overview of the variables used in the model.

Symbol	Domain	Definition
s_j	$\mathbb{R}^+$	Lateness of task j
t_j	$\mathbb{R}$	Starting execution time for task j
$x_{\lambda,j}$	$\{0,1\}$	Selection of team λ for task j
$y_{j,k}$	$\{0,1\}$	Task k is performed before task j

1 Introduction

Autonomous systems are becoming more and more prominent nowadays given their manifold applications — transportation, surveillance and security, warehouse automation, pickup and delivery, and defence. In particular, their adaptability and their importance in relieving humans from performing dangerous, exhausting or tedious duties has been noted. Design and development of multi-agent systems is a research area where the evolution of control, communication, and decision support systems for UAVs constitutes a rapidly evolving field, as suggested by the US Department of Defense UAV Roadmap 2002-2027 (Department of Defense, UAV Roadmap 2002-2027). Deployment of teams of cooperating UAVs to complete various mission is of primary importance for many mission types and allocation of tasks with constraints constitute an essential aspect to consider when advantageously adopting swarms of

UAVs. The task allocation problem consists in determining how to efficiently assign individual tasks to (teams of) UAVs, accounting for their capabilities, different constraints and overall objectives. By effectively distributing the tasks, it is possible to enhance performances, increase resilience and ultimately improve mission success rates. However, various challenges are associated with task allocation problems. First, heterogeneous capabilities of the UAVs, which may have different sensing, communication or payload capacities, should be considered. Second, the allocation should account for dynamic changes in the environment, such as real-time updates and uncertainties or unexpected events. Furthermore, task allocation might take into account factors such as resource constraints (e.g., battery life, communication bandwidth), inter-UAV communication and coordination as well as task priorities. The task allocation problem becomes harder when the number of tasks, the swarm size and timing constraints grows, as the number of possible task-allocation combinations grows exponentially whereas time resources usually remain limited. To address these challenges, researchers have explored various algorithms and approaches, ranging from exact to heuristic methods and including centralized methods, distributed approaches as well as hybrid strategies. We will provide an overview of these in the following section. In this article, we study a task assignment problem where multiple UAVs, able to perform a single task at a time, are required to work at the same time and in teams on tasks that are scheduled over a planning horizon, whilst respecting synchronisation as well as temporal and ordering constraints. We also consider the case of costly or dangerous communication among the UAV fleet, leading us to examine the problem in the context of distributed optimisation. For this problem, we develop a distributed optimisation framework able to deal with different sources of uncertainties simultaneously and enabling us to analyse situations where these uncertainties or incomplete knowledge are dealt with gradually and not all UAVs receive new information at the same time, or in other words where neither perfect knowledge nor perfect ignorance is assumed.

The paper is then organised as follows. In Section 2 we review the recent developments in tackling the task assignment problem under consideration; in Section 3 we present the underlying mathematical model for the optimal task assignment problem and in Section 4 we embed this in a distributed optimisation framework. Subsequently, Section 5 provides some theoretical results essential to guarantee the reliability of the framework, while also enabling the analysis of numerical results. Such results are presented in Section 6, where we show the power of our framework along with its scalability. Finally, in Section 7 we derive a summary of the findings and propose directions for future work.

2 Related Work

Optimal task allocation is a crucial challenge to effectively deploy multi-agent systems like a swarm of UAVs; it involves the optimal assignment of tasks to individual or teams of UAVs based on their capabilities, constraints, mission objectives and many other possible requirements. Gerkey and Mataric (2004) have proposed a broadly

accepted taxonomy for multi-robot task allocation problems, based on the main characteristics of robots, tasks and time:

- Single-task robots vs. multi-task robots: single-task robots can do at most one task at a time, while multi-task robots can work on multiple tasks simultaneously
- Single-robot tasks vs. multi-robot tasks: single-robot tasks require exactly one robot in order to be completed, while multiple robots are needed to complete a multi-robot task
- Instantaneous vs. time-extended assignments: in instantaneous assignments, tasks are allocated as they arrive, while in time-extended assignments, tasks are scheduled over a planning horizon

In addition, Nunes et al. (2017) extended Gerkey and Mataric's framework, expanding the time-extended assignments segment to include temporal and ordering constraints. In light of this, given the broad range of contexts, the development of specialised algorithms is necessary to efficiently assign tasks to cooperative swarms of UAVs (Mataric (1995); Reif and Wang (1999)). Typical approaches involve mathematical models like mixed-integer-linear-programming (MILP), dynamic network flow optimization (NFO) and multiple intelligent agent-based systems (Forsmo et al. (2013); Peng et al. (2012)). Dynamic programming (DP) is also deployed, sometimes exhibiting computational benefits with respects to other approaches (Alighanbari and How, 2005). In any case, solution algorithms and approaches can be categorised according to whether they are *exact* (Passino et al., 2000) or *heuristic* (Parunak et al., 2002) in nature, with a further distinction regarding the use of a *centralized* or a *distributed* approach. In early studies of the optimal task allocation problem, the focus was primarily on exact algorithms embedded within a centralised framework, where a primary entity generates a collaborative plan for the entire swarm of robots (or UAVs). Darrah et al. (2005) proposed a MILP formulation for the multi-agent task assignment problem assuming heterogeneous teams and dynamic target discovery. The authors tackled the problem dynamism by iteratively reformulating and solving the problem with the arrival of new agent or task. Secrest (2001) studied the problem in the context of surveillance missions requiring UAVs to fly from a starting point through defended terrain to targets, and return to a safe destination. The author reduces the problem to a multiple traveling salesman problem (mTSP) allowing the use of consolidated techniques and commercial solvers to find optimal solutions. While exact algorithms have the advantage of ensuring optimality and preserving an accurate representation of the mission, due to the problem complexity (Balas and Padberg (1976); Service and Adams (2011)), they are often subject to significant computational challenges, particularly for large-scale systems. Consequently, heuristic algorithms have emerged. Well-known heuristic methods include genetic algorithms (Shima et al., 2006), Tabu search (O'Rourke et al., 2001), colony optimisation (Chen et al. (2018); Blum (2005)), particle swarm optimization (Wang et al., 2018), among others.

Despite their relatively simple structure, centralised approaches exhibit several

weaknesses (Zhang et al., 2018) such as the requirement of consistent and complete communication between the central entity and each UAV, a high computational demand posed on the central unit, and a weak stability, rendering the entire system vulnerable to a single point of failure. The increased capability microchips and other of onboard resources made work on distributed frameworks feasible, where the mission can continue even when the central decision maker or other UAVs break down. A common way to adopt distributed approaches is via market auctions (Gerkey and Mataric (2002); Oh et al. (2017); Qin et al. (2022); Gudmundsson et al. (2023)), where UAVs place their bids on each task and negotiate to win it. A consensus-based bundle algorithm (CBBA) has been introduced by Choi et al. (2009), subsequently extended by Ponda et al. (2010) and Whitten et al. (2011). As an alternative, Whitbrook et al. (2015) and Zhao et al. (2018) proposed a distributed approach known as performance impact, directly optimising the overall mission. Exploiting heuristic optimisation principles, performance impact can handle time sensitive multi-agent task allocation problems achieving the allocation of duties with reduced time costs with respect to CBBA. Cui et al. (2022) suggested a linear programming formulation for a single-robot single-task time-extended problem with time window constraints and proposed to use a heuristic and a distributed optimisation algorithm based on the decentralized performance impact.

In spite of all the efforts made, realistic scenarios introduce numerous constraints and complications that substantially increase the complexity of the multi-agent optimal task allocation problem (Geng et al., 2014). In particular, the uncertainty and deadlock problem are the main contributors in limiting the effectiveness, optimality, and practicability of the various approaches. The deadlock problem is the result of two or more vehicles mutually waiting for the other to complete their tasks and might lead to an endless waiting situation that renders the solution ineffective or even unreachable (Coffman et al., 1971). Different methods are proposed to handle this complication (Lemaire et al., 2004), although detecting and unlocking deadlocks is another NP-hard problem (Masticola, 1993). The presence of uncertainty during the task assignment process equally necessitates careful consideration. Primary sources of uncertainty involve the unpredictability of the environment (e.g., the number of tasks, their locations and duration), communication stability (Mazdin and Rinner, 2021), as well as uncertainties associated with the UAVs themselves. Several methods and algorithms have been studied, from stochastic programming to robust optimization (Lu, 2015), all of which can be divided into forethought and postprocessing methods. The former deal with uncertainty at the modelling stage, whereas the latter tackle uncertainty at a later stage, typically in the task execution phase. Postprocessing methods are effective in handling online uncertainties like communication problems as well as new target problems (Evers et al. (2014); Tang et al. (1994)). Although postprocessing typically relies on distributed algorithms to consider real-time requirements, replanning method based on centralised algorithms can also be employed. Recently, machine learning techniques also start to emerge (Shyalika et al., 2020). Nevertheless, even though outstanding advancements have been reached, the literature often assumes either perfect knowledge or perfect ignorance (i.e. for instance if a new task or a failure occurs, all active agents are promptly made aware of it or

all of them remain oblivious) and the uncertainty elements are frequently treated separately or one at a time. We refer the interested reader to Nunes et al. (2017), Seenu et al. (2020) and Poudel and Moh (2022) for a more comprehensive review.

As a final consideration, task assignment problems have also been studied in alternative contexts, where (teams of) both humans and/or machines can be deployed to perform different duties. As a flavour, we mention the work of Younas et al. (2018), who examined a problem in which each task is assigned to a group of collaborating agents and a genetic algorithm is used to solve it at scale, while a comparison with other heuristic methods is also provided. Di Martinelly and Meskens (2017) examined the problem of building surgical teams with different capabilities and nurse schedule rosters, proposing an ε-constraint method to solve it when multiple objective have to be accounted. Van Den Eeckhout et al. (2019) proposed a heuristic algorithm to solve a strategic problem that simultaneously decides on the project schedule and the personnel budget. As a final illustration, Saber and Ranjbar (2022) describe their problem as a mixed-integer programming model and develop a multi-objective decomposition-based heuristic to solve a permutation flow shop scheduling problem with the objectives of minimising the the total tardiness of tasks as well as the total carbon emissions. However, even though multi-robot task allocation problems exhibit characteristics similar to typical task assignment problems, to the best of our knowledge only a few authors have considered decentralised approaches or a dynamic environment for task assignment problems. This has already been highlighted by Niknafs et al. (2013) and Fikar and Hirsch (2017), and it appears that no attention has been paid to problems considering heterogeneous team work, decentralisation and uncertainties simultaneously.

With all this in mind, in this article we consider a distributed framework for a problem of assigning jobs to an heterogeneous swarm of UAVs that work simultaneously in teams (multi-robot) on tasks (single-task) planned and executed over a certain period of time (time-extended). We include temporal and ordering constraints, while taking into account uncertain environments where neither perfect knowledge nor perfect ignorance is assumed. In addition, in the following sections we tackle the problem by building a distributed framework free of deadlock situations under fairly mild conditions.

3 The Optimal Task Assignment Problem

3.1 Problem description

The problem considered in this paper regards the study of a multi-robot single-task time-extended task assignment problem, including temporal and ordering constraints. Multiple UAVs, able to perform a single task at a time, are required to work at the same time and in teams to complete tasks positioned in different known (or estimated) locations, scheduled over a planning horizon and with the objective of performing them within a specified time window or with minimum delay. We consider the case of costly or dangerous communication among the fleet, leading us to examine the

Table 4 Example list of required capabilities.

task	capability
1	x
2	y
3	x and y

problem in the context of distributed optimisation. At first, we provide a centralised mathematical optimisation model, phrasing this multi-UAV task assignment problem as a mixed-integer linear programming problem (MILP) that includes realistic characteristics of the mission as well as sufficient fidelity to tackle key questions stemming from real-world applications. Then, in the following sections, we see how to embed the above formulated problem into a distributed optimisation framework enabling us to simultaneously deal with different sources of uncertainties, like execution times or the arrival of new tasks in due course, and analyse scenarios with incomplete knowledge. We examine situations where not all UAVs receive new information at the same time or, as previously stated, neither perfect knowledge nor perfect ignorance is assumed. Theoretical results are provided that ensure the efficacy and convergence of our method. Finally, numerical results are also discussed, showing the efficacy of the proposed approach.

3.2 Mathematical Model for the Optimal Task Assignment Problem

Let m tasks and n UAVs be given, with $m, n \in \mathbb{N}$. Considering situations that occur naturally when different tasks require different capabilities, we allow for various possible teams of UAVs to be formed and be allocated to a particular task. If we have given the capabilities of each UAV as well as the capabilities needed to perform each task, we can compute in a preprocessing step the following information: for each task j we compute the set $\mathcal{L}(j) = \{\Lambda_1(j), \dots, \Lambda_{k(j)}(j)\}$ with each $\Lambda_\ell(j) \subseteq \{1, \dots, n\}$, representing the set of all UAV teams that can be allocated to task j. We include the singleton set $\{i\}$ in $\mathcal{L}(j)$, if UAV i can be allocated to task j alone. As a simple example, consider three tasks $j = 1, 2, 3$ with the following necessities: task 1 needs capability x to complete, task 2 needs capability y to complete, and task 3 needs capability x and y to complete as shown in Table 4. We have also four UAVs, whose capabilities are shown in Table 5. Thus, we can derive the sets of possible formations $\mathcal{L}(\cdot)$ as shown in Table 6. For instance, the set of teams that can be assigned to task 3 is described by $\mathcal{L}(3) = \{\Lambda_1(3), \Lambda_2(3)\}$ where $\Lambda_1(3) = \{3\}$ and $\Lambda_2(3) = \{1, 4\}$.

In what follows, we number all those teams $\Lambda_\ell(j)$ occuring in the above consecutively through $\lambda = 1, \dots, \Lambda$, including repetitions, and we use the notation $\Lambda_\ell(j) = \#\lambda$ if the team of UAVs $\Lambda_\ell(j)$ is numbered as the λ-th set. Continuing on the previous example, Table 7 provide an illustration of such numbering. With this in mind, to

Table 5 Example list of UAV capabilities.

UAV	capability
1	x and z
2	z
3	x and y
4	y

Table 6 Example list of tasks and sets of teams that can be assigned to those tasks.

task	set of teams $\mathcal{L}(\cdot)$
1	$\{1\}, \{3\}$
2	$\{3\}, \{4\}$
3	$\{3\}, \{1, 4\}$

Table 7 Example list of numbered teams and their component UAVs.

λ	1	2	3	4	5	6
team	$\{1\}$	$\{3\}$	$\{3\}$	$\{4\}$	$\{3\}$	$\{1, 4\}$
$\Lambda_\ell(j)$	$\Lambda_1(1)$	$\Lambda_2(1)$	$\Lambda_1(2)$	$\Lambda_2(2)$	$\Lambda_1(3)$	$\Lambda_2(3)$

select teams' allocation, it is then natural to introduce the decision variables

$$x_{\lambda,j} = \begin{cases} 1 & \text{if team } \lambda \text{ is allocated to task } j, \\ 0 & \text{otherwise.} \end{cases}$$

Suppose now that we have given, for each task j, a lower bound ℓ_j and upper bound u_j, representing the minimum time at which the task can start to be performed and the maximum required time of completion respectively, as well as (an estimate of) the time d_j necessary to complete task j, with $\ell_j + d_j \leq u_j$ for consistency. In other words, work on task j must not start earlier than time ℓ_j, it will take time $d_j \geq 0$ and they should preferably be finalised before time u_j. To model this situation, we consider variables

$$t_j \geq 0 \qquad j = 1, \dots m$$

indicating the time when task j starts. Variables t_j must then obey the constraints

$$l_j \leq t_j \tag{1}$$

and, to require that task j is completed on time, we do not impose a hard constraint but we rather consider the relevant objective

$$\min \quad \sum_{j=1}^{m} p_j \max\{0, t_j + d_j - u_j\}, \tag{2}$$

where we minimise the sum of each task lateness, $\max\{0, t_j + d_j - u_j\}$, a.k.a. the *total tardiness*, also multiplying each of them by a constant $p_j > 0$, signifying the importance (or priority) of task j. Note that in a similar fashion, it is straightforward to move from hard constraints (1) to soft constraints for working on a task too early, using an additional penalty term in the objective, even if in what follows we do not explore this route. Then, by adding m further artificial variables s_j to the problem, we can linearise the objective function (2) by writing

$$\min \sum_{j=1}^{m} p_j s_j,$$

and adding for each $j = 1, \ldots, m$ the supplementary constraints

$$\begin{aligned} t_j + d_j - u_j &\leq s_j, \\ 0 &\leq s_j. \end{aligned}$$

Acknowledging that we are also asked that each UAV cannot perform two tasks simultaneously, we model the requirement as follows. Consider two teams $\#\lambda$ and $\#\upsilon$, allocated to two tasks j and k respectively, with $j \neq k$. If the two teams have a UAV in common, then we need to ensure that task j is finalised before task k, or vice versa. This can be expressed by the requirement that

$$\text{either} \quad t_j + d_j \leq t_k \quad \text{or} \quad t_k + d_k \leq t_j.$$

Using additional binary variables $y_{j,k} \in \{0, 1\}$, we can write the above inequalities along with the logical constraints via the coupled constraints

$$\begin{aligned} t_j + d_j &\leq t_k + M_1 y_{j,k}, \\ t_k + d_k &\leq t_j + M_1 (1 - y_{j,k}), \end{aligned}$$

with $M_1 > 0$ being some sufficiently large constant. Noting then that these two constraints only need to hold when a UAV i is common between team $\#\lambda$ and $\#\upsilon$, while the two teams are assigned to task j and task k simultaneously (i.e. when $x_{\lambda,j} = x_{\upsilon,k} = 1$), we just need to consider then the adjusted set of constraints

$$\begin{aligned} t_j + d_j &\leq t_k + M_1 y_{j,k} + M_2 (2 - x_{\lambda,j} - x_{\upsilon,k}), \\ t_k + d_k &\leq t_j + M_1 (1 - y_{j,k}) + M_2 (2 - x_{\lambda,j} - x_{\upsilon,k}) \end{aligned}$$

for all $j, k = 1, \ldots, m$ and for all $\lambda, \upsilon = 1, \ldots, \Lambda$ with $\#\lambda \cap \#\upsilon \neq \emptyset$. Analogously to M_1, $M_2 > 0$ is a big number ensuring that the logical implications are satisfied.

At the same time, if estimates of tasks location are available we can incorporate flight times between tasks as follows. Knowing that the different tasks have to be executed at particular locations and denoting by $f_{j,k}$ the flight time from the location

where task j needs to be executed to the location where task k needs to be executed, we can then expand the previous constraints into

$$t_j + d_j + f_{j,k} \leq t_k + M_1 y_{j,k} + M_2(2 - x_{\lambda,j} - x_{\upsilon,k}),$$
$$t_k + d_k + f_{k,j} \leq t_j + M_1(1 - y_{j,k}) + M_2(2 - x_{\lambda,j} - x_{\upsilon,k}),$$

for all teams λ, υ with $\#\lambda \cap \#\upsilon \neq \emptyset$ and $j, k = 1, \ldots, m$.

With this in mind, we are then ready to provide a full model formulation as follows.

OTASF-MCUS: Optimal Task Assignment with Scheduling and Flight Times in a Multi-Capability UAV Swarm

$$\min_{s,t,x,y} \quad \sum_{j=1}^{m} p_j s_j \tag{3}$$

$$\text{subject to} \quad t_j + d_j - u_j \leq s_j \qquad (j = 1, \ldots, m) \tag{4}$$

$$0 \leq s_j \qquad (j = 1, \ldots, m) \tag{5}$$

$$\ell_j \leq t_j \qquad (j = 1, \ldots m) \tag{6}$$

$$t_j + d_j + f_{j,k} \leq t_k + M_1 y_{j,k} + M_2(2 - x_{\lambda,j} - x_{\upsilon,k}) \qquad (\forall \lambda, \upsilon : \#\lambda \cap \#\upsilon \neq \emptyset; j, k = 1, \ldots, m) \tag{7}$$

$$t_k + d_k + f_{k,j} \leq t_j + M_1(1 - y_{j,k}) + M_2(2 - x_{\lambda,j} - x_{\upsilon,k}) \qquad (\forall \lambda, \upsilon : \#\lambda \cap \#\upsilon \neq \emptyset; j, k = 1, \ldots, m), \tag{8}$$

$$\sum_{\lambda=1}^{\Lambda} x_{\lambda,j} \geq 1 \qquad (j = 1, \ldots, m), \tag{9}$$

$$x_{\lambda,j} = 0 \qquad (\forall \lambda \notin \mathcal{L}(j)); \tag{10}$$

$$\sum_{\substack{(\lambda,j): \\ i \in \#\lambda}} x_{\lambda,j} \leq L_i \qquad (i = 1, \ldots, n), \tag{11}$$

$$x_{\lambda,j} \in \{0, 1\} \qquad (\lambda = 1, \ldots, \Lambda; j = 1, \ldots, m), \tag{12}$$

$$y_{j,k} \in \{0, 1\} \qquad (j, k = 1, \ldots, m), \tag{13}$$

In addition to the constraints presented before, we consider constraints (9) indicating that at least one team is assigned to task j, and constraints (10) ensuring that a team is not assigned to a task that it cannot execute. In constraints (11) we sum up over all teams where a particular UAV i belongs, ensuring that the UAV is not overloaded, executing at most L_i tasks. Finally, constraints (12) and (13) represent the obvious requirement of having binary decision variables as stated in their definition.

Having formulated our mathematical model for the optimal task assignment problem, in the following sections we will first see how to embed the model into a distributed framework for decision making whilst ensuring the reliability of the latter. Second, we investigate numerical examples where we show that state-of-the-art solvers can solve the above mathematical model in a reasonable time for problems of realistic size, and enable us to evaluate the effects of uncertainties and incomplete information.

4 Distributed Optimisation

Given a mathematical optimisation problem, the goal of an optimisation algorithm is to compute a (local) minimum of one or multiple objective functions, while respecting

any given constraints. In a distributed optimisation framework we have been given several computational units, often provided with their own local memory and able to deploy optimisation algorithms to solve a given mathematical model using their internal resources. However, communication between these units is usually costly. This seems to be the appropriate context for a fleet of UAVs that have to cooperatively solve a common set of task, a.k.a. the *mission*, especially if distances between UAVs are large or there is a general desire to keep communications limited. In what follows, we describe the proposed structure of our distributed framework, following mostly Bertsekas and Tsitsiklis (1991).

4.1 Generic Distributed Optimisation Framework

Consider to have n UAVs at disposal, and that all parameters of the mathematical programming model described above stem from a set Ω. At each time t of the mission, let UAV i ($i = 1, \ldots, n$) be equipped with its own vector of decision or state variables $x_i(t)$, (estimates of) other UAVs' decision variables $x_1(t), \ldots, x_{i-1}(t), x_{i+1}(t), \ldots, x_n(t)$, a set of data and parameters $P_i(t) \in \Omega$, as well as a *belief update* function $G_i : \Omega^n \to \Omega$. We can think of the vector of variables $x_i(t)$ as the list of the above defined variables $x_{\lambda,j}$ with $i \in \#\lambda$ at a specific time t and it can be interpreted as a list of commands or actions the UAV has to follow, or an approximation thereof. The vector $P_i(t)$ can be thought of as the parameters of the model, representing (an estimate of) the world's status at time t, while the function G_i is a way in which the UAV i updates its knowledge about the world, for example using its onboard sensors. Let UAV i also be equipped with an *update function* F_i taking as argument states $x_1(t), \ldots, x_n(t)$ as well as UAV's belief on the world status $P_i(t)$ and producing a new value for the decision variables x_i. F_i represents the way in which UAV i update its decisions based on the available information and can be thought as a full or partial execution of an optimisation algorithm solving the above model OTASF-MCUS. Collect the current parameter vectors of all UAVs into a vector $P = (P_1, \ldots, P_n)$, where we ignore the time dependency for a moment to lighten the notation. We can then phrase the UAVs mission, and hence the optimal task assignment problem, as a fixed-point problem, where we search for an $x^* = (x_1^*, \ldots, x_n^*)$ such that

$$x^* = F(x^*, P) = (F_1(x^*, P_1), \ldots, F_n(x^*, P_n)) \tag{14}$$

Such an interpretation is always possible for optimisation algorithms that work via improvement steps, i.e. if in some step one is given some x_{old}, the algorithm computes an update step $\Delta(x_{\text{old}})$ and sets

$$x_{\text{new}} := x_{\text{old}} + \Delta(x_{\text{old}}) \tag{15}$$

Often, the update $\Delta(x_{\text{old}})$ is computed in such a way that x_{new} is an improvement on x_{old} either in terms of objective function value or feasibility, or both. The iterations stop when $\Delta(x) = 0$. If such an algorithm is given, one can simply use the update function $F(x) := x + \Delta(x)$ to define the fixed-point problem $F(x) = x$. In particular, we have that $x_{\text{new}} = F(x_{\text{old}})$.

4.2 Synchronous vs. Total Asynchronous Optimisation

Examining now the time development of the mission, we recall that at time t, each UAV or agent has stored its decisions in the vector $x_i(t)$. If then all agents know the most up to date variables of all other agents and we consider a fixed world's status $P = P(t) = P_1(t) = P_2(t) \cdots = P_n(t)\ \forall t$, each agent can update its own decisions via

$$x_i(t + \Delta t) = F_i(x_1(t), x_2(t), \ldots, x_n(t), P) = F_i(x(t), P) \tag{16}$$

with Δt a small time variation. The above describes a situation where all agents use the same information (parameters) and revise their decisions according to (16). This is referred to as the *globally synchronous* update. In this case, if the functions F_i are chosen well, then iterating over time t will result in better and better approximations $x_i(t)$ of a solution x^*.

On the other hand, often communications between UAVs are costly or dangerous, causing UAVs not to have the same and latest information available at time t, and therefore making it impossible to update the local decisions according to (16). We will call such an update *asynchronous*, if there exists at least one time index $\hat{t}$ with

$$x_i(\hat{t} + \Delta\hat{t}) = x_i(\hat{t}) \tag{17}$$

that is if F_i is not evaluated and the old value of x_i is just copied over to the next time step. Typical examples for such a framework in the area of parallel processing are Gauss-Seidel vs Jacobi iterations to solve linear systems. Defining then *delay indices* $\tau_{i,k}(t)$ such that

$$0 \leq \tau_{i,k}(t) \leq t$$

we compute the iterates in the so called *total asynchronous* case by

$$x_i(t + \Delta t) = F_i(x_1(\tau_{i,1}(t)), x_2(\tau_{i,2}(t)), \ldots, x_n(\tau_{i,n}(t), P) \tag{18}$$

Each delay index $\tau_{i,k}(t)$ can be seen as how up-to-date UAV i is with respect tp the beliefs and decisions of UAV k at time t. Without further knowledge on the $\tau_{i,k}(t)$ (i.e. how much "lag" there is in the system) this is, in a sense, the most general case and likewise the worst case. As it turns out, even in this worst case distributed optimisation can be made to work under relatively mild conditions.

4.3 Static vs. Dynamic Case

As mentioned before, each F_i takes as input also a belief of the overall status of the mission environment $P(t)$, which can also include an estimate for the degree of uncertainty present. However, up to this point we considered only the *static case*, where $P = P(t)$ is given and constant throughout the mission, and we search for a fixed point $x^* = F(x^*, P)$. However, a more interesting case is the *dynamic* one, where some problem data either changes over time, or are unknown at the start of the mission, or only estimates of them are available and UAVs can not broadcast these

changes immediately. Potential changes in the problem data necessitate that, at each time step t, each UAV i stores not only its own optimal decision variables $x_i(t)$, but also its own version of the state of the problem, $P_i(t)$. It is worth noting here that it is possible to extend the notation used for the synchronous and asynchronous updates to the case where agents' decisions depend explicit on the available world's information at time t. More precisely, we can write $P_i(\tau_{i,p}(t))$ to account for delays in UAV i believes about the world's status at time t and utilise the update rule

$$x_i(t + \Delta t) = F_i(x(\tau_i(t)), P_i(\tau_{i,p}(t)))$$

with $\tau_i(t) = (\tau_{i,1}(t), \ldots, \tau_{i,n}(t))$. The synchronous update is simply obtained when $\tau_{i,p}(t) = t \ \forall i$.

With this in mind, *decision variables and problem data are both updated during the course of the optimisation algorithm* according to

$$P_i(t + \Delta t) = G_i(P_1(\tau_{i,1}(t)), \ldots, P_n(\tau_{i,n}(t))) \tag{19}$$
$$x_i(t + \Delta t) = F_i(x(\tau_i(t)), P_i(t)) \tag{20}$$

and UAVs communicate not only their current decision variables to help the optimisation process, but also their current belief of what problem needs to be solved.

In light of this, considering a discrete time environment (i.e. $t \in \mathbb{Z}$ and $\Delta t = 1$), it is possible to build a *distributed optimisation framework* where our overall computational model requires each UAV to execute the following steps of what we will call the *UAV Local Control Loop*:

Algorithm (UAV Local Control Loop)

1. Update UAV's belief $P_i(t + 1)$ according to (19)
2. Compute decision variables $x_i(t + 1)$ by evaluating (20)
3. [*Optional*] If possible, execute some of the decisions encoded in $x_i(t + 1)$
4. If allowed, communicate $x_i(t + 1)$, $P_i(t + 1)$ to other (neighbouring) UAVs
5. Also if allowed, receive data x_j and P_j from some other UAV $j \neq i$
6. $t := t + 1$
7. Go to step 1

5 Theoretical Results

Within the context of distributed optimisation and having equipped each UAV with a general decision framework (its local control loop), we would like to guarantee that we are able to solve our optimal task assignment problem under quite mild conditions. The following results will help us to show that we are indeed able to ensure this. Let T_i be the set of time indices with $\tau_{i,k}(t) = t \ \forall k = 1, \ldots, n$. In other words, T_i will be the set of time indices where UAV i has access to the most up-to-date information. In

addition, let P be the fixed status of the world. Then we have the following main result

Theorem 1. *Assume that the synchronous fixed-point iteration $x(t+1) = F(x(t), P)$ converges to a fixed point of F from all starting points. Assume further that all sets T_i are infinite and that for all i and all sequences $(t_\ell)_\ell$ in T_i we have*

$$\lim_{\ell \to \infty} \tau_{j,k}(t_\ell) = \infty$$

for all j and k. Then the totally asynchronous iteration above converges to a fixed point of F.

Proof. see Bertsekas (1983) □

In other words, *any correct optimisation algorithm can be cast in a fully distributed form and executed totally asynchronously*, as long as each UAV updates it's own status infinitely often, and updates from other UAVs never stop, no matter how much "lag" these communications have. Of course, one has to expect to suffer a breakdown in convergence speed if state updates are not provided often enough. The above comment is also reflected in the following results

Corollary 1. *Consider the graph with UAVs as nodes and an edge connecting two nodes if and only if the two UAVs can communicate with each other. Assume that this graph is connected. In each time step t, let each UAV i communicate its variables $x_i(t)$ to all UAVs that are its neighbours in this graph. Then the distributed fixed-point iteration converges.*

Proof. see Bertsekas (1983) □

Corollary 2. *The corollary above holds if communication is only allowed every k optimisation steps.*

Proof. see Bertsekas (1983) □

We have shown here that in the static case and for both the globally synchronous and totally asynchronous iteration, we are able to find a solution for the optimal task assignment problem using the above UAV Local Control Loop, provided that information is updated in a somewhat regular manner. Encouraging results are also valid for the dynamic case, where we are still able to find an optimal task allocation, if the world status $P(t)$ does not change dramatically and too frequently.

Corollary 3. *Suppose that the actual problem data $P(t)$ converges, i. e. $\lim_{t\to\infty} P(t) = P^*$ for some P^*. Write the fixed-point equation under consideration as explicitly depending on the problem data, i.e. $x = F(x, P)$, and assume that for each P there is a unique fixed point $x(P)$ of this equation. Assume that the mapping $P \mapsto x(P)$ is continuous at P^*, or that there are only finitely many changes to $P(t)$. Then the distributed fixed-point iteration converges under the same assumptions as those stated in Theorem 1.*

Proof. see Bertsekas (1983) □

In practice, the problem data $P_i(t)$ might correspond to a message of rather large size, i.e. when $P_i(t)$ is a large dense matrix of floating point numbers. In a reasonably stable environment it is then often better to communicate only updates of the problem data, i.e. the UAVs send $\Delta P_i(t) := P_i(t) - P_i(t-1)$, in particular if this data is sparse.

Having shown the potential of finding optimal tasks assignments within the distributed framework above, we proceed now by showing numerical evidences demonstrating potential effects of dynamic data as well as asynchronous communications.

6 Numerical Results

In this section we study instances of the computational framework outlined previously by numerically investigating the properties of some particular mission example. We establish a simulation workflow that uses (partly) randomised problem data, allowing us to provide sufficiently powerful statistics to draw upon. All the reported computations have been performed on a Viglen Intel Core 2 with 16GB RAM with a Windows 10 operating system and our simulation prototype has been coded in MATLAB R2021b. The optimisation model we considered has been coded in AMPL (Fourer et al., 2003), and for solving these problems we have accessed the high-performance optimisation solver CPLEX (IBM CPLEX) Version 20.1 via the corresponding AMPL interface.

6.1 Preliminaries

Here we provide a simulation tool where UAVs perform of all the steps of their Local Control Loop (including the execution of tasks when possible) to achieve the best allocation and to complete the required tasks. To do so, we implement a simulation framework where we have a "wall clock", know to each UAV and set to $t_0 = 0$, the time when the mission starts. At this time, each UAV is provided with (partial) information about the world and start running its own local control loop. In particular, given that each UAV starts with the same information, no communication is needed so they will start running the optimisation model with the information at hand, obtaining a first task allocation. We can consider this moment as the mission kick-off and UAVs start flying towards the first task they are assigned to. From now on, only the following relevant events can happen

1. *UAV arrival*: at least one UAV reaches a task location
2. *Task completion*: at least one task has been completed
3. *UAV communication*: newer information can circulate
4. *New task incoming / nature update*: the world's status changes and (some) UAVs becomes aware of it
5. *Mission completed*: all tasks have been completed

where the events should not be considered as mutually exclusive. In all these cases, one or more steps of the local control loop are required to be performed. In all other times, executing any step of the local control loop is not necessary, since it would

lead to the same decisions of the previous step. A schematic representation of the described workflow is reported in Figure 1. Depending on the event occuring, different actions, always part of the control loop, are then required. More precisely, in case of

1. *UAV arrival*: arrived UAVs check if the working team they are supposed to work with is at task's location and, in that case the team starts working on the assigned task. Otherwise, UAVs are tasked to update their world status and if possible, communicate any updates;
2. *Task completion*: the working team of UAVs communicate the information about the task termination, if presently allowed;
3. *UAV communication*: UAVs have just been allowed to communicate and they are asked to inform the neighbour UAVs about newer information;
4. *New task coming / nature update*: some UAVs are made aware of changes in the world status and are tasked to communicate the information, if allowed.

In addition to the above, if at any relevant time a UAV is in possess of newer information, it can solve its local optimisation model using the newer data and amend its own actions accordingly.

This framework provides us with the opportunity to *dynamically change problem data during the mission* and therefore to analyse more realistic situations. To provide a better understanding of the incomplete information implications, we will consider here only the case of *new tasks occurring during mission time*. New tasks occurring can be seen as special instances of changes in starting time ℓ_j and duration d_j. Namely, in our framework, we can consider each potentially occurring task as an existing task j right from the start, with $\ell_j = 0$ and $d_j = 0$. These 'dummy tasks' are then assigned to some team by the first optimisation step. When their data is updated later (i.e. when the actual new tasks come into existence), a new optimisation run can be performed by all UAVs aware of change in data.

With all this in mind, as a numerical example, we will consider a case with $n = 10$ UAVs and $m = 15$ tasks. We represent the fact that we have costly or limited communication capacity using a communication graph as depicted depicted in Figure 2. Each UAV is represented as a node in the graph and, whenever a communication event occur, it is allowed to send information only to the neighbour UAVs (i.e. the adjacent UAVs directly connected to it). The 10 UAVs can form 29 different teams and these, together with their possible allocations to tasks, are listed in Table 8.

Further problem parameters are fixed as follows. The maximum number of tasks a UAV can be assigned to is given in Table 9. The earliest starting times ℓ_j for the 15 tasks as well as their duration d_j are provided in Table 10. Travel times between task locations are furnished in Table 11. Finally, we define termination times u_j for all tasks j as

$$u_j := \hat{u} := 2808 \qquad \forall j \in \{1, \dots, m\} \tag{21}$$

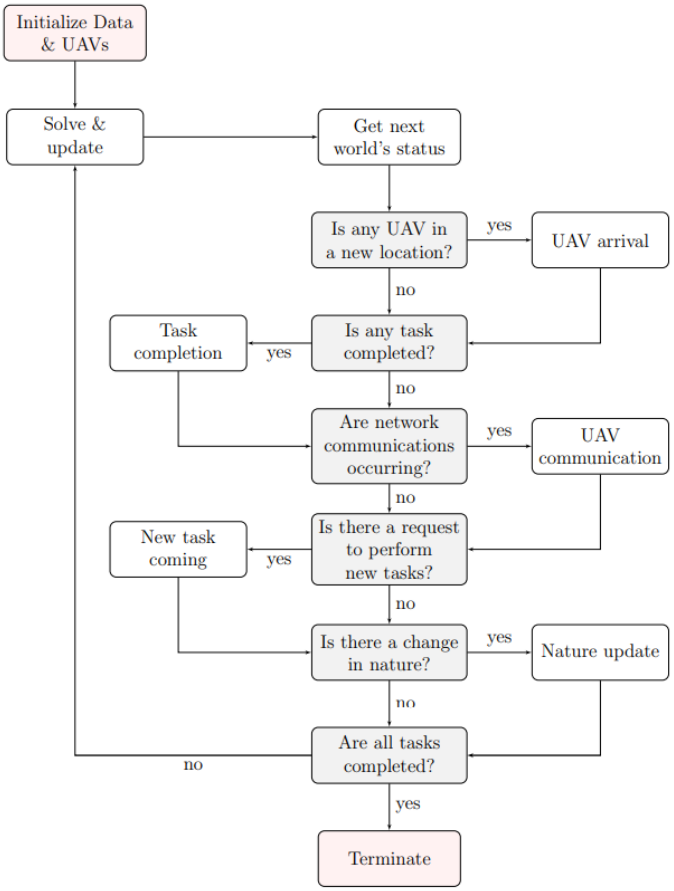

Fig. 1 Simulation Workflow

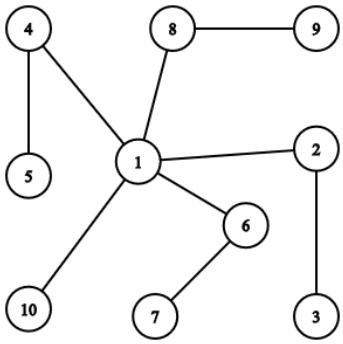

Fig. 2 Structure of the communication network with 10 UAVs.

Table 8 Teams composition and possible allocation of teams to tasks. For each task j, the "Team" column furnishes the list of teams that can work on it, previously denoted with $\Lambda_\ell(j)$. Finally, teams are numbered by λ according to the notation used for the OTASF-MCUS model.

task	Team	$\Lambda_\ell(j)$	λ
1	{1,2,5}	$\Lambda_1(1)$	1
1	{3,4}	$\Lambda_2(1)$	2
1	{1,4,5}	$\Lambda_3(1)$	3
2	{6,9}	$\Lambda_1(2)$	4
3	{4,2,3}	$\Lambda_1(3)$	5
3	{2,4,9}	$\Lambda_2(3)$	6
4	{2,4,7}	$\Lambda_1(4)$	7
4	{2,8}	$\Lambda_2(4)$	8
4	{9,10}	$\Lambda_3(4)$	9
5	{5}	$\Lambda_1(5)$	10
6	{1,9}	$\Lambda_1(6)$	11
6	{2,4,8}	$\Lambda_2(6)$	12
7	{5,8}	$\Lambda_1(7)$	13
7	{1}	$\Lambda_2(7)$	14
7	{1,2,3}	$\Lambda_3(7)$	15
8	{2,6}	$\Lambda_1(8)$	16
9	{10}	$\Lambda_1(9)$	17
10	{6,5,7}	$\Lambda_1(10)$	18
10	{1,2,3}	$\Lambda_2(10)$	19
10	{4,5}	$\Lambda_3(10)$	20
11	{2,4,7}	$\Lambda_1(11)$	21
11	{5,9,10}	$\Lambda_2(11)$	22
11	{2,3,4}	$\Lambda_3(11)$	23
12	{2,3,5}	$\Lambda_1(12)$	24
12	{4,6}	$\Lambda_2(12)$	25
13	{1,6,8}	$\Lambda_1(13)$	26
13	{6}	$\Lambda_2(13)$	27
14	{1,6,10}	$\Lambda_1(14)$	28
15	{5,8}	$\Lambda_1(15)$	29

Table 9 Maximum number of tasks L_i that can be execute by each UAV i.

UAV i	1	2	3	4	5	6	7	8	9	10
L_i	9	13	10	8	7	9	13	6	9	12

Table 10 Earliest starting time l_j and duration d_j for each task j, in seconds.

task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ℓ_j	2080	4244	1887	859	3040	2995	1574	3854	4798	4709	610	1115	113	2916	532
d_j	886	827	331	506	322	577	738	691	117	761	130	384	997	926	341

and all the $p_j = 1$, or equivalently stated, we have been given a maximum mission time $\hat{u}$ to fulfill all tasks and the same priority level to every tasks.

With all this in mind, by comparing three different cases, we provide useful insight on the distributed optimal task assignment problem, while at the same time showing the efficacy of the proposed approach. The case examined are the following.

- Static Case: where all the information about the world status are known in advance by all UAV;
- Dynamic Case I: where some new tasks appear during the mission and only one UAV is made aware of them;

Table 11 Travel time distances between tasks, in seconds.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0														
2	616	0													
3	169	177	0												
4	219	283	409	0											
5	253	338	427	347	0										
6	219	203	229	293	628	0									
7	439	286	515	320	405	384	0								
8	338	431	364	293	373	438	277	0							
9	439	409	461	334	262	447	223	375	0						
10	370	209	271	156	275	189	358	313	445	0					
11	540	98	273	311	620	432	446	567	201	301	0				
12	293	339	213	336	430	108	293	457	405	378	478	0			
13	138	452	328	103	535	278	589	231	543	313	18	186	0		
14	101	221	373	587	627	550	316	290	338	256	463	361	166	0	
15	478	220	249	300	331	237	250	203	241	575	403	159	260	560	0

- Dynamic Case II: where the same new tasks as in Dynamic Case I appear during the mission, but further UAVs are made aware of them.

6.2 Static Case

Let us first consider a *benchmark case* in which the status of the world P is constant and all UAVs are provided with all the information described in the preliminaries since the beginning of the mission. In this situation, optimal task allocation can be reached by centrally solving the instance of the OTASF-MCUS problem and inform all UAVs at once about the initial plan that will then be executed. An execution of the optimal control loop is then redundant and only one communication is needed. We describe here the main characteristics of the solution computed, in order to compare these with those of the dynamic cases. Figure 3 provides an overview over the mission timeline. The horizontal axis represents mission time, and all 15 tasks are plotted along the vertical axis. If a task is allocated to a particular team at a certain point in time, the colour green is used. As it is natural, all tasks are in green right from the start of the mission and expected to be allocated to exactly one team of UAVs. In fact, in this benchmark case all information on all tasks is available to all UAVs from the mission start. Blue dots and lines represent when the designated team reaches a task location, works on it and complete the job. The vertical red line corresponds to the envisioned maximum mission time $\hat{u}$. As it can be seen, this benchmark problem is actually quite a difficult one, from a mission planning perspective. We have been given 15 tasks, but we can complete at best only 7 of them on time (these are tasks no. 3, 4, 7, 11, 12, 13, 15). The remaining tasks are tackled only with delay.

We can also look at how tasks terminate over time, and how the UAV swarm retains a collective interpretation as to how many tasks are left and late. Figure 4 provides us with a corresponding timeline, with the blue graph showing the total number of completed tasks over time, exhibiting the expected staircase pattern over time, and the red curve showing the number of known tasks that have not been completed and

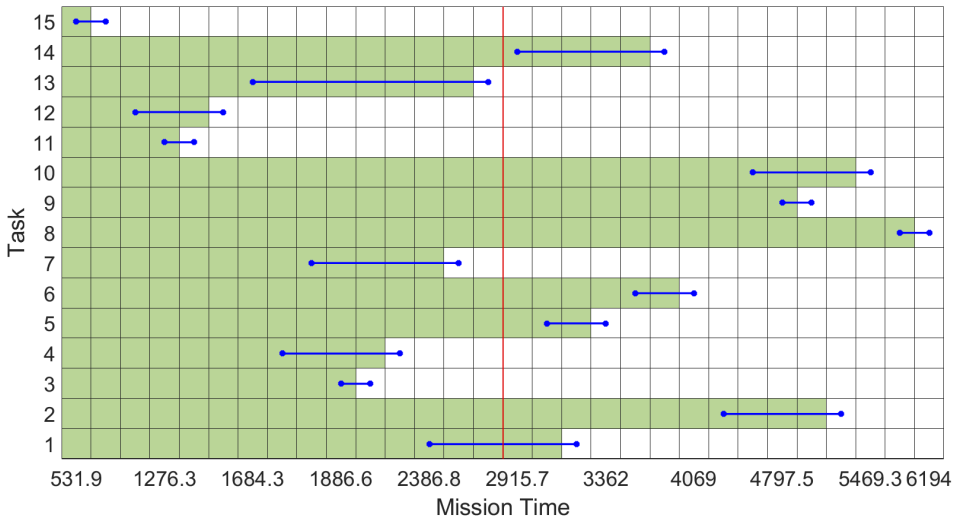

Fig. 3 Static Case representation. Along the Mission Time (x-axis), for each task (reported on the y-axis) the green colour indicates that a whole team is agreeing to work on it at some point of the mission, with blue dots and lines depicting the actual time in which the corresponding task is executed. A vertical red line represents the maximum mission time where tasks completed on its right are deemed to be late.

that are deemed as late by the swarm of UAVs. One notices of course that the number of late tasks is known from the beginning (a constant red line until $\hat{u}$) and the resulting red stair curve after $\hat{u}$ is entirely symmetric to the blue one. The information contained in the graph will be particularly useful compared to the following cases.

6.3 Dynamic Case I

Using the same setting as in the benchmark case, we can examine the circumstances where UAVs have *incomplete information*. In particular, we consider that tasks 11–15 "arrive" during the mission, while all UAVs are already working on the other tasks no. 1–10. We also assume that, as soon as one of the new tasks 11-15 becomes available, UAV no. 3 only is made aware of its existence and, when allowed, it is made responsible of informing the rest of the swarm about it. Figure 5 provides then an overview of the corresponding mission profile. As before, the green colour indicates that a team has been assigned to perform a task and is expected to execute it at some point in the future. In addition, we use dark green to represent times where more than one team of UAVs (in this case two) has been assigned to a task. As before, blue dots and lines represent times at which the assigned team is working on the given task. Furthermore, black dots indicate when new tasks become known to UAV 3, and light green dots indicate when UAV 3 communicates the corresponding updates to its neighbouring UAVs. As it can be seen, the overall mission time necessary for the completion of all tasks increases by ca. 20%, and new tasks 11–15 are only allocated to teams after the a-priori defined end of mission time $\hat{u}$ (indicated as before by

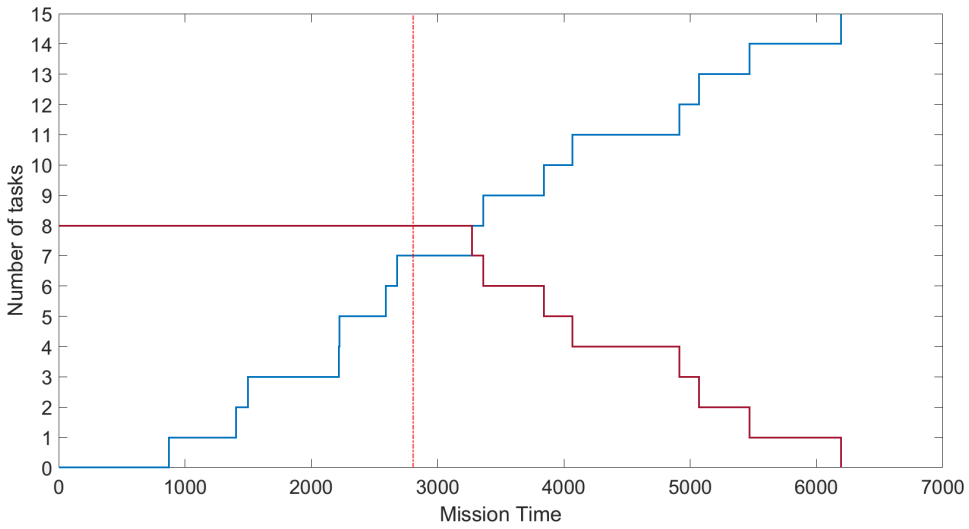

Fig. 4 Expected number of late tasks in the Static Case. Along the Mission Time (x-axis), the number of current task that the swarm (averaged among all UAVs) deems to be late is depicted in burgundy, while a blue line indicates the number of completed tasks up to that point, with tasks completed after the vertical red dashed line, expressing the maximum mission time, are deemed to be late.

the red vertical line). Furthermore, task 3 and 6 becomes unassigned for certain time periods. By this, the effects of different UAVs working on different data, some outdated and some not, can clearly be seen: while executing its local control loop, each UAV computes an optimal solution to the OTASF-MCUS instance it holds it in its memory. However, the different local versions of the the status of the world result in a disagreement regarding the team composition that should perform those tasks, leaving some tasks uncovered. Fortunately, Theorem 1 rescues us, guaranteeing that those tasks will be properly allocated and the mission can terminate, if the UAVs are allowed to communicate often enough.

Figure 6 shows how the number of finished tasks evolved over time (blue line), as well as the expected number of tasks that are not yet finished and will only be finished late, after the maximal mission time $\hat{u}$. One can clearly see the effect of different UAVs working on different sets of data for the time period mentioned above. In particular, one notes that the red line is not constant at the beginning of the overall mission, indicating how knowledge spreads among the swarm.

Finally, Figure 7 can provide a complete overview of the mission under consideration by reporting the evolution of each UAV's path. Starting from the hangar (corresponding to the location of task 0), each UAV assigns itself to a task and commences to fly toward its location (dashed lines). We indicate the time when it arrives to the task's location with an empty dot, while the moment in which it starts working on the task with the designated team is indicated by full dots and continuous lines. In

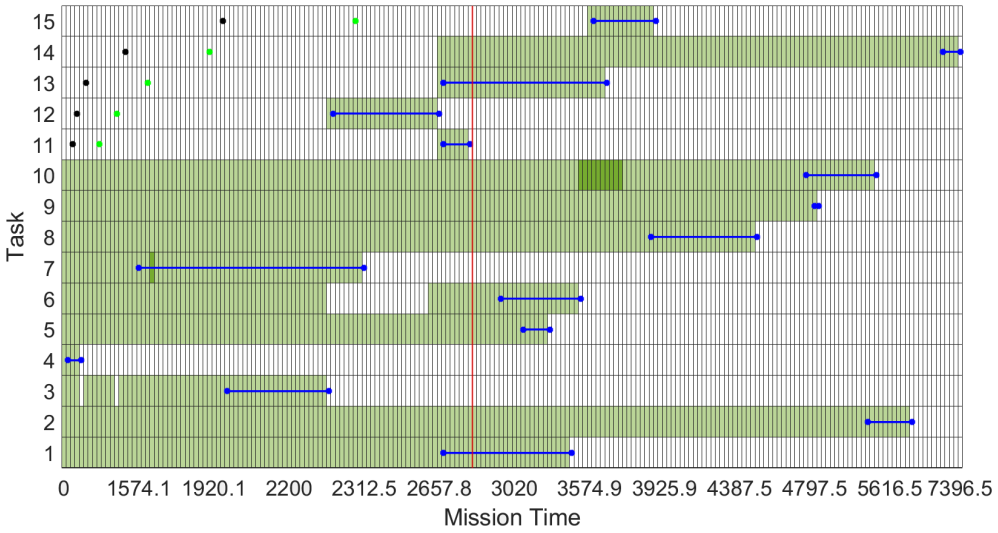

Fig. 5 Dynamic Case I representation. Along the Mission Time (x-axis), for each task (reported on the y-axis) the light green colour indicates that a whole team is agreeing to work on it at some point of the mission, while a darker green is used when there is more than a team that decides to work on the task and white cells stand for times where no team agrees on executing the task. Blue dots and lines depict the actual time in which the corresponding task is executed, with a vertical red line expressing the maximum mission time. Tasks completed afterwards, on red line's right, are deemed to be late. Black and green dots at tasks $11 - 15$ indicate when the corresponding task is revealed to the designated UAV (black) as well as when the latter sends the first communication about them out to the swarm (green).

this way it is possible to look at the state of each UAV over time and their "believes" on what is the next task they should be working on. One can see that the arrival of new tasks during the mission causes changes of plans of individual UAVs as well as moments of idling, where UAVs wait for team members that will never arrive. Plan adjustments happen throughout all the mission, with UAV 2 for example assigning itself to task 12 even though it has already been completed by other UAVs.

6.4 Dynamic Case II

Analogously to Dynamic Case I, this case considers a scenario where more than one UAV in the swarm is made aware of the presence of new tasks. In particular, UAVs 3, 5, 9 and 10 are made aware of new tasks and can then communicate the new information to their neighbours. As before, the overall mission time necessary for the completion of all tasks is ca. 20% higher than for the static case, even though the mission terminates slightly earlier than in the previous dynamic situation. Figure 8 shows that for this mission one can observe an earlier and more pronounced disagreement phenomenon, reflected in times where some tasks are not allocated to any team (the white spaces between green ones). Tasks 11, 12 and 15 are now completed early on, with task 11 performed immediately after its existence has become known to a member of the swarm. Indeed UAV 5, 9 and 10 are promptly made aware of the task

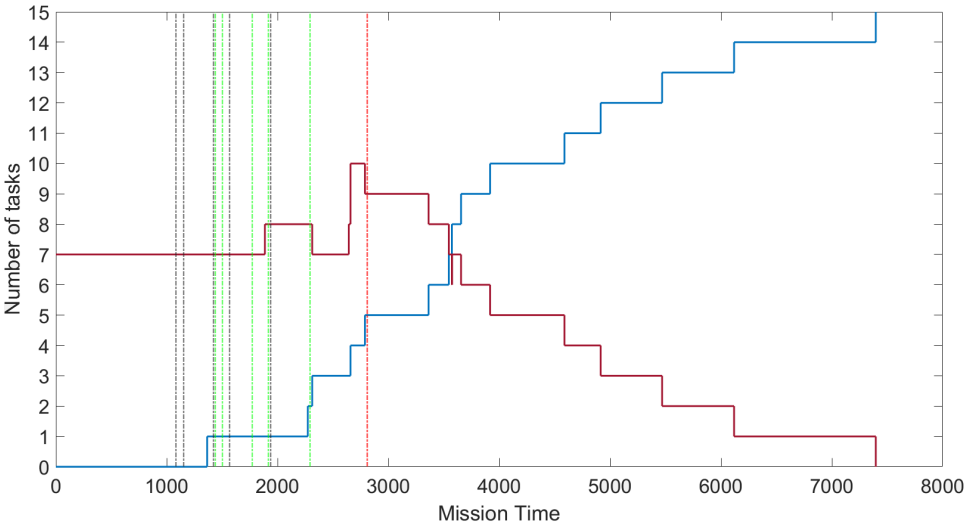

Fig. 6 Expected number of late tasks in Dynamic Case I. Along the Mission Time (x-axis), the number of current task that the swarm (averaged among all UAVs) deems to be late is depicted in burgundy, while a blue line indicates the number of completed tasks up to that point, with tasks completed after the vertical red dashed line, expressing the maximum mission time, are deemed to be late. Vertical dashed lines indicate when a new task is revealed (black) as well as when the first communication about it is sent out to the swarm (green).

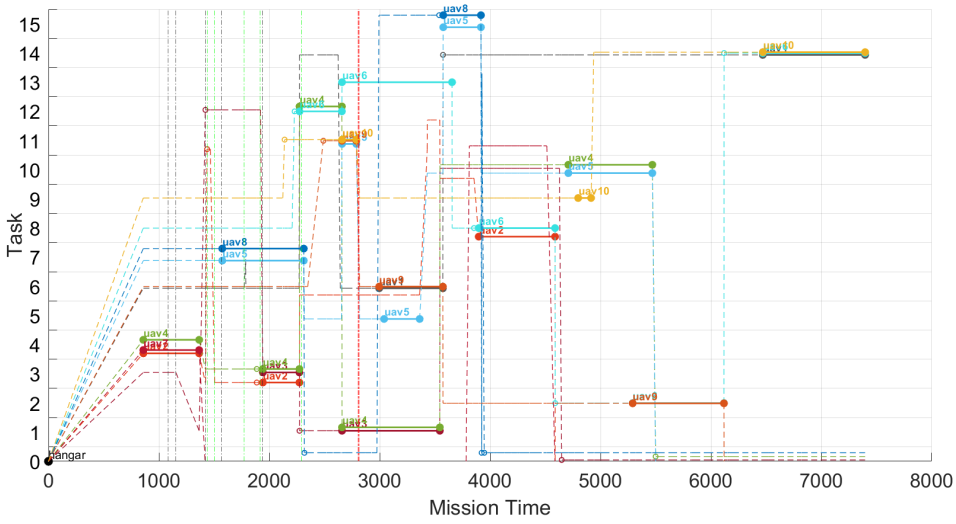

Fig. 7 Mission scheduling for Dynamic Case I. Each UAV path is followed starting from the hangar (located at task 0), during its flight (dashed lines) and working period (continuous lines). Empty dots represent the time when the corresponding UAV arrives to the task's location, while filled dots depict the moment in which it starts or finishes working on the assigned task.

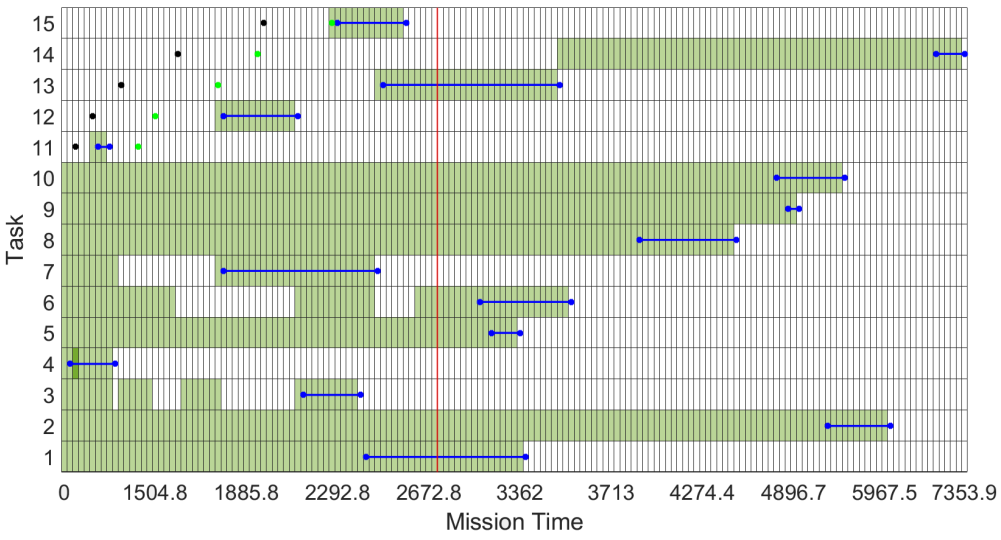

Fig. 8 Dynamic Case II representation. Along the Mission Time (x-axis), for each task (reported on the y-axis) the light green colour indicates that a whole team is agreeing to work on it at some point of the mission, while a darker green is used when there is more than a team that decides to work on the task and white cells stand for times where no team agrees on executing the task. Blue dots and lines depict the actual time in which the corresponding task is executed, with a vertical red line expressing the maximum mission time. Tasks completed afterwards, on red line's right, are deemed to be late. Black and green dots at tasks $11-15$ indicate when the corresponding task is revealed to the designated UAVs (black) as well as when they send the first communication about the task out to the swarm (green).

and are then assigned to it. As before, Theorem 1 comes into play guaranteeing that the remaining tasks will be properly allocated and the mission can terminate, if the UAVs are allowed to communicate often enough. In addition, early disagreement is also reflected in Figure 9 where one can observe a more dynamic behaviour of the number of tasks the swarm deems to deliver late.

Similarly to case I, Figure 10 provides a complete overview of the mission under consideration by reporting the evolution of each UAV's path. One can again observe that the arrival of new tasks during the mission causes changes of plans of individual UAVs as well as moments of idling, where UAVs wait for team members that will never arrive. Differently from case I though, due to a faster information spreading, plan adjustments happen mostly close to the arrival of a new task, while in the second part of the mission the situation becomes much less dynamic.

6.5 Computation Time

We have seen that in both static and dynamic case we are able to find a solution for the task assignment problem with the above designed distributed optimisation framework allowing us to analyse the impact of real-world and real-size situations.

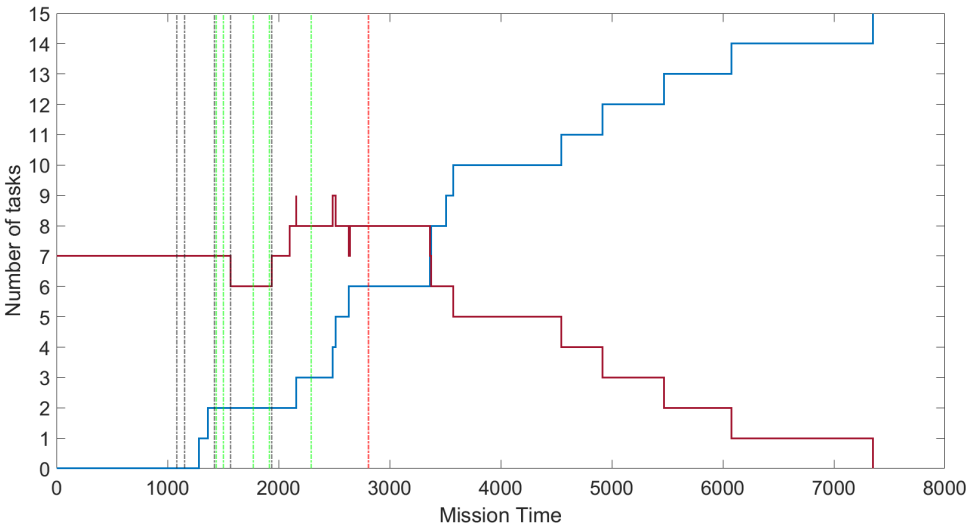

Fig. 9 Expected number of late tasks in Dynamic Case II. Along the Mission Time (x-axis), the number of current task that the swarm (averaged among all UAVs) deems to be late is depicted in burgundy, while a blue line indicates the number of completed tasks up to that point, with tasks completed after the vertical red dashed line, expressing the maximum mission time, are deemed to be late. Vertical dashed lines indicate when a new task is revealed (black) as well as when the first communication about it is sent out to the swarm (green).

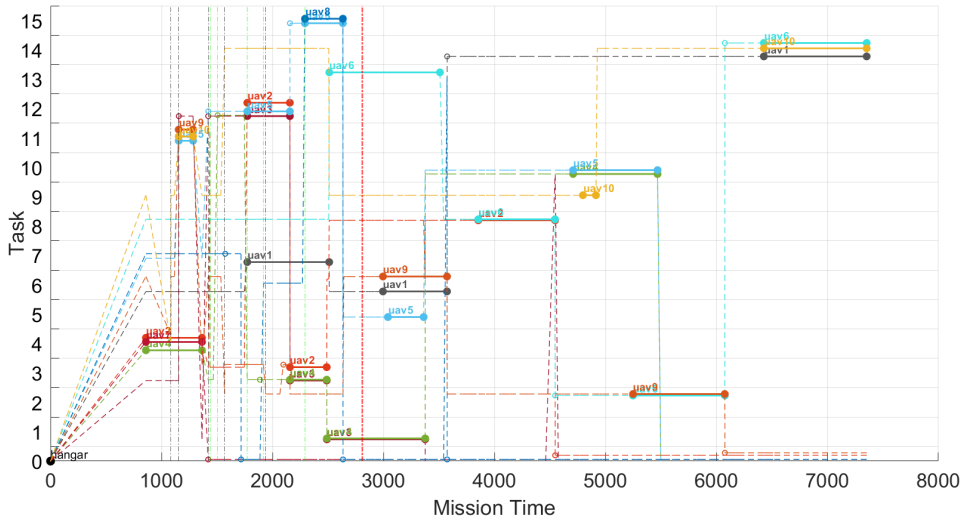

Fig. 10 Mission scheduling for Dynamic Case II. Each UAV path is followed starting from the hangar (located at task 0), during its flight (dashed lines) and working period (continuous lines). Empty dots represent the time when the corresponding UAV arrives to the task's location, while filled dots depict the moment in which it starts or finishes working on the assigned task.

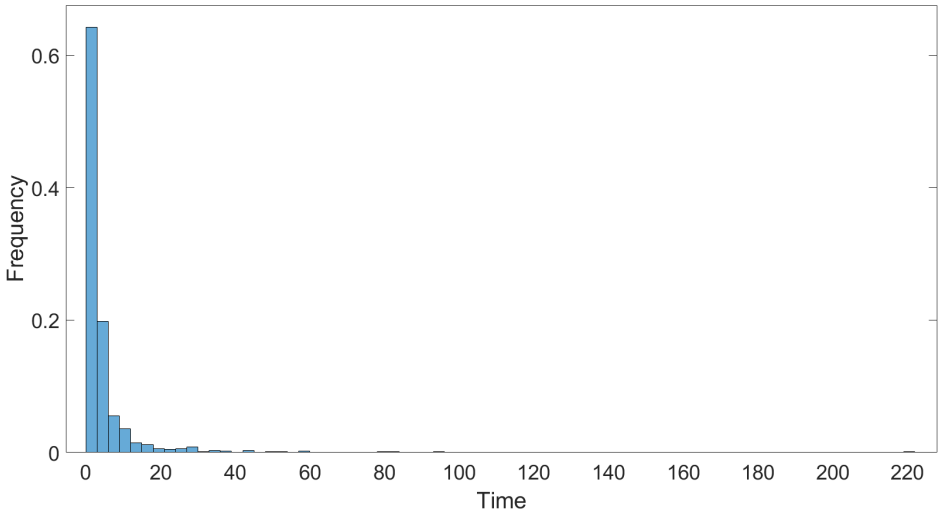

Fig. 11 Histograms of computational times (in seconds) needed for solving instances of the optimisation problem OTASF-MCUS with 10 UAVs, 15 tasks as well as randomised task durations d_j and start times l_j. The other parameters are described in the Section 6.

Nonetheless, the OTASF-MCUS model is of sufficient complexity to make computation times a potentially relevant subject, at least for mid- to large-sized problems. In other words, we consider in this subsection how long it take for a UAV to update its own decisions depending on the complexity of the problem in terms of number of UAVs, and consequently teams, number of tasks as well as mission time limit.

In what follows, we only provide computational times for the actual solution process of the given mixed-integer problem instances, i.e. the time it takes for CPLEX to generate a solution. Our numerical experiments indicate that the additional overhead for executing relevant Matlab scripts bears no significance on the overall results.

6.5.1 Empirical Distribution of Computational Times

We consider again the problem setup given above for the OTASF-MCUS model. With this setting, we run 1000 simulations, in each of which we randomise both the task durations d_j as well as all task start times ℓ_j. More precisely, we assume that $d_j \sim \mathcal{U}[0, 1200]$ and that $\ell_j \sim \mathcal{U}[0, 2400]$, $\forall j \in \{1, \ldots, m\}$. The corresponding empirical distribution of computational times is depicted in Figure 11. More than 60% of all solution times are below 3s. However, a small number of difficult instances have computation times in the range of 40 – 220 seconds, leading to an average computational time of 4.64s. This effect is even more evident if we compare it with the median computational time, equal to 2.12s. If we compute the average solution time conditional upon it being in the 80th percentile, we arrive at 2.02s. In contrast, the average computational time conditional upon it not being in the 80th percentile of the overall distribution is of ca. 15.13s, a massive difference.

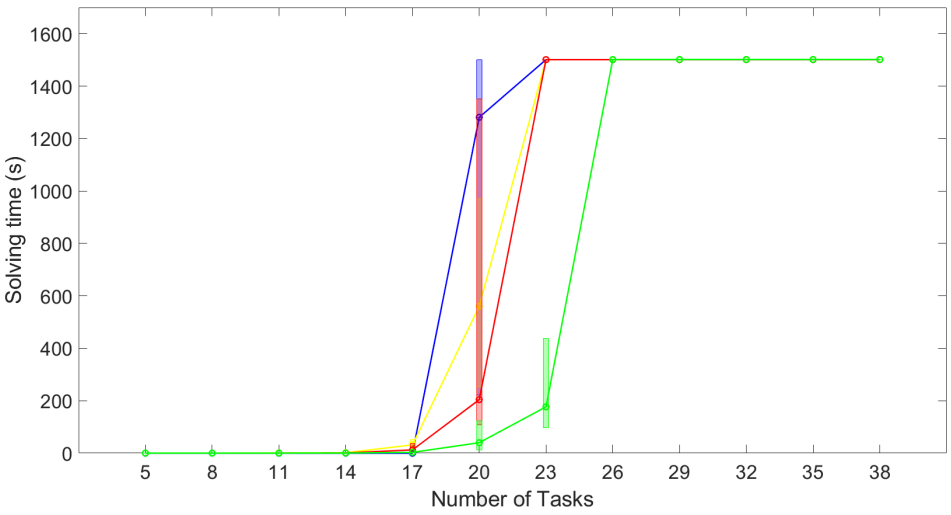

Fig. 12 Sensitivity analysis of computation times with respect to the number of tasks. Fixing a mission time limit, for different number of available resources (blue - 8 UAVs, yellow - 15 UAVs, red - 22 UAVs and green - 29 UAVs), percentiles of solving times distributions (y-axes) are depicted for each number of tasks considered (x-axis). Mission instances are randomly created.

6.5.2 Computation Times for Varying Numbers of UAV and Tasks

In this section, we consider different instances of the OTASF-MCUS problem to see how computational times vary with respect to changes in the number of tasks and UAVs used. In particular, we consider instances where

- Number of UAVs varies from 8 to 29
- Number of Tasks goes from 5 to 38

while the mission time limit $\hat{u}$ is set as before at 2160s. Then, for each combination of no. of UAV and no. of tasks, teams of UAVs are randomly determined, with the preference of building teams constituted by 2 or 3 UAVs. All the other problem parameters (i.e. UAV capacities L_i, task start times ℓ_j, travel distances $f_{j,k}$, etc) are also randomly selected. Subsequently, for each setting, 30 problem simulations and corresponding solutions are computed, using randomly generated task duration times $d_j \sim \mathcal{U}[0, 1200]$. Results are reported in Figure 12, where bar limits in the graph correspond to the 25-th and 75-th percentile of the computing times distribution, while medians are represented by dots. A cap of 1500s (25min) has been set to the computational times. As it can be seen, computation times increase strongly when the number of tasks reaches a critical threshold,

6.5.3 Computation Times for Various Mission Time Limits

On the other hand, it is possible to examine the effects of a different mission time limit and, given the last observation of the previous section, we fix the no. of tasks to 23, while considering instances where

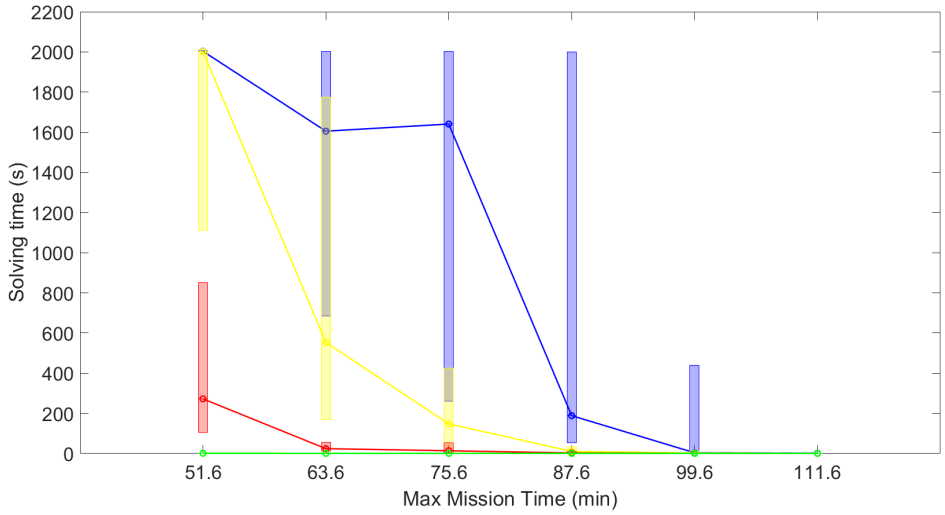

Fig. 13 Sensitivity analysis of computation times with respect to the mission time constraints. Fixing the number of tasks to 23, for different number of available resources (blue - 8 UAVs, yellow - 15 UAVs, red - 22 UAVs and green - 29 UAVs), percentiles of solving times distributions (y-axes) are depicted for each mission time limits (x-axis). Mission instances are randomly created.

- Number of UAVs varies again from 8 to 29
- Mission time limit increases from ca. 50min to 110min

As before, for all combinations of the two elements above, we generate randomly all parameters and 30 problem simulations are computed, along with the corresponding solutions. Allowing a larger limit of 2000s, computational times are collected and the results are reported in Figure 13. It is possible to observe that relaxing the mission time limit has beneficial effects on easing the problem, with a change of roughly 12min being sufficient to significantly reduce the difficulty of the problem. As a final consideration, it is possible to notice that computational times can exhibit high variability, also depending on the duration time d_j necessary for task's completion.

7 Conclusions

This paper provides a way to investigate into task assignment problems as they are faced by a cooperative fleet of UAVs. We have provided a distributed computational framework where each UAV can make its own decisions to reach swarm goals within uncertain circumstances and quite general conditions as well as many real-world aspects of the problem. Numerical experiments show that by leveraging on state-of-the-art optimisation algorithms and solvers, we can exploit the structure of the underlying mathematical optimisation model and solve problems of realistic size in seconds. In our case study it can be seen that it appears to be detrimental when even a small number of UAVs work on outdated data, thereby revealing the importance of communication.

Nonetheless, several interesting model extensions fell outside the scope of this study and any investigation of them has not been pursued yet. We would like to highlight what we believe are the most relevant avenues for further research. First, it is possible to consider *different objectives* for the mathematical model, whose changes might significantly affect the computation times needed to produce optimal solutions. In addition, computational times encountered appear to make it feasible to consider interactive approaches to solving real-world task assignment problems, and explore the (implicit) preferences of any operator. As a second area of investigation is the consideration of *hard prioritisation of tasks*, where the mathematical optimisation model is extended to consider the case enforcing certain tasks to be finished before their assigned due time. Finally, a *dynamic communication structure* can be explored, where the communication network structure might change with time and according to UAVs positions.

Statements and Declarations

- **Funding**: The authors did not receive support for the submitted work;

- **Competing Interests**: The authors have no relevant financial or non-financial interests to disclose;

- **Ethics approval**: The authors do not acknowledge any ethical conflict;

- **Consent to participate/ publish**: All authors agreed on the above manuscript and on the publication of the work;

- **Availability of data and materials**: All data used are made available along the manuscript;

- **Code availability**: The code will be made available on request;

- **Authors' contributions**: All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Luigi Bobbio and Joerg Fliege. The first draft of the manuscript was written by Luigi Bobbio, Joerg Fliege and Antonio Martinez-Sykora and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

References

Department of Defense, UAV Roadmap 2002-2027: Department of Defense, UAV Roadmap 2002-2027

Gerkey, B.P., Mataric, M.J.: A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research* **23**, 939–954

(2004)

Nunes, E., Manner, M., Mitiche, H., Gini, M.: A taxonomy for task allocation problems with temporal and ordering constraints. *Robotics and Autonomous Systems*, 55–70 (2017)

Mataric, M.J.: Designing and understanding adaptive group behaviors. *Adaptive Behavior* **4** (1995)

Reif, J., Wang, H.: Social potential fields: A distributed behavioral control for autonomous robots. *Robotics and Autonomous Systems* **27**, 171–194 (1999)

Forsmo, E.J., Grøtli, E.I., Fossen, T.I., Johansen, T.A.: Optimal search mission with unmanned aerial vehicles using mixed integer linear programming. *International Conference on Unmanned Aircraft Systems*, 253–259 (2013)

Kendall, E.N., Phillip, R.C., Meir, P.: Dynamic network flow optimization model for air vehicle resource allocation. *Proceedings of the American Control Conference*, 1853–1858 (2001)

Peng, J., Wen, M.F., Xie, G.Q., Zhang, X.Y., Lin, K.C.: Coordinated dynamic mission planning scheme for intelligent multi-agent systems. *Journal of Central South University* **19**, 3170–3179 (2012)

Alighanbari, M., How, J.P.: Cooperative task assignment of unmanned aerial vehicles in adversarial environments. *American Control Conference*, 4661–4666 (2005)

Passino, K., Polycarpou, M., Jacques, D., Pachter, M., Liu, Y., Yang, Y., Flint, M., Baum, M.: Cooperative control for autonomous air vehicles. *Cooperative Control and Optimization*, 233–277 (2000)

Parunak, H.V.D., Purcell, M., O'Connell, R.: Digital pheromones for autonomous coordination of swarming uavs. *Tech. Conference Workshop Unmanned Aerospace Vehicles* **3446**, 1548–1579 (2002)

Darrah, M., Niland, W., Stolarik, B.: Multiple uav dynamic task allocation using mixed integer linear programming in a sead mission. *Proceedings of the Infotech at Aerospace* (2005)

Secrest, B.R.: Traveling salesman problem for surveillance mission using particle swarm optimization. *School of Engineering and Management, Air Force Institute of Technology Wright-Patterson* (2001)

Balas, E., Padberg, M.W.: Set partitioning: A survey. *SIAM Review* **18**, 710–760 (1976)

Service, T.C., Adams, J.A.: Coalition formation for task allocation: Theory and algorithms. *Autonomous Agents and Multi-Agent Systems* **22**, 225–248 (2011)

Shima, T., Rasmussen, S.J., Sparks, A.G., Passino, K.M.: Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms. *Computers & Operations Research* **33**, 3252–3269 (2006)

O'Rourke, K.P., Carlton, W.B., Bailey, T.G., Hill, R.R.: Dynamic routing of unmanned aerial vehicles using reactive tabu search. *Military Operations Research* **6**, 5–30 (2001)

Chen, Y., Yang, D., Yu, J.: Multi-uav task assignment with parameter and time-sensitive uncertainties using modified two-part wolf pack search algorithm. *IEEE Transactions on Aerospace and Electronic Systems* **54**, 2853–2872 (2018)

Blum, C.: Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews* **2**, 353–373 (2005)

Wang, D., Tan, D., Liu, L.: Particle swarm optimization algorithm: An overview. *Soft Computing* **22**, 387–408 (2018)

Zhang, K., Collins, E.G., Shi, D.: Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction. *ACM Transactions on Autonomous and Adaptive Systems* **7**, 1–22 (2018)

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

Oh, G., Kim, Y., Ahn, J., Choi, H.-L.: Market-based distributed task assignment of multiple unmanned aerial vehicles for cooperative timing mission. *AIAA - Journal of Aircraft* **54**, 2298–2310 (2017)

Qin, B., Zhang, D., Tang, S., Wang, M.: Distributed grouping cooperative dynamic task assignment method of uav swarm. *MDPI Applied Sciences* **12** (2022)

Gudmundsson, J., Hougaard, J.L., Platz, T.T.: Decentralized task coordination. *European Journal of Operational Research* **304**, 851–864 (2023)

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

Ponda, S., Redding, J., Choi, H.-L., How, J.P., Vavrina, M., Vian, J.: Decentralized planning for complex missions with dynamic communication constraints. *In Proceedings of the 2010 American Control Conference*, 3998–4003 (2010)

Whitten, A.K., Choi, H.-L., Johnson, L.B., How, J.P.: Decentralized task allocation with coupled constraints in complex missions. *In Proceedings of the 2011 American Control Conference* **22**, 1642–1649 (2011)

Whitbrook, A., Meng, Q., Chung, P.W.H.: A novel distributed scheduling algorithm for time-critical multi-agent systems. *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 6451–6458 (2015)

Zhao, W., Meng, Q., Chung, P.W.H.: A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario. *IEEE TRANSACTIONS ON CYBERNETICS* **46**, 902–915 (2018)

Cui, W., Ruilin, L., Yanxiang, F., Yikang, Y.: Distributed task allocation for a multi-uav system with time window constraints. *MDPI drones* **6**, 226 (2022)

Geng, L., Zhang, Y.F., Wang, J., Fuh, J.Y., Teo, S.H.: Cooperative mission planning with multiple uavs in realistic environments. *Unmanned Systems* **2**, 73–86 (2014)

Coffman, E.G., Elphick, M., Shoshani, A.: System deadlocks. *ACM Computing Surveys* **3**, 67–78 (1971)

Lemaire, T., Alami, R., Lacroix, S.: A distributed tasks allocation scheme in multi-uav context. *Proceedings of the International Conference on Robotics and Automation*, 3622–3627 (2004)

Masticola, S.P.: Static detection of deadlocks in polynomial time. PhD thesis, Graduate School—New Brunswick Rutgers, The State University of New Jersey (1993)

Mazdin, P., Rinner, B.: Distributed and communication-aware coalition formation and task assignment in multi-robot systems. *IEEE Access* **9**, 35088–35100 (2021)

Lu, Z.: Task assignment under uncertainty: stochastic programming and robust optimization approaches. *International Journal of Production Research* **53**, 1487–1502 (2015)

Evers, L., Barros, A.I., Monsuur, H., Wagelmans, A.: Online stochastic uav mission planning with time windows and time-sensitive targets. *European Journal of Operational Research* **238**, 348–362 (2014)

Tang, Y., Xing, X., Karimi, H.R., Kocarev, L., Kurths., J.: Tracking control of networked multi-agent systems under new characterizations of impulses and its applications in robotic systems. *IEEE Transactions on Industrial Electronics* **63**, 1299–1307 (1994)

Shyalika, C., Silva, T., Karunananda, A.: Reinforcement learning in dynamic task scheduling: A review. *Springer Nature Computer Science* **1** (2020)

Seenu, N., Kuppan Chetty, R.M., Ramya, M.M., Mukund Nilakantan, J.: Review on state-of-the-art dynamic task allocation strategies for multiple-robot systems. *Industrial Robot* (2020)

Poudel, S., Moh, S.: Task assignment algorithms for unmanned aerial vehicle networks: A comprehensive survey. *Vehicular Communications* **35** (2022)

Younas, I., Kamrani, F., Bashir, M., Schubert, J.: Efficient genetic algorithms for optimal assignment of tasks to teams of agents. *Neurocomputing* **314**, 409–428 (2018)

Di Martinelly, C., Meskens, N.: A bi-objective integrated approach to building surgical teams and nurse schedule rosters to maximise surgical team affinities and minimise nurses' idle time. *International Journal of Production Economics* **191**, 323–334 (2017)

Van Den Eeckhout, M., Maenhout, B., Vanhoucke, M.: A heuristic procedure to solve the project staffing problem with discrete time/resource trade-offs and personnel scheduling constraints. *Computers & Operations Research* **101**, 144–161 (2019)

Saber, R.G., Ranjbar, M.: Minimizing the total tardiness and the total carbon emissions in the permutation flow shop scheduling problem. *Computers & Operations Research* **138** (2022)

Niknafs, A., Denzinger, J., Günther, R.: A systematic literature review of the personnel assignment problem. *Proceedings of the International MultiConference of Engineers and Computer Scientists* **II** (2013)

Fikar, C., Hirsch, P.: Home health care routing and scheduling: A review. *Computers & Operations Research* **77**, 86–95 (2017)

Bertsekas, D.P., Tsitsiklis, J.N.: Some aspects of parallel and distributed iterative algorithms—a survey. *Automatica* **27**, 3–21 (1991)

Bertsekas, D.P.: Distributed asynchronous computation of fixed points. *Mathematical Programming* **27**, 107–120 (1983)

Fourer, R., Gay, D.M., Kernighan, B.W.: AMPL. A modeling language for mathematical programming (2003)

IBM CPLEX: IBM CPLEX Optimizer, `https://www.ibm.com/uk-en/analytics/cplex-optimizer`

Ajtai, M., Komlós, J., Szemerédi, E.: The longest path in a random graph. *Combinatorica* **1**(1), 1–12 (1981)

Bertsekas, D.P., Tsitsiklis, J.N.: *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, Nashua (2015)

Israel, R. personal communication, University of British Columbia, University of British Columbia

Knowledge Hub 'Maths for CEME' online portal, `https://khub.net/group/maths-of-ceme`

Mataric, M.J.: Behavior based control: Examples from navigation, learning, and group behavior. *Journal of Experimental & Theoretical Artificial Intelligence* **9**, 323–336 (1997)

Steels, L.: The artificial life roots of artificial intelligence. *Artificial Life* **1**, 75–110 (1994)