Proceedings of the Thirty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2025)

A Flow Based Planning Method for MultiAgent Progression with Deployable Agents and Communication Constraints

Emile Siboulet[1 2], Roland Godet[1 3], Arthur Bit-Monnot[1], Marc-Emmanuel Coupvent Des Graviers[2], Christophe Guettier[2], Simon Lacroix[1]

[1]LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France
[2]Safran Electronics & Defense, Massy, France
[3]ONERA/DTIS, Université de Toulouse, France

Abstract

This paper deals with the problem of planning multiple agent movements through a mission area modeled as a graph. The agents undergo classic communication and temporal constraints, and the quantitative objective is the minimization of the team's traversal makespan. Additional specificities make the problem a particularly complex routing one: on some nodes are associated durative and coordinated actions to perform, which can involve either the co-presence of several agents or time dependencies. Also, some agents are deployable and able to move on denser graphs: namely, aerial robots can take off and land on a ground vehicle at any planned position, and can fly above ground obstacles. We model the problem as a Constraint Satisfaction Problem (CSP) and solve it with a network flow model. Results show the efficiency of the model and resolution scheme, which provides solutions one to two orders of magnitude faster than a numeric-temporal hierarchical planning model, with only a few percent losses of optimality.

Introduction

This paper considers the planning of a *multiagent progression mission*. A progression mission consists in traversing an area modeled by a traversability graph, performing tasks on some vertices with temporal constraints, such as synchronicity or sequences. Such missions pertain to various applications contexts, such as firefighting (Shahidi, Ramezanian, and Shahparvari 2022), hostile zone exploration (Guettier 2007; Kenzin, Bychkov, and Maksimkin 2019), or rescue scenarios (Bevacqua et al. 2015). The scenario we based our study is depicted in (Godet, Lesire, and Bit-Monnot 2023; Siboulet et al. 2024; CoH 2023b,a)

According to the taxonomy proposed in (Korsah, Stentz, and Dias 2013), a progression mission is a multiagent multitask problem with temporal assignments and compound tasks. Planning for such missions is particularly complex to solve and is generally tackled by Constraint Satisfaction Problem (CSP) dedicated solvers (Shiroma and Campos 2009), or by domain-specific algorithms (Zlot 2006).

Our problem share with the Vehicle Routing (VRP) and Team Orienteering (TO) problems depicted in (Gunawan, Lau, and Vansteenwegen 2016; Vansteenwegen, Souffriau, and Van Oudheusden 2011; Guastalla, Aringhieri, and Hosteins 2024), the requirements to solve allocation, scheduling and routing subproblems. Of particular interest is the approach of leveraging a CSP with flow constraints which allows efficient problem-solving (Bredström and Rönnqvist 2008), while having sufficient expressiveness to plan tasks and trips (Ferreira et al. 2024). We adopt this modeling approach, and consider two additional aspects that correspond to realistic missions, but also add significant complexity:

- Agents evolve under communication constraints. In particular, one of them must be permanently connected to all others: this instantiates the presence of a mobile control-command station that is involved in the progression;
- Some agents associated to *carrier agents* can be deployed, but must be recovered before a maximum duration. Namely, such agents are flying vehicles that can in particular observe parts of the environment prior to traversal by ground agents (De Petris et al. 2022).

The following section presents the formalization of the considered progression problem. The next one depicts in detail the used model to efficiently solve the problem. Results are then provided, and in particular compared with a classical planning model approach. A discussion concludes the paper.

Problem Formalization

The problem consists of moving n^a agents from entry positions Et_a to exit positions Ex_a on a graph in which edges represent motions that depend on each agent's traverse capacity or speed: $G_a = (V, E_a)$. The traversal time of an edge (v, v') by an agent a is $Tt_{a(v,v')}$.

Tasks and associated constraints A task p is associated to a vertex (or position) $Pt_p \in V$, has a duration Dt_p and must be achieved within a time window Wt_p. Tasks are typed, and can only be achieved by an agent that is not in the subset Rt_p. Dependencies between tasks are introduced: pairs of synchronized tasks (p, p'), not necessarily associated with the same vertex, must start at the same time (pairs are stored in S); and precedence conditions between two tasks p and p' impose that task p must finish before p'

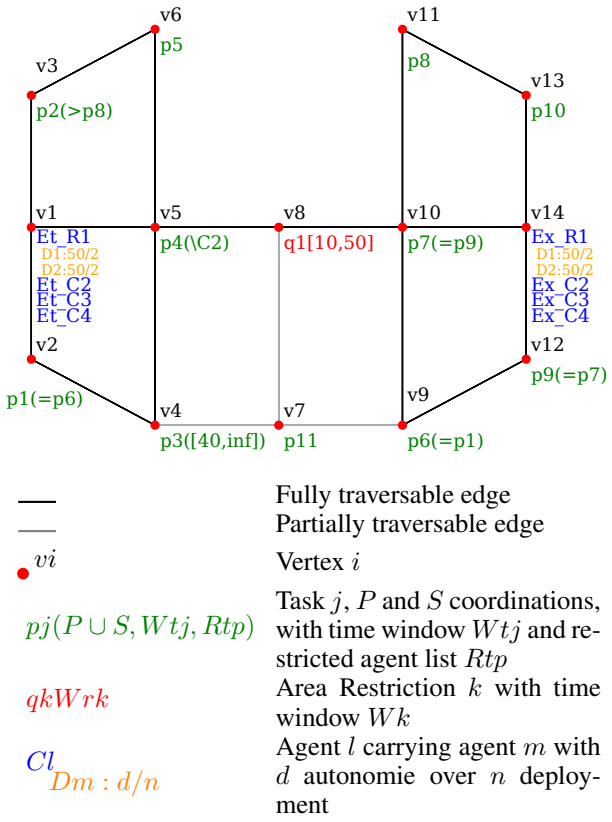


Figure 1: Illustration of a progression planning problem for a fleet of heterogeneous agents on a graph with tasks and temporal constraints

starts (pairs are stored in P). In Figure 1, tasks and constraints are depicted in green, $task(constraints)$, for instances $p4(\backslash C2)$ means that $p4$ cannot be performed by $C2$. $p1(= p6)$ means that $p1$ is to be done synchronously to $p6$ These temporal constraints can be defined over more than two tasks. In particular, there is a list F of agents that can only reach a vertex after another agent has reached it (this instantiates the notion of scouting).

Temporal exclusions A series of temporal exclusions q are defined: an agent $a \notin Ar_q$ (the set of agents exempted from the temporal exclusion) must not lie on the position (vertex) Pr_q during a given time window Wr_q (this instantiates temporary dangerous areas). In Figure 1, temporal exclusion is in red $q1[10, 50]$ means that no agent should be on $v8$ from 10 to 50.

Deployable agents A set D of deployable agents is defined. Each deployable agent $a \in D$ is associated to a carrier agent C_a. A deployable agent $a \in D$ can be deployed at most Nd_a times, each time for a maximum duration Fd_a, this model energy capacity of flying agents. The deployment and recovery of deployable agents impose that the associated carrier agent and the deployable agent perform a synchronous durative action at the same vertex that lasts Dd_a. This deployment and recovery actions are exclusive of the

Symbol	Meaning
V	Vertices of the navigation graph
G_a	Navigation graph of agent a
E_a	Edges of agent a navigation graph
$Tt_{a(v,v')}$	Travel time of agent a for the edge (v, v')
Et_a	Entry position of agent a
Ex_a	Exit position of agent a
Pt_p	Task p position
Dt_p	Task p duration
Wt_p	Task p time window
Rt_p	Task p set of non authorized agents
S	Set of tasks pairs that have to be synchronized
P	Set of tasks pairs that have to be sequential
F	set of agents that cannot be the first to reach vertices
Pr_q	Temporal exclusion q position
Wr_q	Temporal exclusion q time window
Ar_q	Set of agents exempted from temporal exclusion q
D	Set of deployable agents
C	Set of carrier agents
Nd_a	Maximum number of agent a deployments
Fd_a	Maximum duration of a deployment of an agent a
Dd_a	Maximum duration of the starting or ending phases of the agent a deployment
C_a	Carrier associated to a deployable agent $a \in C$
Gr	Communication graph
Er	Bidirectional edge of the communication graph
a^r	Central agent

Table 1: Problem constant symbols

completion of any other task for both agents. In Figure 1, carrier agents are in blue and deployable agent in orange, with the deployable agent under its carrier agent . In our figure, $R1$ is carrying $D1$ and $D2$ that can be deployed twice for 50 units of time.

Communications A bidirectional communication graph $Gr = (V, Er)$ is defined. A specific agent a^r must be directly connected through this graph to *all* the other agents *at all times* (this instantiates a central monitoring agent). This monitoring agent cannot change during the mission. Communication connectivity on edges is ensured by the connectivity of the two associated vertices. Communications are not depicted on Figure 1.

Solution definition $\mathcal{T}_{av}$ denotes the arrival time of an agent a on vertex v and $\mathcal{D}_{av}$ the duration the agent a stays on v. Both should be 0 if the agent a does not pass through the vertex. The completion of a task p by an agent a is stored in the variable $\mathcal{R}t_{ap}$, and is executed at time $\mathcal{T}t_p$. The satisfaction of a temporal exclusion q is stored in the variable $\mathcal{R}r_{aq}$.

A solution plan is the complete assignment of all the

Symbol	Meaning
$\mathcal{T}_{av}$	Arrival time of agent a at vertex v
$\mathcal{D}_{av}$	Duration of agent a at position v
$\mathcal{T}t_p$	Task p execution time
$\mathcal{R}t_{ap}$	Realization of task p by agent a
$\mathcal{R}r_{aq}$	Realization of the temporal exclusion q by the agent a

Table 2: Problem variables

variables $\mathcal{T}_{av}, \mathcal{D}_{av}, \mathcal{T}t_p, \mathcal{R}t_{ap}, \mathcal{R}r_{aq}$. The optimal plan is the one that satisfies all the constraints and minimizes the makespan $\mathcal{M}$ for the carrier agents (note that all the deployable agents must have been retrieved by their associated carriers by this date).

$$\mathcal{M} = \max_{a \in C}(\mathcal{T}_{aEx_a} + \mathcal{D}_{aEx_a}) \quad (1)$$

Our domain does not allow agents to return to a previously visited position. This method of resolution is conditioned by the scenario, which recommends that each agent should only pass over each position at most once. Should we wish to relax this constraint, we can duplicate the navigation graph n times to enable n position-based replay.

An instance of the considered problem is shown in Figure 1. The navigation graph comprises 14 vertices $v^1, ..., v^{14}$. The mission is performed by 4 agents: $R1, C2, C3, C4$ starting at v^1 and ending it at v^{14}. They are assisted by two agents $D1$ and $D2$, associated to the carrier agent $R1$, that can each be deployed twice for a maximum duration of 50 units. Various durative tasks $p^1, ..., p^{11}$ are to be performed, sometimes with inter-task of absolute time constraints. The presence of any agent on v^8 between time instants 10 and 50 is precluded. The v^7 vertex is only accessible to the $D1$ or $D2$ agents, so solving p^{11} requires the deployment and recovery of a deployable agent. Agent $R1$ is the team's central agent and must be in communication with all other agents at any time.

Constraint Programming Model

Our approach to solve the mission planning problem is inspired by Guettier (2007); Shiroma and Campos (2009) on similar progression problems. It consists in encoding the problem into a CSP formalism with finite-domain integer variables, and in deducing the actions to be performed by reading the variable values satisfying the CSP. Our approach implies additional assumptions:

- A task to be performed at a position is carried out at the time the agent arrives at the position.
- The actions of deploying and recovering deployable agents are performed just before leaving a position.

These assumptions do not impede the correctness of the found solutions, but might result a slight increase of mission makespan.

Symbol	Meaning
Constants	
G'	CSP graph
V'	CSP vertex
Vd	Ordered set of agent storage positions
E'_a	CSP Edge of agent a
$Tt'_{a(v,v')}$	CSP Travel time of agent a on edge (v, v')
B_{av}	Agent a balance at position v
v^s	Starting position in the graph
v^e	Ending position in the graph
Variables	
$\mathcal{F}_{ak}$	Agent a flow on the k^{th} edge
$\mathcal{D}p_{av}$	Deployment of the agent a at position v

Table 3: CSP introduced symbols

Problem CSP Encoding

Boolean variables are represented as binary integer variables where 1 encodes *true* and 0 encodes *false*. To lighten the notation, we do not systematically recall the definition range of all variables. If not specified, the variable takes all values over which it can be defined. The disjunctions (resp. conjunctions) are denoted by a vertical line (resp. a brace) on the left of formulas. Table 3 lists the symbols associated with the CSP encoding.

Constraint-based modeling of flows on graphs for mission planning implies path continuity for all agents. However, deployable agents can be deployed and recovered at any vertex on the problem graph. We therefore introduce additional vertices that enable the solver to find continuous paths for all agents. To produce the solution plan, these vertices are just ignored.

The complete graph given to the constraint solver is $G'_a = (V', E'_a)$, which includes the graph G:

$$V' = V \cup \{v^s, v^e\} \cup Vd \quad (2)$$

Where v^s and v^e are the starting and ending position common to all agents, $Vd = \{v_d^1, ..., v_d^{n^d}\}$, with $n^d = \max_{a \in D}(Nd_a) + 1$, are artificial vertices, called *storage positions*, for the Nd_a deployable agents to remain on their associated carrier agent. These positions are linked to all other positions in the problem graph by instantly traversable edges. The conditions for using these edges are specified later in this section.

To define the CSP carrier agents graph, we need to add 2 edges for the graph's entry and exit positions:

$$\forall a \in C, E'_a = E_a \cup \{(v^s, Et_a), (Ex_a, v^e)\} \quad (3)$$

To define the CSP deployable agents graph, we also need to add edges to the storage positions. Each deployable agent has access to $Nd_a + 1$ deployment vertices, to perform at most Nd_a deployments. These vertices are linked to all the vertices of the problem graph, as well as to the start and end vertices.

To sum up, the edges E'_a of the CSP graph G'_a are:

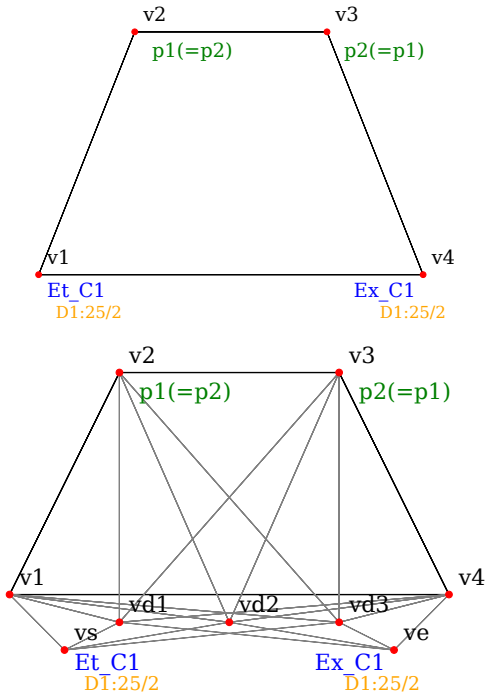


Figure 2: From a problem graph (top) to a CSP graph (bottom)

$$\begin{aligned}\forall a \in D, E'_a = E_a \\ \cup \{(v^s, v^n_d) | n \in [\![n^d - Nd_a, n^d]\!]\} \\ \cup \{(v^n_d, v^e) | n \in [\![n^d - Nd_a, n^d]\!]\} \\ \cup \{(v, v^n_d) | v, n \in V \times [\![n^d - Nd_a, n^d]\!]\} \\ \cup \{(v^n_d, v) | v, n \in V \times [\![n^d - Nd_a, n^d]\!]\}\end{aligned} \tag{4}$$

Figure 2 illustrates the augmentation of a problem graph into a CSP graph.

Edge traversal duration $Tt'_{a(v,v')}$ defined for E'_a are the same as the original problem. Time values on the edges of the CSP graph are defined for coherent time calculations: the mission starts with a time equal to 1, so edges coming from the start position and going to the end position have a crossing time of 1. Deployments are modeled by an edge that can be crossed instantly. The conjunction of these definitions is:

$$\begin{cases} e \in E_a & \implies Tt'_{ae} = Tt_{ae} \\ v^s \in e \vee v^e \in e & \implies Tt'_{ae} = 1 \\ e \notin E_a \wedge v^s \notin e \wedge v^e \notin e & \implies Tt'_{ae} = 0 \end{cases} \tag{5}$$

Constraints Encoding

Path integrity The uniqueness of the path followed by each agent is enforced by a flow constraint. We define $\mathcal{F}_{a(v,v')} \in [\![0,1]\!]$ the flow representing the path of the agent a on the edge (v, v'). Thus, ensuring flow consistency is equivalent to compare incoming, outgoing, and balance flows. To do so, for each position, we have to constrain incoming and outgoing flows to be equal to balances.

For all agents, the balance is 1 at graph entry, -1 at the exit, and 0 otherwise. In our case, for all a, $B_{av^s} = 1$, $B_{av^e} = -1$, otherwise $B_{av} = 0$.

$$\sum_{(v',v) \in E'_a} \mathcal{F}_{a(v',v)} - \sum_{(v,v'') \in E'_a} \mathcal{F}_{a(v,v'')} = B_{av} \tag{6}$$

Time Propagation The primary objective of the mission plan is to schedule the various agents' actions: a time metric based on the agent's achievements is therefore needed. We represent time with two variables $\mathcal{T}_{av}, \mathcal{D}_{av} \in [\![0, tmax]\!]^2$ that represent the time of arrival and the time spent on the vertex v for the agent a.

$$\mathcal{T}_{av} = \sum_{(v',v) \in E'_a} \mathcal{F}_{a(v',v)} (\mathcal{T}_{av'} + \mathcal{D}_{av'} + Tt'_{a(v',v)}) \tag{7}$$

$$\mathcal{T}_{av} = 0 \implies \mathcal{D}_{av} = 0 \tag{8}$$

$\mathcal{T}_{av} = 0$ indicate that vertex v is not in the path of agent a in the found solution. We add a constraint on D_{av} so that it cannot stay or perform tasks at positions that are not on its path. This also enforces the presence of all the carrier agents on the first position at $t = 1$.

Tasks constraints The variable $\mathcal{R}t_{am} \in [\![0,1]\!]$ indicates task p completion by the agent a. We first add the constraint that p has to be completed by one agent that is allowed to achieve it (Eq. 9).

$$\sum_a \mathcal{R}t_{ap} = 1 \wedge (a \in Rt_p \implies \mathcal{R}t_{ap} = 0) \tag{9}$$

$$\mathcal{R}t_{ap} \implies \mathcal{D}_{aPt_p} \geq Dt_p \tag{10}$$

$$\mathcal{R}t_{ap} \implies \mathcal{T}t_p = \mathcal{T}_{aPt_p} \wedge \mathcal{T}_{aPt_p} \in Wt_p \tag{11}$$

$$p \neq p' \wedge \mathcal{T}t_p = \mathcal{T}t_{p'} \implies \mathcal{R}t_{ap} + \mathcal{R}t_{ap'} \leq 1 \tag{12}$$

In planning model, we consider that completing task p requires the agent to remain on the associated vertex longer than the task duration (Eq. 10)

We define the variable $\mathcal{T}t_p \in [\![0, tmax]\!]$ as the starting time of realization of the task p. This variable is used for synchronization constraint verification. In addition, we have to ensure that the task satisfies the appropriate time window (Eq. 11).

Two tasks cannot be performed simultaneously by the same agent at the same position (Eq. 12) .

Coordination between tasks The succession of two tasks (p, p') is ensured by starting the second task after the completion the first one 13. The synchronization of two tasks (p, p') is ensured by starting both tasks at the same time 14. These two constraints are exclusives.

$$(p, p') \in S \Rightarrow \mathcal{T}t_p + \mathcal{D}t_p \leq \mathcal{T}t_{p'} \tag{13}$$

$$(p, p') \in P \Rightarrow \mathcal{T}t_p = \mathcal{T}t_{p'} \tag{14}$$

Temporal exclusion constraint The variable $\mathcal{R}r_{aq}$ defines the realization of the restriction q for the agent a:

$$a \notin Ar_q \implies \mathcal{R}r_{aq} = 1 \tag{15}$$

$$\mathcal{R}r_{aq} = 1 \iff \mathcal{T}_{aPr_q} + \mathcal{D}_{aPr_q} < Wr_q \vee \mathcal{T}_{aPr_q} > Wr_q \tag{16}$$

To satisfy this constraint, the agent must pass through the associated vertex Pr_q before or after the temporal exclusion window Wr_q, which is expressed by the disjunction of Eq. (16).

Position discovery Some agents $a \in F$ are not allowed to go through a vertex first. For these agents, we introduce the constraint for all the vertices in the problem:

$$\begin{array}{l} \forall (a,v) \in F \times V, \mathcal{T}_{av} > 1 \\ \quad \implies \exists a' \in D \cup C \backslash F, 0 < \mathcal{T}_{a'v} < \mathcal{T}_{av} \end{array} \tag{17}$$

Makespan The makespan calculation is based on the time taken by agents to go from v^s to v^e. By definition of flows, these are the first and last points taken by each agent and $\mathcal{T}_{av^s} + \mathcal{D}_{av^s} = 0$. However, it is necessary to subtract the artificial delay of 2 time units introduced by the CSP model to obtain the correct value:

$$\mathcal{M} = \max_{a \in C} (\mathcal{T}_{av^e} + \mathcal{D}_{av^e}) - 2 \tag{18}$$

Deployable Agents Specific Constraints

We introduce $\mathcal{D}p_{av} \in \{-C_a, 0, C_a\}$ which is $-C_a$ for the recovery of agent a by agent C_a at vertex v, C_a for the deployment of agent a at vertex v by agent C_a, 0 otherwise. We introduce a constraint to ensure that the passage from a position introduced for the CSP model to a position of the problem is marked as a deployment or recovery (remember Boolean are encoded as integer values, so $(Dp_{av} > 0) \in \{0, 1\}$):

$$\forall a, v \in D \times V, \sum_{v' \in Vd} \mathcal{F}_{a(v,v')} = (\mathcal{D}p_{av} < 0) \tag{19}$$

$$\forall a, v \in D \times V, \sum_{v' \in Vd} \mathcal{F}_{a(v',v)} = (\mathcal{D}p_{av} > 0) \tag{20}$$

For any deployment or recovery position $a \in C$, it is necessary to impose the satisfaction of synchronized time actions between the 2 agents:

$$\mathcal{D}p_{av} \neq 0 \implies \begin{cases} \mathcal{T}_{av} > 0 \wedge \mathcal{D}_{av} \geq Dd_a \wedge \mathcal{D}_{C_a v} \geq Dd_a \\ \mathcal{T}_{av} + \mathcal{D}_{av} = \mathcal{T}_{C_a v} + \mathcal{D}_{C_a v} \end{cases} \tag{21}$$

Deployment actions are exclusive, so we introduce the following constraint to ensure the uniqueness of these actions at each position:

$$\forall a^1, a^2 \in D^2, a^1 \neq a^2 \implies \text{abs}(\mathcal{D}d_{a^1}) \neq \text{abs}(\mathcal{D}d_{a^2}) \tag{22}$$

A carrier agent can perform a task and dispatch an agent at the same location. In this case, the time spent at the position must be at least the sum of both actions.

$$\begin{array}{l} \forall a \in C, \mathcal{R}t_{ap} \wedge \text{abs}(\mathcal{D}p_{a'Pt_p}) = a \\ \quad \implies \mathcal{D}_{aPt_p} > Dt_p + Dd_a \end{array} \tag{23}$$

Remember we assume that tasks are executed when entering a position, and deployments or recoveries just before leaving it. This implies that on a given position, a deployable agent can perform a task and be recovered, but not be deployed and immediately perform a task. The former constraint is therefore slightly different for deployable agents:

$$\forall a \in D, \mathcal{R}t_{ap} \wedge \mathcal{D}p_{aPt_p} < 0 \implies \mathcal{D}_{aPt_p} > Dt_p + Dd_a \tag{24}$$

$$\forall a \in D, \mathcal{D}p_{aPt_p} > 0 \implies \mathcal{R}t_{ap} = 0 \tag{25}$$

Maximum deployment time By construction, the storage vertices in *Vd* are identically connected to the start point, the finish point and all other points in the problem. We enforce an order of passage through *Vd* to avoid different equivalent solutions which swap the different times of passage through the *Vd*:

$$\forall a, n \in C \times [\![1, \text{Card}(Vd) - 1]\!], \mathcal{T}_{av^d_n} \leq \mathcal{T}_{av^d_{n+1}} \tag{26}$$

To ensure that deployment times are less than the maximum allowed time, we constrain the time between two positions introduced for the CSP:

$$\begin{array}{l} \forall a, n \in C \times [\![1, \text{Card}(Vd) - 1]\!], \\ \mathcal{T}_{av^d_{n+1}} - (\mathcal{T}_{av^d_n} + \mathcal{D}_{av^d_n}) < Fd_a \end{array} \tag{27}$$

This constraint applies indifferently to all agents, due to the definition of the edges linking the *Vd*, which imposes the maximum number of deployments.

Communication Constraints

To satisfy the communication constraints with the central agent a^r, we proceed by recurrence on the positions taken by the other agents. First, we assert the connection at time 1:

$$\forall a \in C, \mathcal{T}_{av} = \mathcal{T}_{a^r v^r} = 1 \implies (v, v^r) \in Er \tag{28}$$

Then, for each other agent, at each new considered position, we must ensure that it is communicating with the central agent. To do this, we first check that there are candidate positions for communication. A position is a candidate when agent a has arrived at position v before agent a' has arrived at position v', and it is possible to communicate between the 2 positions.

$$Pc(a, v, a', v') : 0 < \mathcal{T}_{av} \leq \mathcal{T}_{a'v'} \wedge (v', v) \in Er \tag{29}$$

To enforce that the candidate position is communicating, we need to check that agent a is still on v when a' arrives at v'

$$Ps(a, v, a', v') : \mathcal{T}_{av} + \mathcal{D}_{av} > \mathcal{T}_{a'v'} \tag{30}$$

or that agent a goes to a position v'' that is also communicating with v':

$$Pg(a, v, a', v') : v'' \in V, \begin{cases} \mathcal{F}_{a(v,v'')} = 1 \\ (v, v'') \in Er \\ \mathcal{T}_{a'v'} < \mathcal{T}_{av''} \end{cases} \tag{31}$$

The communication constraint from all agents a' to the central agent a^r at each position entry defined in the problem is therefore the conjunction:

$$\begin{aligned} &\forall a \in D \cup C \backslash a^r, \forall v \in V, \mathcal{T}_{av} > 0 \\ \implies &\exists v^r \in V, \begin{cases} \mathcal{T}_{a^r v^r} > 0 \\ Pc(a^r, v^r, a, v) \\ Ps(a^r, v^r, a, v) \vee Pg(a^r, v^r, a, v) \end{cases} \end{aligned} \tag{32}$$

In a second step, we need to ensure this constraint for the central agent when it moves. We need to add a predicate: agent a' is on a CSP introduced position v' when agent a enters position v.

$$Pf(a, v, a') : \exists v' \in V_d, \begin{cases} \mathcal{T}_{av} \geq \mathcal{T}_{a'v'} \\ \mathcal{T}_{av} \leq \mathcal{T}_{a'v'} + \mathcal{D}_{a'v'} \end{cases} \tag{33}$$

The overall constraint to ensure communication when the central agent a^r enters a new position is therefore the disjunction between the presence at a storage position and the conjunction of the previously asserted communications:

$$\begin{aligned} &\forall v^r \in V, \mathcal{T}_{a^r v^r} > 0 \implies \forall a \in D \cup C \backslash a^r, \\ &\left| \begin{array}{l} Pf(a^r, v^r, a) \\ \begin{cases} Pc(a, v, a^r, v^r) \\ Ps(a, v, a^r, v^r) \vee Pg(a, v, a^r, v^r) \end{cases} \end{array} \right. \end{aligned} \tag{34}$$

Comparison with a Generic Task Planner

Resolution of the CSP

We have implemented our mathematical model on top of a pre-existing CSP solver. We have created a C++ component called ZORTAC which performs problem-to-CSP and CSP-solution-to-plan. ZORTAC uses MINIZINC (Nethercote et al. 2007) for intermediate constraint representation, and OR-TOOLS (Google LLC 2023) for problem-solving itself.

We observed that for our problem, the number and type of CPU used by OR-TOOLS do not have a significant impact on solving time and solution quality. For all the experiments, we use 4 CPU with 10 GB of RAM.

Model and Resolution With a Generic Task Planner

We compare the performance of our method with the generic task planner ARIES (Bit-Monnot 2023; Godet and Bit-Monnot 2022). We chose ARIES as a state-of-the-art task planner that compiles the problem into a generic CSP model, and solves with its own CP-SAT solver, following an approach similar to ours. To model the problem for ARIES, we are using *unified-planning* library (Micheli et al. 2025). It allows modeling complex problems with temporal, numerical, and hierarchical aspects, at the frontier between PDDL2 (Fox and Long 2003; Edelkamp and Hoffmann 2004) and HDDL (Höller et al. 2020).

As ARIES solves problems formalized in PDDL and HDDL, our constraint model cannot be reused. So we modelled again the same problem with ARIES/PDDL approach (and thus compare our method with an alternative model and solver, as a whole). The problem formalization for ARIES is summarized here.

Because an agent can go to a specific position only once during the mission (a constraint imposed by the flow-based resolution scheme), we create for each agent $a \in C \cup D$ and each position $v \in V$ a unique task *on-v-a*. This task is responsible for bringing the agent a to the position v if needed and to optionally complete the task of this position.

For instance, Figure 3a shows the task for the agent C1 on position v3. One way to do the task is to do nothing with the *noop* method, meaning that the agent does not go through the position during the mission, which is equivalent to the implication 8. The other way is the *act* method, where the agent *move* from its previous position to v3 if there is a path and the communication is possible. Next, the agent stays *safe*, ensuring that the position v3 is not subject to any interdiction while the agent is present. Finally, the agent C1 can optionally *complete* the task p2 while staying *safe*.

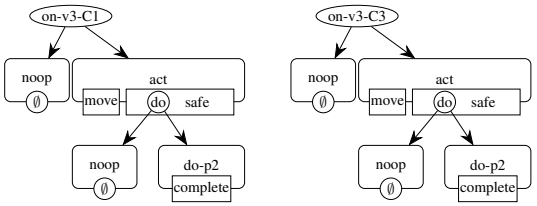
There is one exception for the *act* method when the agent is a carrier and the position is the entry position of the mission. In this case, it is not possible for the agent to *move* from a previous position, and so the *move* action is not present in this case.

One condition to *complete* the task is that no other agent is doing the task at the same time. Therefore, the *do-p2* methods of figures 3a and 3b are mutually exclusive, one of the agents must choose the *noop* method.

In the case where there is no task to complete on the position or if the agent is not allowed to complete the task, *e.g.*, the agent C1 on the position v5, the *do* task is omitted as shown in Figure 3c, reducing the search space.

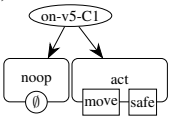
Moreover, a deployable agent d_a can be deployed at most Nd_a time. For each $d_a \in D$, for $i \in [\![1, Nd_a]\!]$, we create a task *deployment-d_a-i* responsible for deploying and recovering the agent d_a for the i^{th} time. One way to do the task is to do nothing with the *noop* method, meaning that the agent is deployed at most $i - 1$ times during the mission. The other way is the *act* method, where the agent is *deployed* from its carrier and *landed* (recovered) on it after. This structure ensures that the agent is not deployed at the end of the mission. The two *deploy* and *land* actions are temporally constrained to ensure that the agent is deployed for at most Fd_a units of time. The figure 3d shows the first deployment of the deployable agent D1.

Finally, each carrier $c_a \in C$ is forced to be at its exit position Ex_{c_a} at the end of the mission, and each task $t \in T$ to be completed.

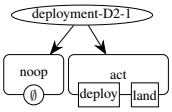


(a) Task on v3 for C1.

(b) Task on v3 for C3.



(c) Task on v5 for C1.



(d) Deployment 1 of D2.

Figure 3: Some tasks used to model the problem for ARIES

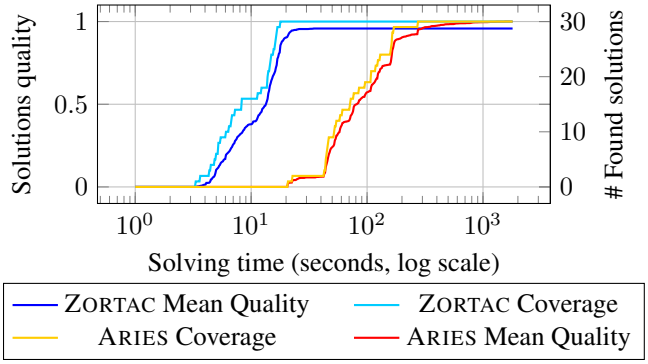


Figure 4: Solutions quality repeatability over 30 solving of a representative instance (1800 seconds timeout)

Test Case Results

To test the repeatability of finding solutions in a given time, we solve the representative problem of Figure 1 30 times. Figure 4 compares the mean solution quality and the number of solved instances.

Quality is normalized with respect to the highest quality found by all planners. Given the additional assumptions made for our model, the optimality results found by ZORTAC and ARIES are not identical. ARIES generally produces better quality plans than ZORTAC thanks to a finer time discretization. The main differences are the time step of each planner and the temporal position of the tasks on the position.

Contrary to what would be expected, integrating the assumptions of ZORTAC in the model of ARIES would *not* speed up its resolution and ARIES was systematically slower, even with a strict implementation of the ZORTAC. ARIES however benefits from the interdictions of multiple visits of the same vertex, which avoids the need for a recursive move decomposition (Godet, Lesire, and Bit-Monnot 2023).

For this problem, the quality criteria is 6.3% in favor of ARIES. Figure 4 shows that ZORTAC is capable of finding an initial solution and optimizing it consistently, and that

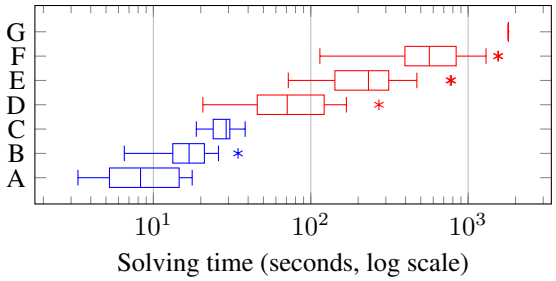


Figure 5: Key solving moments over 30 solving of a representative instance (1800 seconds timeout). ZORTAC First Plan (A), Last Plan (B), Optimal Proof (C). ARIES First Plan (D), Equivalent Quality Plan (E), Last Plan (F), Optimal Proof(G).

ARIES is also capable of finding some solution repeatedly. In the allotted time of 1800 seconds, all resolution instances found an optimal solution. But while ZORTAC succeeded in proving the optimal solution in regard to its model on all instances, ARIES failed to prove the optimality of its solutions on all instances.

The analysis of the resolution time to find the first plan, last plan and first proved optimal plans confirm ZORTAC's efficiency compared with ARIES' performance (Figure 5). The proof of the solution optimality is quickly found by ZORTAC whereas ARIES fails to prove the optimal in the allotted time. We add a metric (Equivalent Quality Plan) which is the time needed by ARIES to find a solution of equivalent quality to ZORTAC's optimal. This solution is found before ARIES' best solution, but well after the time needed for ZORTAC to prove that it is its optimal accordingly to its model hypothesis.

Results on a Generated Problem Set

To further characterize the performance of our models, we generated problem instances with random variations according to an experiment design process. We vary the values of different parameters, as shown in Table 4. The range of parameter values was chosen to produce ZORTAC satisfiable and solvable plans in 1800 seconds. This is much longer than the time required in the field, but it does allow us to properly assess the differences between the two approaches.

The parameters that have the greatest impact on the average resolution time are the number of tasks for deployable agents, and the number of positions. The values of the remaining parameters have a strong impact on the satisfiability of the automatically generated plan.

The random problem generation starts with positioning vertices along a 2D grid. Vertices are then randomly pruned according to a pruned position ratio of positions so as to not have only grid. Then, we ensure that all vertices are linked by at least 2 edges to improve the feasibility of the generated instance. Finaly, tasks are randomly associated to positions, and synchronized tasks are randomly selected.

For tasks that can only be performed by deployable

Parameters	Values
Number of positions	9,16,25,36
Number of carrier agents	2,3,5
Number of Deployable agents	1,3,5
Position with tasks ratio (%)	0,25,50
Synchronized task ratio (%)	20
Pruned position ratio (%)	10
Communication constraints	true, false
Number of tasks accessible by flying agents	0,2,5

Table 4: Parameters and associated values for randomly generated instances

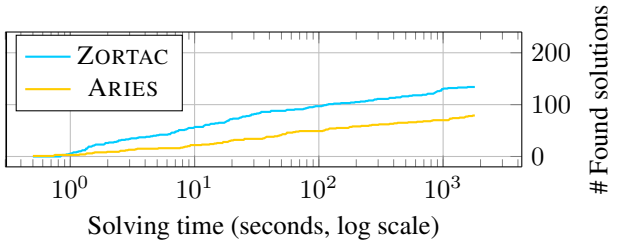


Figure 6: Coverage over 216 randomly generated instances populated with values of Table 4 (1800 seconds timeout)

agents, we draw enough position pairs to reach the required number of positions only accessible by deployable agents. We place a new position in the middle of the two drawn positions, and connect it to these two positions with routes only accessible to deployable agents. If communication constraints are imposed, the communication graph is simply defined by checking agent distance: agents closer than 2.5 times the grid edges distance are in communication. Departure and arrival positions are common to all agents, and are on opposite sides of the field. The set of parameter levels is generated by Generalized Subset Designs (GSD) with a reduction coefficient of 3. 216 instances are generated.

Domain coverage over solving time is displayed in Figure 6. ZORTAC quickly proves that 25 problems are unsatisfiable. This can happen when the instance configuration requires agents to move back and forth between different positions (unsatisfiability detection was not checked with ARIES). On the 46 plans where ZORTAC and ARIES proved the optimality with respect to their model, the average quality deviation was 7.9%. The largest quality deviation is 35% for an instance with a expecialy short optimal solution quality. It is difficult to generalize about these gaps, as ARIES has a hard time proving optimality. Small plans with low absolute but high relative gaps are overrepresented. We do not find a strong correlation between the different plan generation parameters and their complexity in finding an efficient solution.

The evolution of domain coverage depicted in Figure 6 exhibits an increasing gap between the two solvers. ZORTAC is better at producing first solutions than ARIES.

Constraint models, datasets and results are available at https://gitlab.laas.fr/esiboulet/icaps25_dataset_code

Discussion

Performance of Other CSP Solvers

We have measured the effect of choosing another solver instead of OR-TOOLS in ZORTAC: We have used the CSP solvers OR-TOOLS v9.11 (Google LLC 2023), CHOCO-SOLVER v4.10 (Prud'homme and Fages 2022), GECODE v6.2 (Schulte and Stuckey 2008), CHUFFED v0.13.2 (Chu et al. 2018), and CBC v2.10 (Forrest et al. 2024), all controlled by MINIZINC v2.8 (min 2023). The results are shown Table 5.

We have found that the solvers are ordered by performance as in the Minizinc Challenge (Stuckey et al. 2014), thus confirming experimental performance comparison. Moreover, we note the good performance of the planning method used by ARIES. This shows that it is the coupling of our model to OR-TOOLS that enables optimal performance, and that our model itself is not sufficient to provide good solving performance: it needs to be coupled with OR-TOOLS to surpass ARIES performances. Aries is based on its own CP-SAT solver to find a plan, and it is currently impossible to use any other solver with it.

The ZORTAC model has been designed to work with OR-TOOLS, and the performances observed during model development led to the use of a flow-based constraint model. The resolution of this type of constraint is not equivalent in all CSP solvers. It is likely that the use of path-finding methods such as path search or sub-circuit search would modify the performance differences found between the different solvers.

ZORTAC	62.5%
ARIES	36.6%
ZORTAC-CHOCO-SOLVER	25.3%
ZORTAC-GECODE	13.9%
ZORTAC-CHUFFED	5.56%
ZORTAC-CBC	3.70%

Table 5: Coverage over 216 randomly generated instances populated with values of Table 4 in 1800 seconds

Performance on a Realistic Scenario

We assessed the performances of our model on a realistic scenario, inspired by a challenge organized by one country ministry of defense. The scenario has 17 positions, 3 of which are only accessible to deployable agents. One agent carrying 2 deployable agents does not need to achieve any task on any position, and 2 carrier agents do. The average graph degree is higher than that of the randomly generated instances (Figure 7). Deployable agents have a larger navigation graph than carrier agents. There are 7 actions to be performed, including one on a position that only a deployable agent can reach, and two that must be performed synchronously.

Solving this specific instance up to good quality, takes 90min (5400s) for ZORTAC. This long solving time makes the approach suitable for mission preparation. However, alternative methods will be necessary for plan adaptation dur-

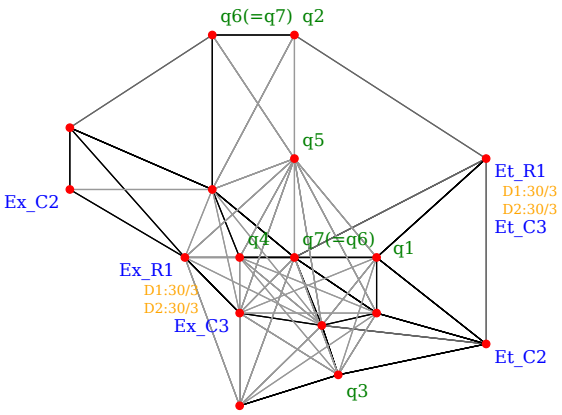


Figure 7: Realistic problem navigation graph and tasks

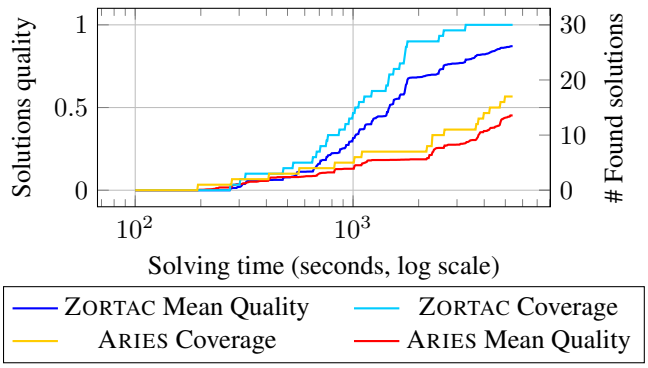


Figure 8: Solutions quality repeatability over 30 solving of a realistic instance (5400 seconds timeout)

ing execution, in order to dynamically take into account unforeseen events and contingencies. As depicted Figure 8, our model (ZORTAC with OR-TOOLS) allows automated planning of such realistic instances, which was not possible with classical planning approaches such as ARIES.

Conclusion

We presented a progression mission problem, a specific routing problem enriched by the introduction of the concept of deployable agents and the consideration of communications constraints. We provide a CSP model for this problem and compare its performance with that of an automated planning approach. The relevance of our approach is assessed for a realistic scenario, demonstrating that our domain-dependent model coupled with OR-TOOLS efficiently yields valid plans for complex mission progressions.

Various improvements are foreseen. For instance, domain-related dedicated heuristics could drastically speed up the resolution: learning such heuristics with graph neural networks may be a relevant approach. Providing explanations of the unsatisfiability of problems would be a key asset to the approach, but achieving this in a reasonable time is certainly a difficult issue.

Finally, a plan execution will eventually reveal discrepancies with respect to planning model. Analyzing the produced plans to identify admissible delays would enhance its robustness with respect to such discrepancies, *e.g.* by using a simple temporal network with uncertainty plan. Also, we believe that it is possible to further enrich the model to cope with more or less predictable evolutions of the mission, as in the case of deploying wildfire countermeasures against a dynamic firefront. In such missions, the use of deployable agents that can gather up-to-date information would be very beneficial.

References

2023a. Challenge CoHoma, RETEX sur la 2e édition. Technical report, Armée de Terre.

2023b. La robotique dans l'armée de Terre, CoHoMa 2023. Technical report, Armée de Terre.

2023. MiniZinc. http://www.minizinc.org/. Accessed: 2024-03-12.

Bevacqua, G.; Cacace, J.; Finzi, A.; and Lippiello, V. 2015. Mixed-Initiative Planning and Execution for Multiple Drones in Search and Rescue Missions. *Proceedings of the International Conference on Automated Planning and Scheduling*, 25(1): 315–323.

Bit-Monnot, A. 2023. Experimenting with Lifted Plan-Space Planning as Scheduling: Aries in the 2023 IPC. In *2023 International Planning Competition at the 33rd International Conference on Automated Planning and Scheduling*. Prague, Czech Republic.

Bredström, D.; and Rönnqvist, M. 2008. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European journal of operational research*, 191(1): 19–31.

Chu, G.; Stuckey, P. J.; Schutt, A.; Ehlers, T.; Gange, G.; and Francis, K. 2018. Chuffed, a lazy clause generation solver.

De Petris, P.; Khattak, S.; Dharmadhikari, M.; Waibel, G.; Nguyen, H.; Montenegro, M.; Khedekar, N.; Alexis, K.; and Hutter, M. 2022. Marsupial walking-and-flying robotic deployment for collaborative exploration of unknown environments. In *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 188–194. IEEE.

Edelkamp, S.; and Hoffmann, J. 2004. PDDL2. 2: The language for the classical part of the 4th international planning competition.

Ferreira, B. A.; Petrović, T.; Orsag, M.; Martínez-de Dios, J. R.; and Bogdan, S. 2024. Distributed allocation and scheduling of tasks with cross-schedule dependencies for heterogeneous multi-robot teams. *IEEE Access*.

Forrest, J.; Ralphs, T.; Vigerske, S.; Santos, H. G.; Forrest, J.; Hafer, L.; Kristjansson, B.; jpfasano; EdwinStraver; Jan-Willem; Lubin, M.; rlougee; a andre; jpgoncal1; Brito, S.; h-i gassmann; Cristina; Saltzman, M.; tosttost; Pitrus, B.; MATSUSHIMA, F.; Vossler, P.; SWGY, R. .; and to st. 2024. coin-or/Cbc.

Fox, M.; and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *JAIR 2003*, 20: 61–124.

Godet, R.; and Bit-Monnot, A. 2022. Chronicles for Representing Hierarchical Planning Problems with Time. In *ICAPS Hierarchical Planning Workshop (HPlan)*. Singapore, Singapore.

Godet, R.; Lesire, C.; and Bit-Monnot, A. 2023. Multi-Robot Task Planning to Secure Human Group Progress. In *ECAI Agents and Robots for reliable Engineered Autonomy Workshop (AREA)*, volume 391, 113–126.

Google LLC. 2023. OR-Tools. https://developers.google.com/optimization/. Accessed: 2024-03-12.

Guastalla, A.; Aringhieri, R.; and Hosteins, P. 2024. The Team Orienteering Problem with Service Times and Mandatory & Incompatible Nodes. *arXiv preprint arXiv:2410.12368*.

Guettier, C. 2007. Solving planning and scheduling problems in network based operations. *Proceedings of Constraint Programming (CP)*.

Gunawan, A.; Lau, H. C.; and Vansteenwegen, P. 2016. Orienteering Problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2): 315–332.

Höller, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; and Alford, R. 2020. HDDL: An Extension to PDDL for Expressing Hierarchical Planning Problems. In *AAAI 2020*, volume 34, 9883–9891.

Kenzin, M.; Bychkov, I.; and Maksimkin, N. 2019. Autonomous coordination of heterogeneous vehicles for persistent monitoring problem with route and fuel constraints. In *ICCS-DE*, 78–87.

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

Micheli, A.; Bit-Monnot, A.; Röger, G.; Scala, E.; Valentini, A.; Framba, L.; Rovetta, A.; Trapasso, A.; Bonassi, L.; Gerevini, A. E.; Iocchi, L.; Ingrand, F.; Köckemann, U.; Patrizi, F.; Saetti, A.; Serina, I.; and Stock, S. 2025. Unified Planning: Modeling, manipulating and solving AI planning problems in Python. *SoftwareX*, 29: 102012.

Nethercote, N.; Stuckey, P. J.; Becket, R.; Brand, S.; Duck, G. J.; and Tack, G. 2007. MiniZinc: Towards a standard CP modelling language. In *International Conference on Principles and Practice of Constraint Programming*, 529–543. Springer.

Prud'homme, C.; and Fages, J.-G. 2022. Choco-solver. *Journal of Open Source Software*, 7(78): 4708.

Schulte, C.; and Stuckey, P. J. 2008. Efficient constraint propagation engines. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 31(1): 1–43.

Shahidi, A.; Ramezanian, R.; and Shahparvari, S. 2022. A greedy heuristic algorithm to solve a VRP-based model for planning and coordinating multiple resources in emergency response to bushfires. *Scientia Iranica*.

Shiroma, P. M.; and Campos, M. F. M. 2009. CoMutaR: A framework for multi-robot coordination and task allocation. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 4817–4824.

Siboulet, E.; Bit-Monnot, A.; des Graviers, M.-E. C.; Yelloz, J.; Guettier, C.; and Lacroix, S. 2024. Plan Generation for Multi-Robot Missions Requiring Active Operator Involvement. In *ICAPS'24 Workshop on Planning and Robotics (PlanRob)*.

Stuckey, P. J.; Feydy, T.; Schutt, A.; Tack, G.; and Fischer, J. 2014. The MiniZinc Challenge 2008–2013. *AI Magazine*, 35(2): 55–60.

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

Zlot, R. M. 2006. *An auction-based approach to complex task allocation for multirobot teams*. Ph.D. thesis, Carnegie Mellon University, The Robotics Institute.