Configuration of Heterogeneous Agent Fleet: a Preliminary Generic Model

Thomas Pouré[1,†], Stéphanie Roussel[2,†], Elise Vareilles[1,3,*,†] and Gauthier Picard[2,†]

[1]*ISAE SUPAERO, Université de Toulouse, 10 avenue Édouard Belin, BP 54032 - 31055 Toulouse CEDEX 4, France*
[2]*DTIS, ONERA, Université de Toulouse, 2 avenue Édouard Belin, BP 74025 - 31055 Toulouse CEDEX 4, France*
[3]*CGI / IMT Mines Albi, Université de Toulouse, allée des sciences, 81000 Albi, France*

Abstract
A multitude of autonomous agents – encompassing a range of technologies, including robots and drones – represent a crucial modern tool for the execution of a multitude of tasks, including surveillance, delivery and the saving of lives. In order to optimally utilise these agents, it is vital to configure each agent, the composition of the entire fleet of agents and the mission plan associated with each agent in the most effective manner possible. The following article presents a knowledge model for the configuration of a fleet of heterogeneous agents, encompassing the three levels of configuration: agent configuration, agent fleet configuration, and mission plan configuration. It explicitly delineates the relationships between these three configuration levels, thereby facilitating rapid, efficient, robust, and simultaneous configuration. A toy problem illustrates our first proposals.

Keywords
Multi-level Configuration, Autonomous Agent, Knowledge Formalisation, Heterogeneous Fleet,

1. Introduction

With the increasing autonomy of drones and robots, fleets of agents are now being used for many different types of missions, such as exploration, rescue, disaster relief, civil and military security. In this article, the term "agent" is used to refer to any system that is capable of acting autonomously in a variety of environments, including ground, water, and air. The term encompasses a diverse range of platforms, including quadrupeds, bi-blades, underwater rockets, and others. Additionally, the term "agent" encompasses a wide range of capabilities, including communication, rescue, and delivery. Therefore, the term "agent" can be used to describe a diverse range of systems, from household robots to high-tech stealth military drones. Some of these applications require heterogeneous agent fleets, i.e. with different platforms, capabilities, mobility and equipment. Such fleet of heterogeneous agents may or may not be coordinated autonomously to carry out the missions to which the fleet is dedicated. For example, an exploration mission may require the collaboration of ground agents with at least the ability to *Travel* and *Communicate*, and aerial agents with at least the ability to *Observe* and *Communicate*. The success of a multi-agent mission depends, among other things, on the configuration of the fleet executing it [1].

This paper addresses the problem of *multi-level configuration of heterogeneous agent fleets*, as presented in Fig. 1. By multi-level configuration, we mean the several interleaved problems that must be solved when setting up a fleet to carry out a mission. The first level is the simultaneous configuration of each agent (Agent Configuration Problem, ACP). The second consists in configuring the fleet itself (Fleet Configuration Problem, FCP), i.e. defining precisely what the composition of the fleet is. The final level is the fleet deployment problem in order to carry out dedicated missions in an efficient and robust way (Plan Configuration Problem, PCP). This multi-level configuration problem requires an analysis of the relationships between these three configuration levels, both upstream in fleet composition and downstream in fleet operation.

This multi-level configuration problem raises many research questions, such as:

- the representation/modeling of configuration knowledge (compact modeling language),
- eliciting constraints (what is allowed or forbidden) and criteria (what is preferable) that apply both to the fleet configuration and to each robot in it, and
- the development of algorithms to generate optimal or, at least, good-quality solutions.

This problem can be tackled in several ways. First of all, there is the question of how to express knowledge,

ConfWS'24: 26th International Workshop on Configuration, Sep 2–3, 2024, Girona, Spain
*Corresponding author.
†These authors contributed equally.
thomas.poure@student.isae-supaero.fr (T. Pouré); stephanie.roussel@onera.fr (S. Roussel); elise.vareilles@isae-supaero.fr (E. Vareilles); gauthier.picard@onera.fr (G. Picard)
https://onera.academia.edu/SRoussel (S. Roussel); https://pagespro.isae-supaero.fr/elise-vareilles/ (E. Vareilles); https://gauthier-picard.info/ (G. Picard)
0000-0001-7033-555X (S. Roussel); 0000-0001-6269-8609 (E. Vareilles); 0000-0002-9888-9906 (G. Picard)

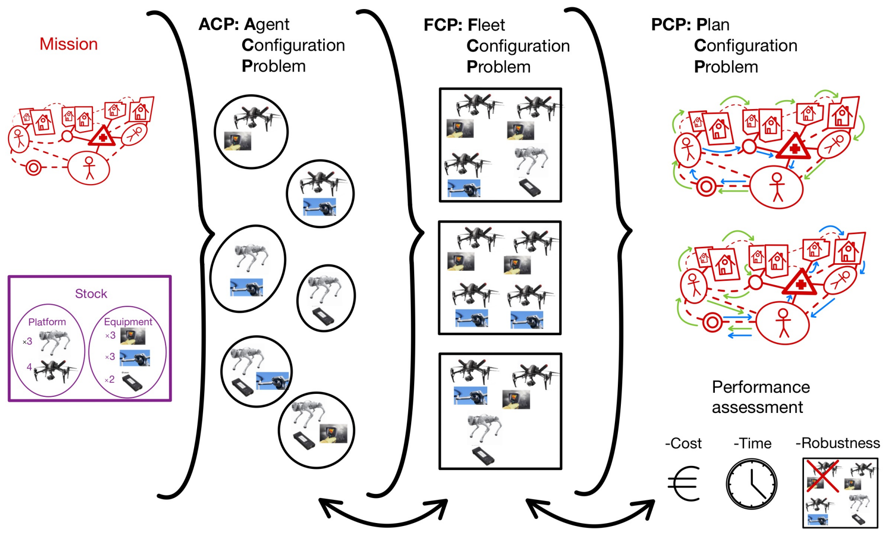

Mission
ACP: Agent Configuration Problem
FCP: Fleet Configuration Problem
PCP: Plan Configuration Problem
Stock
Platform
Equipment
×3
4
×3
×3
×2
Performance assessment
-Cost
-Time
-Robustness

Figure 1: Multi-level configuration of an heterogeneous fleet of robots.

constraints and preferences, both from the point of view of fleet configuration and from the point of view of performance and robustness in the context of mission [2]. Approaches such as constraint programming and multi-agent modeling [3, Chap.2 and 15] appear to be suitable candidates.

Following several works dedicated to Search and Rescue applications such as [4] and [5], a mission consists here in the execution of several tasks distributed on an intervention zone represented by a graph. A fleet and a plan of action are configured in order to accomplish the mission, i.e. successfully complete all the tasks. The performance of a fleet for a mission can be evaluated along several criteria: the global time required for performing all tasks, the fleet cost (platform, equipment), the fleet and the plan robustness (capacity of the fleet/the plan to support damages and complications), etc.

This article focuses on initial ideas for modeling the knowledge of this multi-level configuration problem of heterogeneous agents fleets. More precisely, we propose a formal modelling of the inputs of each level configuration problem, along with the decisions that have to be made. The formalization of constraints associated with each level are out of scope of this paper and are left for future work.

The paper is organized as follows. In Section 2, we formally describe the type of mission we consider. Then, Sections 3, 4 and 5 are respectively dedicated to the Agent Configuration Problem or ACP, the Fleet Configuration Problem or FCP and the Plan Configuration Problem or PCP. In each of these sections, we formally present the inputs of the problem, the associated decision variables and an illustrative example. Finally, we conclude and discuss future works in Section 6.

2. Mission

A mission allows to represent the several tasks that the agents have to perform and the graph on which they can move. The elements composing a mission can be represented in a UML diagram as illustrated in Fig. 2. Those elements are first briefly described and then formalized in a second step. In our work, we have made several assumptions on a mission. A mission is therefore:

- **deterministic**: the mission is perfectly known from the beginning and during the fleet's intervention, and agents cannot suffer from malfunctions,
- **static**: the mission remains static throughout the fleet's intervention. No edges or vertices are introduced or removed during the mission.

2.1. Description

A Mission is composed of the following elements.

- The location on which the agents can evolve is represented by a connected and non-directed Graph. Such a graph is composed of vertices (Vertex class), representing way points or places of interest in the mission context, and edges (Edge class), representing routes for moving.

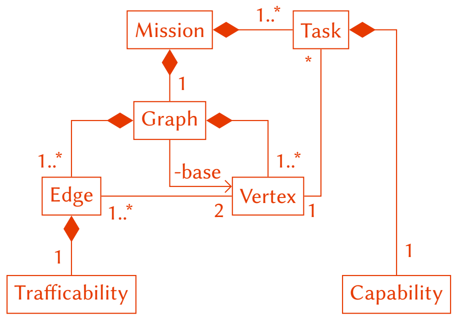


Figure 2: UML representation of a mission.

- One of the vertices is called the base, and is the location at which the agents start and finish their missions.
- The actions that the agents have to perform to achieve the mission are called Tasks. Each task is assigned to a single vertex, that represents the location at which it must be executed. A capability is associated to each task, it is the requirement to perform a task.

For the agent to be able to move through the graph and execute task, we define two additional classes.

- A Capability describes how an agent accomplishes the mission's tasks. More precisely, each task requires a single specific capability to be executed. Examples of capabilities are *observe*, *grab material*, *transport a injured person*, etc.
- One instance of Trafficability is associated to each edge, representing the edge practical environment for agent moves. Instances of Trafficability could be *Aerial*, *Terrestrial* or more fine-grained properties such as *Forest*, *Field*, *Street*, etc. Note that a trafficability could also be combinations such as *Terrestrial and Aerial*.

2.2. Formalization

We propose here a mathematical formalization of the mission, that can be used as input for the multi-level configuration problem.

A mission is a tuple $m = (V, E, T, TV, C, CT, R, RE)$ where:

- $V = (1, \dots, n_V)$ is the vector of vertices. We suppose that vertex with number 1 is the base.
- $E = (e_{i,j})_{i,j \in [1..n_V]^2}$ is the adjacency matrix of size n_V^2 that represents the connection between vertices V. For all vertices $i, j \in [1..n_V]^2$, $e_{i,j} = 1$ if there exists an edge between vertices i and j, $e_{i,j} = 0$ otherwise.
- $T = (1, \dots, n_T)$ is the vector of tasks that have to be performed during the mission.
- $TV = (tv_i)_{i \in [1..n_T]}$ is a vector of size n_T such that for all task $i \in [1..n_T]$, $tv_i \in [1..n_V]$ is the vertex the task i is assigned to.
- $C = (1, \dots, n_C)$ is the vector of capability types.
- $CT = (ct_i)_{i \in [1..n_T]}$ is a vector of size n_T, such that for each task $i \in [1..n_T]$ in $m, ct_i \in [1..n_C]$ represents the capability required by task i.
- $R = (1, \dots, n_R)$ is the vector of traficabilities.
- $RE = (re_{i,j})_{i,j \in [1..n_V]^2}$ is a matrix of size n_V^2, such as for each edge $(i,j) \in [1..n_V]^2$, $re_{i,j} \in [1..n_R]$ is the trafficability of the edge $e_{i,j}$ in m.

We call the graph associated to a mission m the pair (V, E). A mission m is said to be well-formed if the following assumptions hold:

- The graph does not contain any edge from a vertex to itself.
$$\forall i \in [1..n_V], e_{i,i} = 0 \qquad (1)$$
- The graph is non-oriented and the trafficability matrix is symmetrical.
$$E = E^T \qquad (2)$$
$$RE = RE^T \qquad (3)$$
- The graph is connected, i.e. from any two vertices i and j, there exists a path of edges connecting them. Formally, $\forall i, j \in [1..n_V]^2$, $\exists k \in \mathbb{N}^*$, $\exists (v_1, \dots, v_k) \in [1..n_V]^k$, such that:
$$v_1 = i, v_k = j \qquad (4)$$
$$\forall r \in [1..k-1], \quad e_{v_r, v_{r+1}} = 1 \qquad (5)$$
- For any vertices $i, j \in [1..n_V]^2$, $e_{i,j} = 1$ means that there is exactly one edge between vertices i and j.

In order to illustrate the notations defined previously, we consider the following toy example.

2.3. Toy Problem Mission

We define a simple Search & Rescue mission m, illustrated in Fig. 3, composed of the following elements.

- The vertices vector of locations is $V = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$, where 1 is the "base" (🏁), 2 is the "ruins" (🏢), and 3 is the "aid camp" (➕).

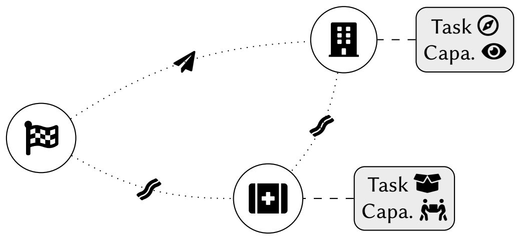


Figure 3: Illustration for Example 2.3. Three locations are considered: a base (🏁), ruins (🏢) to explore, and an aid camp (🏥) to supply. Moving from the base to the ruins requires an aerial agent (✈), while moving to the camp requires a terrestrial agent (∫).

- The edges matrix of paths is $E = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$. For instance, $e_{1,3} = 1$ holds, meaning that it is possible to directly go from vertex 1 ("base") to vertex 3 ("aid camp").
- The tasks vector is $T = \begin{pmatrix} 1 & 2 \end{pmatrix}$, where 1 is "explore the ruins" (🧭), and 2 is "deliver supplies" (📦).
- The assignment of tasks to the vertices is the vector $TV = \begin{pmatrix} 2 & 3 \end{pmatrix}$, representing that task 1 ("explore the ruins") and task 2 ("deliver supplies") must respectively be executed in vertex 2 ("ruins") and vertex 3 ("aid camp").
- The capabilities vector is $C = \begin{pmatrix} 1 & 2 \end{pmatrix}$, where 1 is "carry" (🧍), and 2 is "observe" (👁).
- The assignment of capabilities to the tasks is the vector $CT = \begin{pmatrix} 2 & 1 \end{pmatrix}$, meaning that capability 2 ("observe") is required for task 1 ("explore the ruins") and capability 1 ("carry") is required for task 2 ("deliver supplies").
- The traficabilities are $R = \begin{pmatrix} 1 & 2 \end{pmatrix}$, where 1 is "terrestrial" (∫), and 2 is "aerial" (✈).
- The assignment of traficabilities to the edges is the matrix $RE = \begin{pmatrix} 0 & 2 & 1 \\ 2 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$. For instance the path $(1, 2)$ has the trafficability 2 ("aerial"), whereas the path $(2, 3)$ has the trafficability 1 ("terrestrial").

3. Agent Configuration Problem

In this section, we present the model associated with the Agent Configuration Problem (ACP), which consists in deciding agents' composition wrt. a catalog of platform types and equipment types, by using the notion of agent pattern.

3.1. Description

As illustrated in Fig. 4, an AgentPattern represents a type of robot or a type of drone that can act somewhat autonomously. Elements composing an agent pattern are divided as follows:

- Platform represents the skeleton of an agent pattern. Each agent pattern has a single platform.
- Each Platform is associated to a unique PlatformType representing the agent pattern skeleton type. Examples of such platform types could be *aerial*, *terrestrial*, *marine*. It would also be possible to consider more fine-grained platform types, such as *quadcopter* or *submarine*. The platform type limits and defines most of the agent pattern characteristics.
- Equipment represents the payload that can equip an agent pattern. An agent pattern can be equipped with several equipments.
- Each Equipment is associated to a unique EquipmentType, which represents the type of the equipment (e.g. camera, sensor, motor).
- Available PlatformTypes and EquipmentTypes are grouped in a Catalog.

An agent is able to interact with the mission throughout two connections to the mission description:

- Each Equipment instance has a set of Capability instances, allowing agents to execute tasks. If an agent pattern is equipped with an equipment that provides the capability associated to a task, then any agent following that pattern will be able to perform the task.
- Each PlatformType instance is associated with a set of Trafficability instances representing the types of environments it is compatible with. Consequently, an agent pattern is compatible with an edge if and only if the edge trafficability belongs to the agent pattern platform type set of compatible traficabilities.

3.2. Formalization

We first formalize the inputs of the agent configuration problem and then define the decision variables. We next present some assumptions on the problems we consider and finally illustrate the concepts on the toy example.

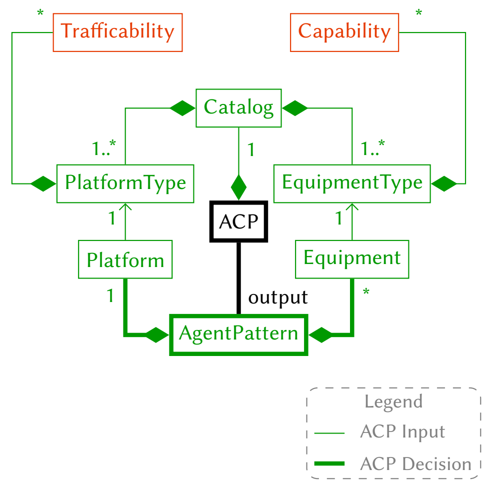


Figure 4: UML representation the ACP.

3.2.1. Inputs

Let m be a mission. A catalog on m is a tuple $cat_m = (P, Q, max_Q, RP, CQ)$ where:

- $P = (1, \dots, n_P)$ is the platform types vector,
- $Q = (1, \dots, n_Q)$ is the equipment types vector,
- $max_Q \in \mathbb{N}^*$ is an upper bound on the number of instances of each equipment type that can be carried by an agent pattern,
- $RP = (rp_{i,j})_{i,j \in [1..n_P] \times [1..n_R]}$ is the platform/traficability compatibility matrix of size $n_Q.n_R$. For each platform type $i \in [1..n_P]$ and each trafficability $j \in [1..n_R]$, $rp_{i,j} = 1$ if the platform type i is compatible with trafficability j. Otherwise, $rp_{i,j} = 0$.
- $CQ = (cq_{i,j})_{i,j \in [1..n_Q] \times [1..n_C]}$ is the equipment/capability relation matrix of size $n_Q.n_C$. For each equipment type $i \in [1..n_Q]$ and each capability $j \in [1..n_C]$, $cq_{i,j} = 1$ if the equipment type i provides the capability j. It equals 0 otherwise.

The catalog is the only input of the ACP.

3.2.2. Assumptions

A catalog cat should satisfy the following assumptions.

- **Task Feasibility.** For each task, there is at least one equipment type in the catalog that provides its capability, which translates into:

$$\forall j \in [1..n_T], \sum_{i=1}^{n_Q} cq_{i,ct_j} \geq 1 \tag{6}$$

- **Task Reachability.** For each task, there exists a platform type and a path from the base to the task's vertex such that the platform type is compatible with all the path's edges trafficabilities. Formally, $\forall i \in [1..n_T], \exists j \in [1..n_P], \exists k \in \mathbb{N}^*, (v_1, \dots, v_k) \in [1..n_V]^k$, *s. t.*

$$v_1 = 1, v_k = tv_i \tag{7}$$

$$\forall r \in [1..k-1]^2, \quad e_{v_r, v_{r+1}} = 1 \tag{8}$$

$$rp_{j, re_{v_r, v_{r+1}}} = 1 \tag{9}$$

Those two assumptions ensure that for each task in the mission, there exists an agent pattern compatible with the task perform it.

3.2.3. Decision Variables

We present here the decision variables that must be assigned a value when solving an ACP. To do so, we first formally define an *agent pattern*.

For a given catalog cat, an agent pattern is a tuple $\mathbf{a}_{cat} = (\mathbf{ap}, \mathbf{AQ})$ where :

- $\mathbf{ap}$ is an integer in $[1..n_P]$ that represents the platform of catalog cat associated with $\mathbf{a}_{cat}$.
- $\mathbf{AQ} = (\mathbf{aq_i})_{i \in [1..n_Q]}$ is the $\mathbf{a}_{cat}$ equipment vector of size n_Q. For all equipment type $i \in [1..n_Q]$, $\mathbf{aq_i}$ is an integer in $[1..max_Q]$ that represents the number of equipment type i present in $\mathbf{a}_{cat}$.

For a given catalog cat, the objective of ACP is to compute a tuple $\mathcal{T}_{cat} = (1, \dots, n_{\mathcal{T}})$ where each element is an index of an agent pattern, as defined previously, and $n_{\mathcal{T}}$ the number of elements in the tuple.

As we do not consider any constraint in this paper, there are $n_{\mathcal{T}} = n_P \cdot n_Q^{max_Q}$ possible agent patterns. In real world applications, the ACP should of course satisfy some constraints (e.g. max payload, mission's budget, etc.) and could optimize some criteria (e.g. cost minimization). This is out of scope of this paper, and so are the precise definitions of platform and equipment attributes related to them (such as weight, price, etc.). Note that even with constraints consideration, the vector $\mathcal{T}_{cat}$ might be too large to be exhaustively explored.

3.3. Toy Problem ACP

We consider the mission m defined in Subsection 2.3.

We define the catalog cat the following way.

- The platform types vector is $P = \begin{pmatrix} 1 & 2 \end{pmatrix}$, where 1 is "UAV" (🚁) and 2 is "rover" (🚙).
- The equipment types vector is $Q = \begin{pmatrix} 1 & 2 \end{pmatrix}$, where 1 is "camera" (📷) and 2 is "trunk" (🚚).

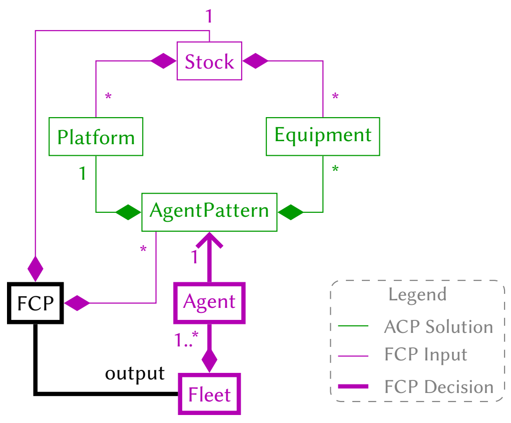


Figure 5: UML representation of the FCP.

- the maximum number for each equipment instance on an agent pattern is $max_Q = 1$.
- The platform/trafficability compatibility matrix is $RP = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. In this example, $rp_{1,2} = 1$ holds, meaning that platform 1 ("UAV") is compatible with the trafficability 2 ("aerial"). However, as $rp_{1,1} = 0$, platform 1 is not compatible trafficability 1 ("terrestrial").
- The equipment/capability relation matrix is $CQ = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. In this example, $rp_{1,1} = 1$ holds, meaning that the equipment 1 ("camera") provides the capability 2 ("observe"). However, $rp_{1,2} = 0$, which means that this equipment does not provide capability 2 ("carry").

The two following agent patterns belong to $\mathscr{T}_{cat}$:

- $\mathbf{a}_1 = (1, (1 \quad 0))$ is a UAV equipped with one camera and zero trunk.
- $\mathbf{a}_2 = (2, (1 \quad 1))$ is a rover equipped with one camera and one trunk.

There is a total of $n_{\mathscr{T}} = 6$ possible agent patterns ($\mathscr{T}_{cat} = (\mathbf{a}_1, \dots, \mathbf{a}_6)$).

4. Fleet Configuration Problem

In this section, we present the model associated with the Fleet Configuration Problem (FCP), which aims at deciding the composition of the fleet wrt. the available stock.

4.1. Description

Fig. 5 contains a UML representation of the Fleet Configuration Problem. The FCP class takes as an input the set of AgentPattern computed by the ACP, as presented in the previous section. Its output is a Fleet, *i.e.* a collection of Agents, where each Agent is associated to a unique AgentPattern.

In order to model the fact that equipment and platform are available in limited quantities, we define the class Stock. Such a class is associated to a set of Platforms and a set of Equipments. The FCP takes an instance of Stock as an input. Even if it is clear that the stock will impose hard constraints on FCP, the precise formalization of these constraints is left for future work.

4.2. Formalization

We first formalize the inputs of the agent configuration problem, then define the decision variables and illustrate the formalization on the toy example.

4.2.1. Inputs

Let *cat* be a catalog. The FCP associated to this catalog has two inputs:

- A stock associated with *cat*, denoted s_{cat}, and defined by a pair (P_s, Q_s) where:
  - $P_s = (p_i)_{i \in [1..n_P]}$ is a vector of size n_P such that for each platform type $i \in [1..n_P]$ in *cat*, $p_i \in \mathbb{N}^*$ defines how many type i platform instances are in the stock.
  - $Q_s = (q_j)_{j \in [1..n_Q]}$ is a vector of size n_Q such that for each equipment type $j \in [1..n_Q]$ in *cat*, $q_j \in \mathbb{N}^*$ defines how many type j equipment instances are in the stock.
- A vector of agents pattern $\mathscr{T}_{cat}$. Such a vector can for instance come from the output resulting from the ACP solving.

4.2.2. Decision Variables

Given a catalog *cat*, a stock s_{cat} on this catalog, and $\mathscr{T}_{cat}$ a vector of the agent patterns, a fleet is a tuple $\mathbf{f}_{s_{cat}, \mathscr{T}_{cat}} = (\mathbf{n_a}, \mathscr{A}_\mathbf{f})$ where:

- $\mathbf{n_a}$ is the size of the fleet.
- $\mathscr{A}_\mathbf{f} = (\mathbf{a_i})_{i \in [1..\mathbf{n_a}]}$ is the finite vector of size $\mathbf{n_a}$ of agents in the fleet such as, for each $i \in [1..\mathbf{n_a}]$, $\mathbf{a_i} \in [1..n_{\mathscr{T}}]$ is the index of the agent pattern of the agent i in the fleet.

Note that the model allows to have the same agent pattern present several times in $\mathscr{A}_\mathbf{f}$, representing the fact that there are some identical agents in the fleet.

4.3. Toy Problem FCP

We consider the mission m defined in Subsection 2.3, the catalog cat and the agent patterns $\mathscr{T}_{cat}$ defined in Subsection 3.3.

We define the stock s_{cat} the following way.

- The platform instance vector is $P_s = \begin{pmatrix} 2 & 1 \end{pmatrix}$, meaning that there are 2 instances of type 1 platform ("UAV" -) and 1 instance of type 2 platform ("rover" -) in the stock.
- The equipment instances vector is $Q_s = \begin{pmatrix} 2 & 1 \end{pmatrix}$. In this example, there are two instances of type 1 equipment ("camera" -) and one instance of type 2 equipment ("trunk" -) in the stock.

With this stock, it is possible to configure several fleets of agents. For instance, we define two fleets as follows:

- $\mathbf{f}^1_{s_{cat},\mathscr{T}_{cat}} = (1, (\mathbf{a}_2))$, is a fleet composed of a single agent with the pattern $\mathbf{a}_2$ (a rover equipped with one camera and one trunk - + +).
- $\mathbf{f}^2_{s_{cat},\mathscr{T}_{cat}} = (2, (\mathbf{a}_1, \mathbf{a}_2))$, is a fleet composed of two agents with the respective patterns $\mathbf{a}_1$ (a UAV equipped with one camera - +) and $\mathbf{a}_2$ (a rover equipped with one camera and one trunk - + +).

5. Plan Configuration Problem

In this section, we present the model associated with the Plan Configuration Problem (PCP), which aims at deciding the agents' positions and tasks all along the mission.

5.1. Description

The plan configuration is the last problem to solve in order to get a solution for the multi-level configuration problem. As illustrated on Fig. 6, it takes as input a Mission and a Fleet. Its output is a Plan which consists of an AgentPlan for each Agent in the fleet. For each agent in the fleet, an AgentPlan describes exhaustively at any given time step the position of the agent and the task currently executed, if any.

5.2. Formalization

We first formalize the inputs of the plan configuration problem and then define the decision variables.

5.2.1. Inputs

For a catalog cat, a stock on this catalog, s_{cat}, the PFD associated to this stock requires two additional inputs:

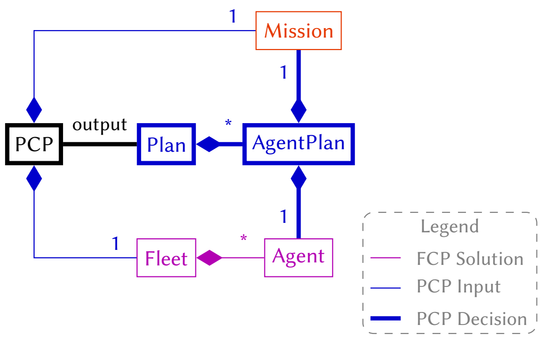


Figure 6: UML representation of the PCP.

- a mission m,
- a fleet $f_{s_{cat},\mathscr{T}_{cat}}$.

5.2.2. Decision Variables

In order to represent the position of each agent in the solution plan, we use binary decision variables (**Vpl** matrix) that indicate whether an agent is at a given position at each time step. Similarly, for each task, we use binary decision variables indicating whether an agent executes this task at the time step (**Tpl** matrix).

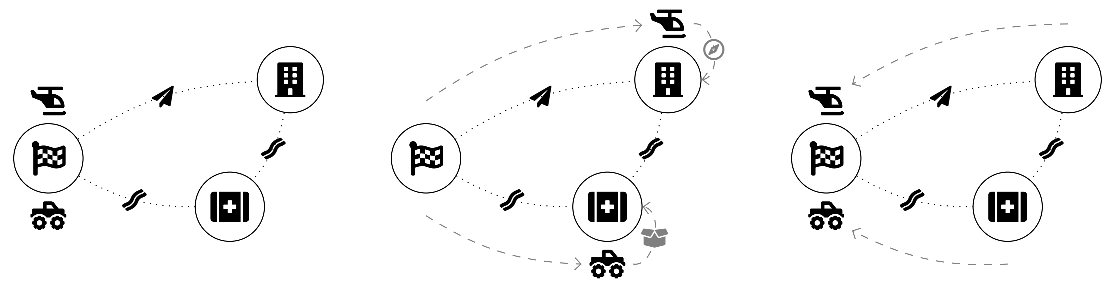
Formally, for a catalog cat, a stock on this catalog, s_{cat}, a mission m, a fleet $f_{s_{cat},\mathscr{T}_{cat}}$, a *plan* is a tuple $\mathbf{pl}_{m,f_{s_{cat},\mathscr{T}_{cat}}} = (\mathbf{H}, \mathbf{Tpl}, \mathbf{Vpl})$ where:

- $\mathbf{H} \in \mathbb{N}^*$ is the temporal plan horizon.
- $\mathbf{Tpl} = (\mathbf{tpl}_{i,j,h})_{i,j,h \in [1..n_a] \times [1..n_T] \times [1..\mathbf{H}]}$ is the allocation of tasks over agents for each time steps, represented as a tensor of size $n_a \cdot n_T \cdot \mathbf{H}$. For each agent $i \in [1..n_a]$, each task $j \in [1..n_T]$ and each time step $h \in [1..\mathbf{H}]$, $\mathbf{tpl}_{i,j,t} = 1$ if the agent $\mathbf{a}_i \in \mathscr{A}_\mathfrak{f}$ is executing the task j at the time h. It equals 0 otherwise.
- $\mathbf{Vpl} = (\mathbf{vpl}_{i,j,h})_{i,j,h \in [1..n_a] \times [1..n_V] \times [1..\mathbf{H}]}$ is the position of the agents for each time steps, defined by a tensor of size $n_a \cdot n_T \cdot \mathbf{H}$. For each agent $i \in [1..n_a]$, each task $j \in [1..n_T]$ and each time step $h \in [1..\mathbf{H}]$, $\mathbf{vpl}_{i,j,t}$ equals 1 if the agent $a_i \in \mathscr{A}_\mathfrak{f}$ is at the vertex j at the time h. It equals 0 otherwise.

Through this plan formalization, moves of agents are not explicitly described, but this piece of information could be retrieved through their positions.

5.3. Toy problem PCP

We consider the definitions of mission m, catalog cat, introduced in the three previous examples.



(a) Step 1 (b) Step 2 (c) Step 3

Figure 7: Execution of plan $\mathbf{pl}_{m,f_{s_{cat},\mathcal{T}_{cat}}}$ from Example 5.3: starting from the base, the UAV moves to the ruins while the rover moves to the aid camp; then, they both perform the required tasks in their respective locations; finally, they both come back to the base.

We consider the following plan for the fleet $f^2_{s_{cat},\mathcal{T}_{cat}} = (2,(a_1,a_2))$:

$\mathbf{pl}_{m,f_{s_{cat},\mathcal{T}_{cat}}} = (3,(\mathbf{Tpl}_1 \quad \mathbf{Tpl}_2)),(\mathbf{Vpl}_1 \quad \mathbf{Vpl}_2))$, illustrated in Fig. 7, where:

- $\mathbf{Tpl}_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ is the task allocation matrix of the first agent of the fleet, that has pattern a_1 (platform). It performs the task "explore the ruins" () at time step 2.
- $\mathbf{Vpl}_1 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ describes the movement of the first agent of the fleet, that has pattern a_1. It starts at the "base" () than goes to the "ruins" () and comes back to the "base" ().
- $\mathbf{Tpl}_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ is the task allocation matrix of the second agent of the team, with pattern a_2 (platform). It performs the task "deliver supplies" () at the time step 2.
- $\mathbf{Vpl}_2 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ describes the movement of the second agent of the team, with pattern a_2. It starts at the "base" () than goes to the "aid camp" () and comes back to the "base" ().

Note that the time steps used in that example plan give a macro view of the agents actions. It would be possible to have a much finer discretization of the time in order to handle temporal constraints such as task duration, or edge traversal duration.

6. Conclusion

In this paper, we model and formalize the multi-level configuration problem for a fleet of heterogeneous agent. This problem is decomposed into three problems, ACP, FCP and PCP and for each of them, we formally define their inputs and their decision variables and we illustrate them on a toy problem. We focus on Search and Rescue missions where tasks have to performed on some nodes of a given graph.

The work presented in this paper is a first step for solving the multi-level configuration problem. As mentioned in the paper, the next step is to formally define the set of constraints and the eventual criteria associated to ACP, FCP and PCP. To do so, it will be possible to study the literature associated with each problem, such as [1] for ACP, [3, 6] for FCP and [7, 8, 9] for PCP.

Then, we have presented the three configuration problems independently but in practice, they are interleaved. For instance the output of ACP is an input of FCP, and the output of FCP is an input for PCP. In the other direction, the evaluation of solutions produced by PCP and FCP can influence the choices made in ACP. If the evaluation of the overall multi-level configuration solution is not satisfactory, there might be several interactions between each level before converging (if any convergence is possible). In order to avoid these interactions, it would be possible to solve all the configuration problems simultaneously. Some works have started contributing towards that objective [10, 11, 12, 2]. Following those works, we aim at proposing a global solver/architecture for solving the multi-level configuration problem.

Finally, we have considered here a simple model of a Search and Rescue mission. It would be possible to make it more realistic in several ways. For instance, it would be possible to consider: more complex mission (e.g. with multiple bases), autonomy constraints on agents forcing them to recharge in some specific locations, more complex tasks (e.g. requiring multiple capabilities, or requiring synchronisation between multiple agents), a non-deterministic setting (e.g. uncertainty on tasks duration) and a dynamic environment (e.g. discover the edges trafficability, agent's loss).

Acknowledgments

The authors would like to thank the ONERA, ISAE-SUPAERO and ENAC Federation for its support of this work. This work is partly founded by the ONERA federative project on cooperative and interactive intelligent multi-robot systems (SICICOD).

References

[1] É. Vareilles, S. Roussel, G. Picard, PERFECT: performant and robust read-to-fly fleet configuration: from robot to mission plan, in: J. M. Horcas, J. A. Galindo, R. Comploi-Taupe, L. Fuentes (Eds.), Proceedings of the 25th International Workshop on Configuration (ConfWS 2023), Málaga, Spain, September 6-7, 2023, volume 3509 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 104–107.

[2] C. Lei, W.-H. Lin, L. Miao, A two-stage robust optimization approach for the mobile facility fleet sizing and routing problem under uncertainty, Computers & Operations Research 67 (2016) 75–89.

[3] G. Weiss, Multiagent systems, Second Edition, MIT press, 2013.

[4] G. Radzki, P. Golinska-Dawson, G. Bocewicz, Z. Banaszak, Modelling robust delivery scenarios for a fleet of unmanned aerial vehicles in disaster relief missions, Journal of Intelligent & Robotic Systems 103 (2021) 1–18.

[5] T. Calamoneri, F. Corò, S. Mancini, A realistic model to support rescue operations after an earthquake via uavs, IEEE Access 10 (2022) 6109–6125.

[6] J. F. Hübner, J. S. Sichman, O. Boissier, Moise+ towards a structural, functional, and deontic model for mas organization, in: Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1, 2002, pp. 501–502.

[7] J. Blythe, An overview of planning under uncertainty, Artificial Intelligence Today: Recent Trends and Developments (2001) 85–110.

[8] Ç. Koç, T. Bektaş, O. Jabali, G. Laporte, Thirty years of heterogeneous vehicle routing, European Journal of Operational Research 249 (2016) 1–21.

[9] L. Berghman, Y. Kergosien, J.-C. Billaut, A review on integrated scheduling and outbound vehicle routing problems, European Journal of Operational Research 311 (2023) 1–23.

[10] R. F. Lemme, E. F. Arruda, L. Bahiense, Optimization model to assess electric vehicles as an alternative for fleet composition in station-based car sharing systems, Transportation Research Part D: Transport and Environment 67 (2019) 173–196.

[11] R. Pinto, A. Lagorio, R. Golini, Urban freight fleet composition problem, IFAC-PapersOnLine 51 (2018) 582–587.

[12] H. R. Sayarshad, R. Tavakkoli-Moghaddam, Solving a multi periodic stochastic model of the rail–car fleet sizing by two-stage optimization formulation, Applied Mathematical Modelling 34 (2010) 1164–1174. doi:https://doi.org/10.1016/j.apm.2009.08.004.