Project Number: 774571
Start Date of Project: 2020/10/08
Duration: 48 months

Type of document 3.5 – V1.0

Farm Activities Planner

Dissemination level	PU
Submission Date	2021-05-31
Work Package	WP3
Task	T3.4
Type	Report
Version	1.0
Author	Matteo Santilli
Approved by	Andrea Gasparri, PMC

Executive Summary

This document proposes a framework to assign the farming operations required by the decision-making process of the project to the robotics platforms or at our disposal. In particular, the proposed framework is composed of two modules: the first one is designed for task allocation when no precedence constraints between pairs of farming operations are involved, whereas the second one builds upon the first module to provide a solution when a sequence of operations is required to be completed before another can start. Applicative examples of the proposed framework to different contexts conclude the document.

The results described in this deliverable have been submitted for publications to the conference 20th International Conference on Advanced Robotics (ICAR) (http://icar-2021.org/).

Table of Content

Abbreviations and Acronyms

UGV	Unmanned Ground Vehicle
UAV	Unmanned Aerial Vehicle
3D	Three Dimensional
RGB	Red Green Blue
RGBD	Red Green Blue Depth
LiDAR	Light Detection and Ranging
NP	Nondeterministic Polynomial (time)

1 Introduction

The given document comprises a detailed description of task *T3.4 – Planner of Farming Activities* addressing PANTHEON's Objective 1.4. This objective is accomplished through the use of two modules that solve the task allocation and task scheduling problems, respectively. Specifically, the task allocation module is built from the adaptation of the matroidal optimization problem proposed in [1] where the authors addressed the task allocation problem by applying the combinatorial theory of *matroids*. In our context, matroidal constraints will be used to model the abstract notion of agent-task constraints, allowing us to encode the fact that a robot may be able to treat suckers with herbicides but at the same time not be able to collect RGBD data due to energy/battery constraints. The task allocation module however does not consider the presence of precedence constraints between different pairs of agronomical operations. These constraints are instead handled by the task scheduling module through the use of a Petri Net as done in [2], [3]. In particular, each agronomical operation is associated to a place of the Petri Net and each precedence constraint models a transition of the Net. In this way, when an operation is completed the transition towards the next constrained operation is enabled and can hence start. As it will be shown later, the task schedule module focuses only on dealing with the precedence constraints while leaving the job of allocating the required activities to the first module. A representative application of our agronomical scenario is used to validate the effectiveness of the proposed framework.

To the best of our knowledge, the framework presented in this deliverable represents the first attempt ever in allocating and scheduling tasks subjected to independent matroidal constraints modeling different feasible pairs of activities for a multi-agent team. Indeed, exploiting and adapting the results of [1], [2], and [3] to the precision farming context, our framework is able to find a feasible task assignment and allocation for teams of agents which may be capable of doing at the same time only a set of certain activities. To the best of our knowledge, the only work that address a problem similar to ours is [4] in which the authors propose a polynomial-time algorithm that yields a $(2+\epsilon)$-approximation for the preemptive concurrent open shop scheduling problem that consider the presence of a matroidal constraint for the task that have to be allocated. However, their modeling focuses only on the task as entities hence not considering the many facets that arise when different sets of agronomical operations and devices equipped by the agents are paired together. Other works at the state of the art which consider matroidal constraints for scheduling problems of teams of agents in their problem formulation are [5] and [6]. Specifically, in [5] a control scheduling problem, i.e., the problem of as-signing agents to react upon a discrete time dynamical system, is solved by a greedy algorithm when the objective function to be minimized by the team of agents possesses a particular property known as α-supermodularity. The authors in [6] propose a greedy algorithm for the solution of a scheduling problem expressed by matroidal constraints which address the recharge of wireless sensor networks. However, all of these works do not fit with the needs that a team equipped with different devices for different activities may require and for this reason we propose a novel task allocation and scheduling framework.

The remainder of the deliverable is organized as follows:

- In Section 2 preliminary notions concerning matroid theory, submodular optimization, and Petri Nets are given.
- Section 3 introduces the task allocation problem and explains how the intersection of matroids can be used to model our agronomical scenario.
- In Section 4 the task scheduling problem is introduced along with the Petri Net framework that will be used to solve it.
- Section 5 details how the frameworks introduced in Section 3 and Section 4 can be integrated to obtain a feasible solution for the allocation and scheduling problem at hand.
- Finally, in Section 6 numerical validations are given to corroborate the proposed framework.

2 Preliminaries

In this section we introduce the notation that will be used in the rest of the document as well as a quick review of the concepts of matroids, submodularity, and Petri Net.

2.1 Notation

We denote vectors and matrices by boldface lowercase letters and uppercase letters, respectively, and denote the i-th component of vector $\mathbf{x}$ by x_i and the (i,j)-th entry of the matrix A by A_{ij}. The vectors with entries all equal to zero (one) are denoted by $\mathbf{0}_n$ ($\mathbf{1}_n$). The $n \times n$ identity matrix is indicated by I_n, whereas $0_{n\times m}$ is used to denote the $n \times m$ zero matrix.

Given a set $\mathcal{S}$ we denote by $|\mathcal{S}|$ its cardinality, i.e., the number of elements contained in $\mathcal{S}$. Furthermore, we define by $\mathbb{B}$ the domain of Boolean numbers, i.e., $\mathbb{B} = \{0,1\}$, $\mathbb{N}$ the set of natural numbers, i.e., $\mathbb{N} = \{1,2,3,\dots\}$, and $\mathbb{N}_0$ the set of natural numbers and the zero, i.e., $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

We denote the logical operations of "not", "or", and "and" as $\neg$, $\vee$, and $\wedge$, respectively. Furthermore, given two Boolean matrices $A \in \mathbb{B}^{n\times m}$ and $B \in \mathbb{B}^{m\times l}$ we define the logical matrix product $C = A \odot B$ with $C \in \mathbb{B}^{n\times l}$ where each element C_{ij} is

$$C_{ij} = \bigvee_{k=1}^{m} \left(A_{ik} \wedge B_{kj}\right), \quad i \in \{1,\dots,n\},\ j \in \{1,\dots,l\}.$$

For example, consider the Boolean matrix $A \in \mathbb{B}^{3\times 5}$ and the Boolean vector $\mathbf{x} \in \mathbb{B}^5$ defined as

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

The logical matrix product $C = A \odot \mathbf{x}$ is the Boolean vector $C \in \mathbb{B}^3$ built as

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \odot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} (0 \wedge 1) \vee (1 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 1) \vee (1 \wedge 0) \\ (1 \wedge 1) \vee (0 \wedge 0) \vee (1 \wedge 0) \vee (1 \wedge 1) \vee (0 \wedge 0) \\ (0 \wedge 1) \vee (1 \wedge 0) \vee (1 \wedge 0) \vee (0 \wedge 1) \vee (0 \wedge 0) \end{bmatrix} = \begin{bmatrix} 0 \vee 0 \vee 0 \vee 0 \vee 0 \\ 1 \vee 0 \vee 0 \vee 1 \vee 0 \\ 0 \vee 0 \vee 0 \vee 0 \vee 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}.$$

Finally, let us introduce a vector function $\varphi(\cdot)$ which maps a vector of integer values to a vector of Boolean values of the same dimension such that each entry is equal to 1 if the argument of $\varphi(\cdot)$ is greater than 0, and 0 otherwise.

For example, consider the vector $\mathbf{v} \in \mathbb{N}_0^6$ defined as $\mathbf{v} = [1,2,0,3,1,0]^T$, then $\varphi(\mathbf{v}) \in \mathbb{B}^6$ is the vector $\varphi(\mathbf{v}) = [1,1,0,1,1,0]^T$.

2.2 Matroid Theory

In this section we review the notion of matroids. Matroids are a mathematical tool that abstracts and generalizes the notion of linear independence for vectors. A reason of interest for matroids in the field of combinatorial optimization is their association with submodular set functions and greedy algorithms who can guarantee optimality bounds. There are many ways to define a matroid; we adopt the more common one based on the concept of independence. For a more in-depth review of the argument the reader is referred to [4].

Definition 1 (Matroid). A matroid $\mathcal{M}$ on E is an ordered pair $(E, \mathcal{I})$ consisting of a finite set E, denoted as the *ground set* of $\mathcal{M}$, and a collection $\mathcal{I}$ of subsets of E, denoted as the *independent sets* of $\mathcal{M}$, satisfying the following three conditions:

I. $\emptyset \in \mathcal{I}$;
II. Every subset of an independent set is independent, i.e., if $X \in \mathcal{I}$ and $Y \subseteq X$, then $Y \in \mathcal{I}$;
III. If X and Y are independent sets, i.e., $X, Y \in \mathcal{I}$, and $|X| < |Y|$, then there is an element $e \in Y \backslash X$ such that $X \cup \{e\} \in \mathcal{I}$.

Condition c) is referred to as *independence augmentation axiom*, and essentially states that if X is an independent set and there exists a larger independent set Y, then X can be extended to a larger independent set by adding an element of $Y \backslash X$. Condition c) also implies that every maximal independent set is maximum from which we can infer that all maximal independent sets have the same cardinality.

A subset of E that is not in $\mathcal{I}$ is called *dependent*. A maximal independent set is called *base* of the matroid, while a minimal dependent set is called *circuit* of the matroid.

Different independence rules define different matroids. Some of the most famous and used in literature are the following:

- **Uniform matroid**. The independence rule is defined as

$$\mathcal{I} = \{X \subseteq E : |X| \leq k\},$$

  for a given $k \in \mathbb{N}$ with $k \leq |E|$. In this type of matroid a base is any set that has cardinality equal to k.
- **Partition matroid**. In this matroid the ground set E is partitioned in ℓ disjoint subsets $E_1, \dots, E_\ell$ and the independence rule is

$$\mathcal{I} = \{X \subseteq E : |X \cap E_i| \leq k_i,\ i = 1, \dots, \ell\},$$

  for given $k_i \in \mathbb{N}$.
- **Linear matroid**. Consider a matrix A and let the ground set E be the set of column indices of the matrix A. For a subset $X \subseteq E$, let A_X denote the submatrix of A composed only of those columns indexed by the elements in X. The independence rule is

$$\mathcal{I} = \{X \subseteq E : \text{rank}(A_X) = |X|\},$$

meaning that a indices set is independent only if the indexed columns are linearly independent. It follows that a base of a linear matroid correspond to set of columns of cardinality rank(A). Notably, the name *matroid* originates from this type of matroid.

In order to explain the basic concepts of a matroid to the reader, let us consider the following linear matroid.

Example 1. Consider the matrix $A \in \mathbb{R}^{2\times 5}$ defined as

$$A = \begin{bmatrix} 2 & 0 & 0 & 3 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Consider the associated matroid $\mathcal{M}$ with ground set $E = \{1,2,3,4,5\}$ denoting the indices of the $\mathbb{R}^2$ vectors composing matrix A. The independence set $\mathcal{I}$ collecting the indices of independent vectors in A is then

$$\mathcal{I} = \{\emptyset, \{1\}, \{2\}, \{4\}, \{5\}, \{1,2\}, \{1,5\}, \{2,4\}, \{2,5\}, \{4,5\}\}.$$

The set of dependent indices $\mathcal{D}$ is instead

$$\mathcal{D} = \{\{3\}, \{1,3\}, \{1,4\}, \{2,3\}, \{3,4\}, \{3,5\}\} \cup X_3.$$

where X_3 is the set collecting all substes of three or more elements of the ground set E, i.e., $X_3 = \{X \subseteq E: |X| \geq 3\}$. The set collecting all bases of the matroid $\mathcal{M}$, i.e., the maximal indepedenent sets of $\mathcal{M}$, is $\mathcal{B} = \{\{1,2\}, \{1,4\}, \{2,4\}, \{2,5\}, \{4,5\}\}$. Finally, the set collecting the circuits of $\mathcal{M}$, i.e., sets whose proper subsets are independent, is $\mathcal{C} = \{\{3\}, \{1,4\}, \{1,2,5\}, \{2,4,5\}\}$.

In this work, we will utilize the power of expressiveness encoded by matroids to model abstract independence constraints in the multi-agent task allocation problem whereby multi-agent we refer to the presence of multiple robotics platforms as Unmanned Ground Robots (UGVs) or Unmanned Aerial Robots (UAVs) as well as human operators on the field.

2.3 Matroid Optimization and Submodularity

In our multi-agent task allocation problem, we are interested in finding feasible solution satisfying matroid independence constraints, i.e., problems of the following form.

Problem 1. Consider a matroid $\mathcal{M} = (E, \mathcal{I})$, and a set function $f: 2^E \to \mathbb{R}$, then define:

$$\begin{aligned} &\max_{\mathcal{S}\subseteq E} f(\mathcal{S}) \\ &\text{s.t. } \mathcal{S} \in \mathcal{I} \end{aligned} \qquad (1)$$

Notably problems in the form of eq. (1) are generally NP-Hard [8]. However, when function $f(\cdot)$ satisfies certain requirements, guarantees of optimality performances can be given for solutions obtained by greedy algorithms. Specifically, when $f(\cdot)$ is *submodular*, Problem Problem **1** is still NP-Hard but greedy algorithms are able to obtain good approximation ratios which in certain cases are also the best-known results in literature [5].

We now quickly review the notion of submodularity; for a better understanding of the topic, the reader is referred to [5],[6].

Definition 2 (Submodularity). A set function $f: 2^E \rightarrow \mathbb{R}$ is submodular if for every $A \subseteq B \subseteq E$ and $e \in E \backslash B$ it holds that

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B). \quad (2)$$

Based on eq. (2), submodularity implies that after performing a set A of actions which provide a benefit $f(A)$, the marginal benefit of any additional action e does not increase if we perform any of the actions in $B \backslash A$. Hence, submodular set functions exhibit a natural diminishing returns property. When eq. (2) is satisfied to the equality, then function $f(\cdot)$ is said *modular*.

Another important class of set functions are *monotone* functions, which are defined as follows.

Definition 3 (Set Function Monotonicity). A set function $f: 2^E \rightarrow \mathbb{R}$ is monotone if for every $A \subseteq B \subseteq E$ it holds that $f(A) \leq f(B)$.

When Problem Problem **1** considers a monotone submodular set function $f(\cdot)$ then it is proven that a greedy algorithm can achieve a $\frac{1}{2}$- approximation ratio, i.e., the value of the solution $\mathcal{S}$ found by the greedy algorithm satisfies $f(\mathcal{S}) \geq \frac{1}{2} f(\mathcal{S}^\star)$ where $f(\mathcal{S}^\star)$ is the value of the optimal solution [5].

A greedy algorithm is a fast and easy-to-implement problem-solving heuristic which consists in choosing the best local solution at each step of the algorithm. Briefly, a greedy algorithm in charge of solving a problem in the form of eq. (1), starts from an empty set and adds at each step an element which i) maximizes the marginal benefit and ii) maintains the solution set independent. To build an independent solution, greedy algorithms require to know if a set is independent or not, i.e., require the presence of the so-called *independence oracle*, whose formal definition is given hereinafter.

Definition 4 (Independence Oracle). Given a matroid $\mathcal{M} = (E, \mathcal{I})$, an independence oracle $\Psi(X)$ for all $X \subseteq \mathcal{I}$ is a function defined as

$$\Psi(X) = \begin{cases} \text{True} & \text{if } X \in \mathcal{I}, \\ \text{False} & \text{if } X \notin \mathcal{I}. \end{cases}$$

Given an independence oracle $\Psi(\cdot)$, the structure of a greedy algorithm able to solve Problem Problem **1** is summarized in Algorithm 1.

Algorithm 1 *Greedy Algorithm for Problem 1*

Require: Independence oracle Ψ

$\mathcal{S} = \{\emptyset\}$
while $\Psi(\mathcal{S})$ **do**
  $e^\star = \underset{e \notin \mathcal{S}:\, \Psi(\mathcal{S} \cup \{e\}) == \text{True}}{\arg\max} \left(f(\mathcal{S} \cup \{e\}) - f(\mathcal{S}) \right)$
  $\mathcal{S} = \mathcal{S} \cup \{e^\star\}$
end while
return $\mathcal{S}$

Algorithm 1 - Greedy algorithm's pseudocode for Problem 1.

2.4 Matroid Intersection

As introduced earlier, in this work we use a matroid to encode abstract independence constraints in the multi-agent allocation problem. In a general problem formulation, however, there may be more constraints that need be considered to represent the system at hand, e.g., the fact that a task may be assigned to a single robot only or the fact that every task must be assigned. In order to encode different kinds of constraints in the matroid domain, the *matroid intersection* is used.

Theorem 1 (Matroid Intersection [5]). *Consider the matroid intersection system given by the intersection of p matroids $\mathcal{M}_1, \dots, \mathcal{M}_p$ as*

$$\bigcap_{i=1}^{p} \mathcal{M}_i = \left(E, \bigcap_{i=1}^{p} \mathcal{I}_i \right), \tag{3}$$

where $\mathcal{M}_k = (E, \mathcal{I}_i)$ are matroids and E is a common ground set. The independence system in eq. (3) models any arbitrary (and possibly non-matroidal) independence system.

We have now the necessary tools to generalize the scope and utility of the optimization problem introduced in Problem Problem **1**.

Problem 2. Consider a monotone (sub)modular set function $f: 2^E \to \mathbb{R}$ and a set of p matroids $\mathbb{M} = \{\mathcal{M}_1, \dots, \mathcal{M}_p\}$ on a common ground set E, then define:

$$\begin{aligned} &\max_{\mathcal{S} \subseteq E} f(\mathcal{S}) \\ &\text{s.t.} \quad \mathcal{S} \in \bigcap_{\mathcal{M} \in \mathbb{M}} \mathcal{M} \end{aligned} \tag{4}$$

Naturally, greedy algorithms are able to solve also optimization problems in the form of eq. (4). Specifically, let $\mathcal{S}^\star$ be the optimal solution and $\mathcal{S}^G$ be the solution found by the greedy algorithm. Then the following results on the optimality of the solution are known.

Theorem 2 (Greedy Approximation [5]). *Consider Problem 2 with a monotone submodular objective function. Then the solution $\mathcal{S}^G$ found by the greedy algorithm satisfies*

$$f(\mathcal{S}^G) \geq \frac{1}{p+1} f(\mathcal{S}^\star),$$

where p is the number of matroids involved in the intersection, i.e., $p = |\mathbb{M}|$. In addition, when the objective function is monotone modular, the bound is improved to

$$f(\mathcal{S}^G) \geq \frac{1}{p} f(\mathcal{S}^\star).$$

The results of Theorem 2 shows that the quality of the solution of the greedy algorithm depends on the number of matroidal constraints involved in the intersection matroid. Specifically, the greater the number of matroidal constraints, the worse the optimality guarantee gets.

2.5 Petri Net

Petri Nets are a class of a mathematical modeling language for discrete event systems. A Petri Net consists of three basic elements: *places*, *transitions*, and *arcs*. Arcs link places to transitions and transitions to places. Places from which an arc starts and arrives to a transition are called *input places* whereas if an arc starts from a transition and arrives to a place, that place is called *output place*. Each place is associated to a number of marks called *tokens*, which can vary between none and a positive integer number. A transition can be *fired* if it is enabled, i.e., if all of its input places contain a sufficient number of tokens. When a transition is fired, the tokens are removed from its input places and added to its output places.

The formal definition of a Petri Net and a Marked Petri Net are given hereinafter.

Definition 5 (Petri Net). A Petri Net is a triplet (P, T, W) in which S is a finite set of places, T a finite set of transitions and $W \subseteq (P \times T) \cup (T \times P)$ is a multiset collecting the arcs of the Petri Net. The places and transitions sets are disjoint, i.e., $S \cap T = \emptyset$ and the arcs do not link two places or two transitions.

Definition 6 (Marked Petri Net). A marked Petri Net is a tuple (P, T, W, M_0) where (P, T, W) is a Petri Net and M_0 is an initial marking of the Petri Net, i.e., the set of initial tokens of the net.

3 Task Allocation

In this section, we introduce the multi-agent allocation problem that we intend to use to model the allocation of farming operations. The problem is adapted to the needs of the project from the work [1].

3.1 Problem Definition

Let us introduce the basic elements from which the problem formulation expressed by a matroid intersection will be derived.

Definition 7 (Agents). Let $A = \{a_1, \dots, a_n\}$ denote the set of agents of size n which can be assigned to perform the farming operations. For instance, agents can be UGVs, UAVs, or agronomical experts.

Definition 8 (Devices). Let $D = \{d_1, \dots, d_m\}$ denote the set of devices of size m which are equipped by the agents and are required to perform the farming operations. For instance, possible devices can be an RGBD camera sensor, a LIDAR scanning sensor, or an herbicide sprayer.

Definition 9 (Operations). Let $O = \{o_1, \dots, o_q\}$ denote the set of farming operations of size q which needs to be executed in the field such as suckers' management or water stress level assessment.

The multi-agent allocation problem is solved if a feasible allocation for every operation O is found. In our context, each operation requires the usage of a single device to be accomplished. For instance, the 3D tree reconstruction operation can be achieved if the LIDAR scanner sensor is used. As an example, consider the scenario depicted in Figure 1 where there are a set of 5 devices (sensors) and a set of 3 farming operations that need to be accomplished. For example, the first farming operation o_1, i.e., the 3D tree reconstruction, can be achieved by using the LIDAR scan sensor d_1, the RGB camera sensor d_2, or the multi-spectral camera sensor d_3. To encode these three possibilities, we represent the feasible pairs as couples (d_1, o_1), (d_2, o_1), and (d_3, o_1). In the rest of the document, we are going to refer to pairs of operations and devices also as *activities*, i.e., an activity is a couple (d_j, o_k) with $d_j \in D$ and $o_k \in O$.

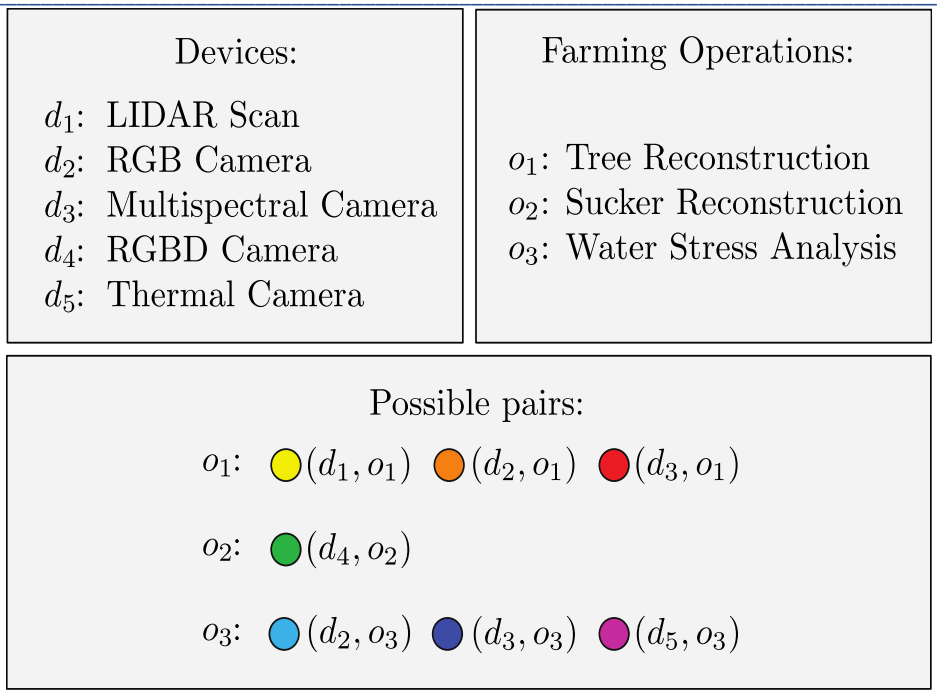

Figure 1 - Scenario of possible associations between devices and operations. A couple (d_j, o_k) denotes the feasibility of the operation o_k with device d_j.

Our multi-agent task allocation problem consists in assigning activities to agents trying to maximize a return value from the selected allocation. For this reason, we consider an allocation as a *triplet* (a_i, d_j, o_k) with $a_i \in A$, $d_j \in D$, and $o_k \in O$ which can be read as "agent a_i performs operation o_k using device d_j".

Having described the basic modeling, we can now proceed introducing the concept of abstract independence constraints that we want to model with matroids. Consider again the scenario depicted in Figure 1 and assume there is a ground robot a_1 who is equipped with sensors d_1, d_2, and d_3. Suppose that the robot needs to carry out operations o_1 and o_3. From the possible pairs data, we know that operation o_1 can be carried out using any of the three sensors that the robot is equipped with, whereas operation o_3 can only be carried out using sensors d_2 or d_3. We want now to model the fact that the robot may not be capable to perform certain activities together, e.g., the battery of the robot may add a constraint that prevents the combined use of the LIDAR scan sensor (d_1) for the tree reconstruction (o_1) with the use of the RGB camera (d_2) for the water stress analysis (o_3). Hence, in the final allocation, the triplets (a_1, d_1, o_1) and (a_1, d_2, o_3) could not coexist. From a matroidal point of view this constraint is translated to the dependence of the set collecting the two triplets, i.e., $\{(a_1, d_1, o_1), (a_1, d_2, o_3)\} \notin \mathcal{I}$, meaning that in a solution the two assignments would not be present.

To expand upon the scenario introduced in Figure 1, consider the possible independent allocation depicted in Figure 2. There are three robots a_1, a_2, and a_3 which are associated with different sets of activities, represented by circles. A single circle represents the fact that the activity associated to that colored circle must be carried out by the robot individually, i.e., no other activity can be assigned to the robot; a couple of circles instead represent the fact those two activities associated with the colored circles can be performed together by the robot; the same goes for a triplet of circles and so on. For example robot a_1 can individually perform operation o_1 or operation o_2 with one of any of the sensors it is equipped with; robot a_1 could also perform operation o_1 using at the same time sensor d_1 and sensor d_2 whereas it is not be able to perform any of the activities for operation o_1 while at the same time performing some other activities for operation o_2 since there are no couples of circles that involve at the same time any of the colors among yellow, orange, and red associated to operation o_1 with any of the colors azure, blue, and violet associated to operation o_2.

Finally, given the chance, robot a_1 could also execute operation o_1 using all three sensors at once as represent by the triplet of yellow, orange, and red circles.

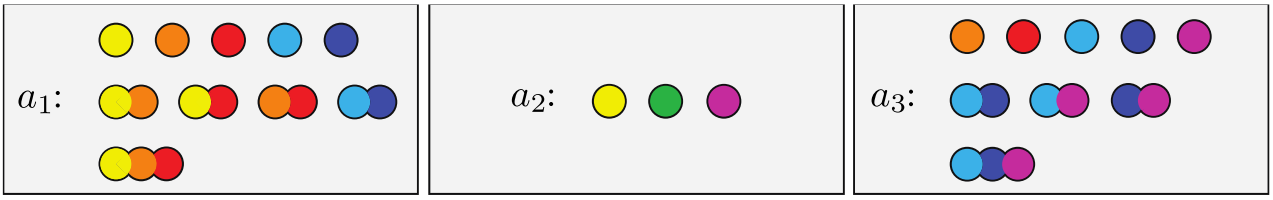

Figure 2 - Scenario of independent allocations to a set of three agents. Each colored circle corresponds to a couple (d_j, o_k) with operation o_k and device d_j as described in Figure 1. An individual circle means that the agent can perform the combination of operation and device only. Two paired circles mean that the agent can be assigned to those two pairs of operation/device at the same time and so on.

In a more formal way, the feasible allocations for each agent a_i can be described by an independence set $\mathcal{I}_{a_i}$ which will contain all the activities that can be assigned to agent a_i. Consider the scenario illustrated in Figure 1 and expanded in Figure 2, the possible allocations for robot a_1 are described by the independence set $\mathcal{I}_{a_1}$ defined as:

$$\begin{aligned}\mathcal{I}_{a_1} = \{\quad & \{\emptyset\}, \{(a_1, d_1, o_1)\}, \{(a_1, d_2, o_1)\}, \{(a_1, d_3, o_1)\}, \\ & \{(a_1, d_2, o_3)\}, \{(a_1, d_3, o_3)\}, \\ & \{(a_1, d_1, o_1), (a_1, d_2, o_1)\}, \{(a_1, d_1, o_1), (a_1, d_3, o_1)\}, \\ & \{(a_1, d_2, o_1), (a_1, d_3, o_1)\}, \{(a_1, d_2, o_3), (a_1, d_3, o_3)\}, \\ & \{(a_1, d_1, o_1), (a_1, d_2, o_1), (a_1, d_3, o_1)\} \quad \}.\end{aligned}$$

Similarly, assuming that agent a_2 is equipped with at least devices d_1, d_4, and d_5, the independent allocations for agent a_2 are encoded by the independence set $\mathcal{I}_{a_2}$:

$$\mathcal{I}_{a_2} = \{\quad \{\emptyset\}, \{(a_2, d_1, o_1)\}, \{(a_2, d_4, o_2)\}, \{(a_2, d_5, o_3)\} \quad \}.$$

Finally, concerning the possible allocations of agent a_3, its independence set $\mathcal{I}_{a_3}$ is:

$$\begin{aligned}\mathcal{I}_{a_3} = \{\quad & \{\emptyset\}, \{(a_3, d_2, o_1)\}, \{(a_3, d_3, o_1)\}, \{(a_3, d_2, o_3)\}, \\ & \{(a_3, d_3, o_3)\}, \{(a_3, d_5, o_3)\}, \{(a_3, d_2, o_3), (a_3, d_3, o_3)\}, \\ & \{(a_3, d_2, o_3), (a_3, d_5, o_3)\}, \{(a_3, d_3, o_3), (a_3, d_5, o_3)\}, \\ & \{(a_3, d_2, o_3), (a_3, d_3, o_3), (a_3, d_5, o_3)\} \quad \},\end{aligned}$$

where it is assumed that agent a_3 is equipped at least with devices d_2, d_3, and d_5.

3.2 Problem Formulation

We have now the necessary tools to state the multi-agent task allocation problem as a matroid intersection optimization problem.

Problem 3 (Multi-Agent Task Allocation). Consider a set of agents, devices and operations described according to Definition 7, Definition 8, and Definition 9. Define a common ground set E as

$$E = \{(a_i, d_j, o_k) : a_i \in A, d_j \in D, o_k \in O\},$$

and a utility function $u(\cdot): 2^E \rightarrow \mathbb{R}$ for all elements $(a_i, d_j, o_k) \in E$. Furthermore, consider that each agent $a_i \in A$ is associated to an independent set $\mathcal{I}_{a_i}$ collecting the feasible pairs of devices and operations that can performed by agent a_i. Then, the multi-agent task allocation problem is encoded by the following formulation:

$$\begin{aligned} \max_{\mathcal{S} \subseteq E} \quad & \sum_{(a_i, d_j, o_k) \in \mathcal{S}} u\left(a_i, d_j, o_k\right) \\ \text{s.t.} \quad & \mathcal{S} \in \mathcal{M}_1 \cap \mathcal{M}_2 \end{aligned} \tag{5}$$

where $\mathcal{M}_1$ and $\mathcal{M}_2$ are matroids encoding the two constraints described hereinafter:

I. Each pair of device and operation (activity) (d_j, o_k) cannot be allocated more than once. This is encoded by the matroid $\mathcal{M}_1 = (E, \mathcal{I}_1)$;
II. The allocations of each agent a_i must lie in the independent set $\mathcal{I}_{a_i}$. This is encoded by the matroid $\mathcal{M}_2 = (E, \mathcal{I}_2)$.

Finally, we assume that the given sets A, D, O, and $\mathcal{I}_{a_i}$ with $a_i \in A$ admit a feasible solution to the problem.

Let us now properly introduce the independence sets associated with the matroids $\mathcal{M}_1$ and $\mathcal{M}_2$ defying the desired constraints described in points I and II.

Constraint I is encoded by the following independence set:

$$\mathcal{I}_1 = \left\{\mathcal{S} \subseteq E: \left|\mathcal{S} \cap G_{jk}\right| \leq 1, \forall \left(d_j, o_k\right) \text{ with } d_j \in D, o_k \in O\right\},$$

where G_{jk} are sets of triplets collecting the possible agents' allocation for a given pair of device d_j and operation o_k, i.e., $G_{jk} = \left\{\left(a_i, d_j, o_k\right): a_i \in A\right\}$.

Constraint II is encoded by the following independence set:

$$\mathcal{I}_2 = \left\{\mathcal{S} \subseteq E: \left|\mathcal{S} \cap \mathcal{I}_{a_i}\right| = 1, \forall\, a_i \in A\right\},$$

where $\mathcal{I}_{a_i}$ are the sets of independent and feasible allocations for agent a_i.

Theorem 3. *The independence systems $\mathcal{M}_1$ and $\mathcal{M}_2$, as defined in eq. (5) for Problem 3, are matroidal.*

Proof. In order to prove this result, we verify the three axioms introduced in Definition 1 for each matroid.

Matroid $\mathcal{M}_1$:

Axiom I):
Consider the subset $\mathcal{S} = \{\emptyset\}$. The independence rule $\left|\mathcal{S} \cap G_{jk}\right| \leq 1$ is then satisfied for any G_{jk} with $d_j \in D$ and $o \in O$ since $\left|\mathcal{S} \cap G_{jk}\right| = 0$. Axiom I) is hence verified.

Axiom II):
Consider a set $\mathcal{S}_1 \subseteq E$ satisfying independence rule $\mathcal{I}_1$, i.e., $\left|\mathcal{S}_1 \cap G_{jk}\right| \leq 1$ ($\mathcal{S}_1 \in \mathcal{I}_1$). This implies that there exists a unique element $\left(a_i, d_j, o_k\right) \in \mathcal{S}_1$ for each pair $\left(d_j, o_k\right)$ with $d_j \in D$ and $o_k \in O$. For some subset $\mathcal{S}_2 \subseteq$

$\mathcal{S}_1$ and for each pair (d_j, o_k), then there will always exists a unique element $(a_i, d_j, o_k) \in \mathcal{S}_2$ satisfying $|\mathcal{S}_2 \cap G_{jk}| \leq 1$. Axiom II) is hence proven.

Axiom III):
Consider two sets $\mathcal{S}_1, \mathcal{S}_2 \in \mathcal{I}_1$ and assume that $|\mathcal{S}_1| > |\mathcal{S}_2| > 0$. We will now prove the axiom by contradiction. Assume that no element $e \in \mathcal{S}_1 \backslash \mathcal{S}_2$ can be added to $\mathcal{S}_2$ while maintaining the independence rule valid, i.e., $\nexists\, e \in \mathcal{S}_1 \backslash \mathcal{S}_2 : \{e\} \cup \mathcal{S}_2 \in \mathcal{I}_1$. This implies that all pairs (d_j, o_k) have been allocated in $\mathcal{S}_2$, i.e., $|\mathcal{S}_2| = |\mathcal{A}|$ where $\mathcal{A}$ is the set collecting all pairs (d_j, o_k). However, since by definition $|\mathcal{S}_1| \leq |\mathcal{A}|$ and by hypothesis $|\mathcal{S}_1| > |\mathcal{S}_2|$, the contradiction arises thus proving axiom III).

Matroid $\mathcal{M}_2$:

Axiom I):
Consider the subset $\mathcal{S} = \{\emptyset\}$. The independence rule $|\mathcal{S} \cap \mathcal{I}_{a_i}| = 1$ for any $\mathcal{I}_{a_i}$ with $a_i \in A$ holds by construction since the independence sets $\mathcal{I}_{a_i}$ contain the empty set $\emptyset$. Axiom I) then holds true.

Axiom II):
Consider a set $\mathcal{S}_1 \subseteq E$ satisfying independence rule $\mathcal{I}_2$, i.e., $|\mathcal{S}_1 \cap \mathcal{I}_{a_i}| = 1$ ($\mathcal{S}_1 \in \mathcal{I}_2$). This implies that there exists a unique element $e_i^1 \in \mathcal{S}_1$ for each agent $a_i \in A$. For some subset $\mathcal{S}_2 \subseteq \mathcal{S}_1$ and each agent a_i then there will always exists a unique element $e_i^2 \in \mathcal{S}_2$ satisfying $|\mathcal{S}_2 \cap \mathcal{I}_{a_i}| = 1$. Axiom II) is hence proven.

Axiom III):
Consider two sets $\mathcal{S}_1, \mathcal{S}_2 \in \mathcal{I}_1$ and assume that $|\mathcal{S}_1| > |\mathcal{S}_2|$, i.e., $\mathcal{S}_1 \backslash \mathcal{S}_2 \neq \emptyset$. We will now prove the axiom by contradiction. Assume that no element $e \in \mathcal{S}_1 \backslash \mathcal{S}_2$ can be added to $\mathcal{S}_2$ while maintaining the independence rule valid, i.e., $\nexists\, e \in \mathcal{S}_1 \backslash \mathcal{S}_2 : \{e\} \cup \mathcal{S}_2 \in \mathcal{I}_1$. This implies that the set $\mathcal{S}_2$ is a base for the matroid $\mathcal{M}_2$. In turn, this implies that the cardinality $|\mathcal{S}_2|$ is maximal. However, since by hypothesis $|\mathcal{S}_1| > |\mathcal{S}_2|$, the contradiction arises thus proving axiom III).

Having detailed the structure of the matroid intersection, we can now characterize the performance properties of the greedy algorithm introduced in Algorithm 1 when used for Problem 3.

Corollary 1. *The greedy algorithm introduced in Algorithm 1 solves Problem 3 with the following optimality bound:*

$$f(\mathcal{S}) \geq \frac{1}{2} f(\mathcal{S}^\star).$$

Proof. The result follows from the application of Theorem 2 in the case of a monotone modular function $f(\cdot)$ with $p = 2$.

4 Task Scheduling

4.1 Model Definition

So far, we have described allocation of tasks with no consideration for possible precedence constraints between activities or different processing times. In this section we will introduce a discrete event system based on a Petri Net [2], [3] that is capable of handling the presence of precedence or time constraints in the multi-agent task allocation problem. We consider each operation to be non-preemptive, i.e., the execution of an operation cannot be interrupted. Let us now introduce the following definitions that will model the discrete event system.

Definition 10 (Input). Let U be the set of input events which may trigger the start of one or more operations $o \in O$, e.g., a sensor input or a user's command. Furthermore, let $\mathbf{u}(\tau) \in \mathbb{B}^{|U|}$ be the vector collecting the active input events at time τ, i.e., the i-th element of $\mathbf{u}(\tau)$ is equal to 1 if the i-th input event has been enabled, and 0 otherwise.

Definition 11 (Output). Let Y be the set of output events which represent the completion of one or more operations $o \in O$. Furthermore, let $\mathbf{y}(\tau) \in \mathbb{B}^{|Y|}$ be the vector collecting the active output events at time τ, i.e., the i-th element of $\mathbf{y}(\tau)$ is equal to 1 if the i-th output event has been enabled, and 0 otherwise.

Definition 12 (Rule). Let X be the set of *rules* which are logical elements linking pairs of operations, representing possible transitions in a Petri Net. Rules hence represent precedence constraints among the operations. Furthermore, let $\mathbf{x}(\tau) \in \mathbb{B}^{|X|}$ be the vector collecting the activation of rules at time τ, i.e., the i-th element of $\mathbf{x}(\tau)$ is equal to 1 if the i-th rule has been fired, and 0 otherwise.

Definition 10, Definition 11, and Definition 12 express information that change over the time horizon of the allocation problem. In our settings, static information such as the relation between input and rules or rules and output are present as well. We collect this information in the following definitions.

Definition 13 (Input Matrix). Let $F^u \in \mathbb{B}^{|X|\times|U|}$ be the Boolean input matrix representing the relationship between inputs and rules. Specifically, an entry F^u_{ij} is equal to 1 if the j-th input event is required for the activation of the i-th rule.

Definition 14 (End-Operation Matrix). Let $F^o \in \mathbb{B}^{|X|\times|O|}$ be the Boolean end-operation matrix representing the relationship between completions of operations and rules. Specifically, an entry F^o_{ij} is equal to 1 if the i-th rule requires the completion of the j-th operation to be enabled.

Definition 15 (Start-Operation Matrix). Let $H^o \in \mathbb{B}^{|O|\times|X|}$ be the Boolean start-operation matrix representing the relationship between rules and completions of operations. Specifically, an entry H^o_{ij} is equal to 1 if the i-th operation starts as consequence of the activation of the j-th rule.

Definition 16 (Output Matrix). Let $H^y \in \mathbb{B}^{|Y|\times|X|}$ be the Boolean output matrix representing the relationship between rules and outputs. Specifically, an entry H^y_{ij} is equal to 1 if the i-th output event is the consequence of the activation of the j-th rule.

Definition 13 and Definition 14 describe the preconditions to certain events and are hence known as *preconditions* matrices. On the contrary, Definition 15 and Definition 16 describe the postconditions to

certain events and are hence known as *postconditions* matrices. The information on preconditions and postconditions can be encoded by two compacts matrices defined hereinafter.

The preconditions matrix $F \in \mathbb{B}^{|X|\times(|U|+|O|+|X|+|Y|)}$ is defined as

$$F = [F^u \quad F^o \quad I_{|X|} \quad 0_{|X|\times|Y|}],$$

and the postconditions matrix $H \in \mathbb{B}^{(|U|+|O|+|X|+|Y|)\times|X|}$ is defined as

$$H = \left[0_{|X|\times|U|} \quad H^{oT} \quad 0_{|X|\times|X|} \quad H^{yT}\right]^T.$$

These two matrices, as we will see later, will be used to model the evolution of the discrete event system.

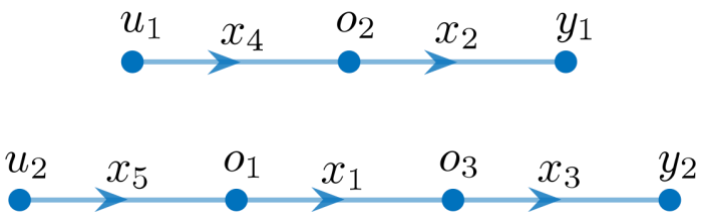

Figure 3 – Scenario of possible precedence constraints between a set of three operations $O = \{o_1, o_2, o_3\}$. Set of inputs, outputs, and rules are also shown.

In order to explain the structure of the preconditions matrix F and the postconditions matrix H consider the scenario depicted in Figure 3 where a set of three operations $O = \{o_1, o_2, o_3\}$ with precedence constraints is given. Specifically, operation o_1 can start only after input u_2 has been enabled, whereas operation o_2 can be started only after the activation of input u_1; operation o_3 instead has to wait for the completion of operation o_2 to be able to start. The output events y_1 and y_2 can be activated only after the completion of operations o_2 and o_3, respectively. The activation or the start of an input/output event and operation is enabled through the set of rules $X = \{x_1, \dots, x_5\}$ which are represented by the arrows. In this case matrix F is composed by $F^u \in \mathbb{B}^{5\times2}$ and $F^o \in \mathbb{B}^{5\times3}$ defined as

$$F^u = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array} \begin{array}{c} \begin{array}{cc} u_1 & u_2 \end{array} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \end{array}, \qquad F^o = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array} \begin{array}{c} \begin{array}{ccc} o_1 & o_2 & o_3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{array}.$$

Hence the overall structure of the preconditions matrix F is

$$F = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

Similarly, the postconditions matrix H is composed of $H^o \in \mathbb{B}^{3\times5}$ and $H^y \in \mathbb{B}^{2\times5}$ defined as

$$H^o = \begin{array}{c} \\ o_1 \\ o_2 \\ o_3 \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}, \quad H^y = \begin{array}{c} \\ y_1 \\ y_2 \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}.$$

The structure of the postconditions matrix H is then

$$\mathrm{H} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

The input, output, and rule vectors in this case are $\mathbf{u} \in \mathbb{B}^2$, $\mathbf{y} \in \mathbb{B}^2$, and $\mathbf{x} \in \mathbb{B}^5$.

4.2 Marking and Discrete Event System Evolution

As described earlier, the preconditions and postconditions matrices describe how the discrete event system evolves, i.e., they are in charge of moving the tokens of the Petri Net in the different states of the system. Before detailing this process, let us introduce the *marking vector* $\mathbf{m}$ which will collect the tokens of the system.

Definition 17 (Marking Vector). Let $\mathbf{m}(\tau) \in \mathbb{N}_0^{|U|+|O|+|X|+|Y|}$ be the marking vector collecting the tokens of the system at time τ defined as

$$\mathbf{m}(\tau) = [\mathbf{u}(\tau)^T \quad \mathbf{t}(\tau)^T \quad \mathbf{c}(\tau)^T \quad \mathbf{y}(\tau)^T]^T,$$

where $\mathbf{t}(\tau) \in \mathbb{N}_0^{|O|}$ and $\mathbf{c}(\tau) \in \mathbb{B}^{|X|}$ are task and control vector at time τ, respectively. The task vector $\mathbf{t}(\tau)$ collects the tokens associated to the operations, i.e., the i-th entry of $\mathbf{t}(\tau)$ is equal to h if the i-th operation has h tokens, whereas the control vector $\mathbf{c}(\tau)$ is in charge of allowing the firing of the rules $\mathbf{x}(\tau)$, i.e., the i-th entry of $\mathbf{c}(\tau)$ is equal to 1i if the i-th rule can be fired, 0 otherwise.

Definition 17 however does not consider the processing times that each operation may have. To account for this aspect the marking vector $\mathbf{m}(\tau)$ is split in two components: ${}^{\bullet}\mathbf{m}(\tau)$ and $\mathbf{m}^{\bullet}(\tau)$ such that $\mathbf{m}(\tau) = {}^{\bullet}\mathbf{m}(\tau) + \mathbf{m}^{\bullet}(\tau)$. The vector ${}^{\bullet}\mathbf{m}(\tau)$ collects the tokens that can be consumed as soon as the transitions fire; if the transitions have been fired and the time associated with the places whose transitions have been fired has lapsed, the tokens of ${}^{\bullet}\mathbf{m}(\tau)$ are moved in the vector $\mathbf{m}^{\bullet}(\tau)$ and the output transitions can be fired. Specifically, the *preconditions marking vector* ${}^{\bullet}\mathbf{m} \in \mathbb{N}_0^{|U|+|O|+|X|+|Y|}$ is defined as

$$ {}^{\bullet}\mathbf{m}(\tau) = \left[\mathbf{u}(\tau)^T \quad {}^{\bullet}\mathbf{t}(\tau)^T \quad \mathbf{c}(\tau)^T \quad \mathbf{0}_{|Y|}^T\right]^T, \tag{6} $$

whereas the *postconditions marking vector* $\mathbf{m}^{\bullet} \in \mathbb{N}_0^{|U|+|O|+|X|+|Y|}$ as

$$ \mathbf{m}^{\bullet}(\tau) = \left[\mathbf{0}_{|U|}^T \quad \mathbf{t}^{\bullet}(\tau)^T \quad \mathbf{0}_{|O|}^T \quad \mathbf{y}(\tau)^T\right]^T, \tag{7} $$

in which ${}^{\bullet}\mathbf{t}(\tau) \in \mathbb{N}_0^{|O|}$ and $\mathbf{t}^{\bullet}(\tau) \in \mathbb{N}_0^{|O|}$ collect the tokens of the operations that can potentially be started and the operations that have been completed at time τ.

We can now provide the logical equations describing the evolution of the discrete event system [2]. The firing rule vector $\mathbf{x}(\tau)$ at time τ can be computed as

$$ \mathbf{x}(\tau) = \neg\left(F \odot \neg\, \varphi\left({}^{\bullet}\mathbf{m}(\tau)\right)\right). \tag{8} $$

The marking vectors ${}^{\bullet}\mathbf{m}(\tau)$ and $\mathbf{m}^{\bullet}(\tau)$ can be updated as

$$ \begin{aligned} {}^{\bullet}\mathbf{m}(\tau+1) &= {}^{\bullet}\mathbf{m}(\tau) - F^T\,\mathbf{x}(\tau), \\ \mathbf{m}^{\bullet}(\tau+1) &= \mathbf{m}^{\bullet}(\tau) + \mathrm{H}\,\mathbf{x}(\tau). \end{aligned} \tag{9} $$

In the next section we are going to describe how the matroid framework and the Petri Net can be integrated to solve the multi-agent task allocation problem with precedence constraints, whose definition is given hereinafter.

Problem 4 (Multi-Agent Task Allocation and Scheduling). Consider the same setting as Problem 3. In addition, consider possible precedence constraints between pairs of operations encoded by the discrete event system introduced in eq. (8) and eq. (9). Furthermore, consider that each operation $o \in O$ requires a processing time p_{ado} that depends on the agent a and device d executing it. Finally, we assume that the given sets A, D, O, and $\mathcal{I}_{a_i}$ with $a_i \in A$ admit a feasible solution to the problem.

5 Integrated Framework

In order to solve Problem 4 we propose a framework that tries to assign and schedule at each time τ the operations that are both conflict-free and independent in the agents' independence sets. This procedure is modelled by the discrete event system introduced in eq. (8) and eq. (9) which triggers the allocation of feasible operations with the greedy algorithm introduced in Algorithm 1 - Greedy algorithm's pseudocode for Problem 1. each time an input event is enabled or an operation is completed, until all operations have been allocated and scheduled. The steps of the proposed framework are summarized in Algorithm 2.

Algorithm 1 *Task Allocation and Scheduling*

Require: Independence oracle Ψ.
Input: $\boldsymbol{u}$.
Output: $\boldsymbol{y}$.
1: **Init:** $\tau = 1$, $\overline{\boldsymbol{y}} = \mathbf{0}_{|O|}$, ${}^{\bullet}\boldsymbol{m}(0) = \mathbf{0}_{|O|}$, $\boldsymbol{m}^{\bullet}(0) = \mathbf{0}_{|O|}$.
2: **while** $\exists i \in \{1, \ldots, |O|\}$ such that $\overline{\boldsymbol{y}}_i == 0$ **do**
3: Update external input vector $\boldsymbol{u}(\tau)$;
4: Extract ${}^{\bullet}\boldsymbol{t}(\tau)$ from ${}^{\bullet}\boldsymbol{m}(\tau)$;
5: Extract $\boldsymbol{y}(\tau)$ and $\boldsymbol{t}^{\bullet}(\tau)$ from $\boldsymbol{m}^{\bullet}(\tau)$;
6: **if** An operation o_i has been fully processed **then**
7: Update ${}^{\bullet}\boldsymbol{t}_i(\tau) = h$ with $h \in \mathbb{N}$ number of tokens associated to operation o_i;
8: **end if**
9: **if** $\boldsymbol{u}_i(\tau) == 1 \vee {}^{\bullet}\boldsymbol{m}_j(\tau) == 1$ for $i \in \{1, \ldots, |U|\}$ and $j \in \{1, \ldots, |O|\}$ **then**
10: $\mathcal{S}$ = Algorithm 1 with ground set $\overline{E}(\tau)$ defined as in eq. (10);
11: Compute allocation vector $\boldsymbol{z}(\tau)$ as defined in eq. (11);
12: Compute control vector $\boldsymbol{c}(\tau)$ as defined in eq. (12);
13: **else**
14: $\boldsymbol{c}(\tau) = \mathbf{0}_{|X|}$;
15: **end if**
16: Compute preconditions marking vector ${}^{\bullet}\boldsymbol{m}(\tau)$ as defined in eq. (6);
17: Compute rule vector $\boldsymbol{x}(\tau)$ as defined in eq. (8);
18: Update preconditions and postconditions marking vector ${}^{\bullet}\boldsymbol{m}$ and $\boldsymbol{m}^{\bullet}$ for time $\tau + 1$ as in eq. (9);
19: $\overline{\boldsymbol{y}} = \overline{\boldsymbol{y}} \vee \boldsymbol{y}(\tau)$;
20: $\tau = \tau + 1$;
21: **end while**

Algorithm 2 – Framework algorithm's pseudocode.

The framework initializes at line 1 the preconditions and postconditions marking vector at time 0 and vector $\overline{\mathbf{y}} \in \mathbb{B}^{|O|}$ as vectors of zeros, as well as setting time $\tau = 1$. The vector $\overline{\mathbf{y}}$ keeps track of the operations that have been completed and is used to conclude the execution of the algorithm; indeed, as shown at line 2 the algorithm continues to iterate until all entries of $\overline{\mathbf{y}}$ are equal to 1. Vectors ${}^{\bullet}\mathbf{t}(\tau)$, $\mathbf{t}^{\bullet}(\tau)$, and $\mathbf{y}(\tau)$ are updated according to the composition of the preconditions and postconditions marking vector defined in eq.

(6) and eq. (7) at lines 4–5. If any operation is completed, lines 6–8 update the vector ${}^{\bullet}\mathbf{t}(\tau)$ adding the tokens necessary for the activation of the operations that can be executed after the completion of the first one. At lines 9–13 the allocation of the available operations is made if the vectors $\mathbf{u}(\tau)$ or ${}^{\bullet}\mathbf{t}(\tau)$ have been enabled. Specifically, Algorithm 1 is run using a restricted ground set $\overline{E}(\tau)$ which contains only the triplets of the operations that have no precedence constraints at time τ, i.e., $\overline{E}(\tau) \subseteq E$ is defined as

$$\overline{E}(\tau) = \{(a_i, d_j, o_k): a_i \in \mathrm{A}, d_j \in \mathrm{D}, o_k \in O, \text{with } o_k \text{ allowed to start at time } \tau.\}. \quad (10)$$

The resulting schedule $\mathcal{S}$ is then used to compute an allocation vector $\mathbf{z}(\tau) \in \mathbb{N}_0^{|O|}$ in which each k-th entry is defined as

$$\mathbf{z}_k(\tau) = \begin{cases} a_i & \text{if operation } k \text{ has been assigned to agent } a_i, \\ 0 & \text{otherwise,} \end{cases} \quad (11)$$

with $k \in \{1, \dots, |O|\}$. Finally, the control vector $\mathbf{c}(\tau)$ can be computed using the following law

$$\mathbf{c}(\tau) = \left(H^{o^T} \odot \varphi(\mathbf{z}(\tau))\right) \vee \left(F^o \, {}^{\bullet}\mathbf{t}(\tau) \ \wedge \ H^{y^T} \mathbf{1}_{|Y|}\right). \quad (12)$$

If no update of the vectors $\mathbf{u}(\tau)$ or ${}^{\bullet}\mathbf{t}(\tau)$ occurs, then the control vector is set to a vector of zeros at line 14. The preconditions marking vector ${}^{\bullet}\mathbf{m}(\tau)$ is then updated with the computed $\mathbf{c}(\tau)$ and the possibly updated vectors $\mathbf{u}(\tau)$, ${}^{\bullet}\mathbf{t}(\tau)$ as defined in eq. (6) (line 16). At this point the discrete event system can be updated first computing the rule vector $\mathbf{x}(\tau)$ as in eq. (8) and then updating the preconditions and postconditions marking vector as described in eq. (9) (lines 17–18). Finally, the logical "or" operation is computed on the vector $\overline{\mathbf{y}}$ to keep track of completion events occured at time τ (line 19), and the time τ is incremented (line 20).

We also want to recall that the considered problem is NP-Hard (supposing that $P \neq NP$) and can hence not be solved optimally in polynomial time. Our framework is able to achieve a $\frac{1}{2}$−approximate solution when no precedence constraints are involved, as detailed in Corollary 1. When precedence constraints are considered, our framework is only able to guarantee that the allocations made at each time step τ are such that each solution (allocation) guarantees a $\frac{1}{2}$−approximation ratio. However, no guarantees on the optimality of the overall solution can be given at the current time. Future works will investigate provable approximation algorithm for guaranteeing a specific optimality ratio for the provided schedule.

6 Numerical Validation

In this section we numerically validate the multi-agent task allocation and scheduling framework for agronomic operations with three sets of examples: in the first one we show how Algorithm 1 can be applied to obtain an allocation of tasks for the team of three agents introduced in Section 3; in the second one we provide the results of an abstract scenario which involves 5 agents, 10 operations, and 6 devices; finally, in the third one we provide the results of the allocation and scheduling of a representative scenario of our farming activities.

6.1 Task Allocation

In this section we present the results of the task allocation framework using Algorithm 1, i.e., the problem of allocating a team of agents when no precedence constraints among the farming operations are present. Specifically, we consider the scenarios introduced in Figure 1 and Figure 2 where each triplet $(a_i, d_j, o_k) \in E$ has been associated to a positive integer processing time $p_{ijk} \in \mathbb{N}$. The utility function u introduced in eq. (5) has been defined as to minimize the processing time as $1/p_{ijk}$ for all triplets in the ground set.

The processing time p_{ijk} for the three agents are reported in the following three matrices where the rows are associated to the devices and the columns to the operations:

$$a_1: \begin{bmatrix} 2 & 1 & 3 \\ 4 & 5 & 5 \\ 5 & 10 & 4 \\ 5 & 4 & 2 \\ 5 & 3 & 8 \end{bmatrix}, \quad a_2: \begin{bmatrix} 6 & 6 & 7 \\ 2 & 9 & 10 \\ 6 & 3 & 10 \\ 2 & 8 & 5 \\ 2 & 1 & 6 \end{bmatrix}, \quad a_3: \begin{bmatrix} 7 & 3 & 1 \\ 1 & 7 & 6 \\ 5 & 7 & 2 \\ 2 & 3 & 9 \\ 3 & 4 & 9 \end{bmatrix}.$$

Notice that even if the processing time are given for all the elements in the ground set E, not every pair of device operation can be assigned to a certain agent. Indeed, the feasibility of the triplet is checked by the independence oracle Ψ during the construction of the greedy solution in Algorithm 1.

The results of the greedy algorithm are shown in Table 1 where the first three columns depict the selected triplets (a_i, d_j, o_k) and the second three columns specify the start, end, and processing time, respectively.

Agent	*Device*	*Operation*	*Start Time*	*End Time*	*Processing Time*
3	2	1	0	1	1
2	4	2	0	8	8
1	3	3	0	4	4

Table 1 – Results of task allocation for the scenario introduced in in Figure 1 and Figure 2.

Figure 4 depicts the Gantt diagram for the same scenario where the horizontal axis depicts the processing times and the vertical one contains the information related to each single operation. Each bar corresponds to an allocation where each color represents a different robot, i.e., blue for robot a_1, red for robot a_2, and yellow for robot a_3.

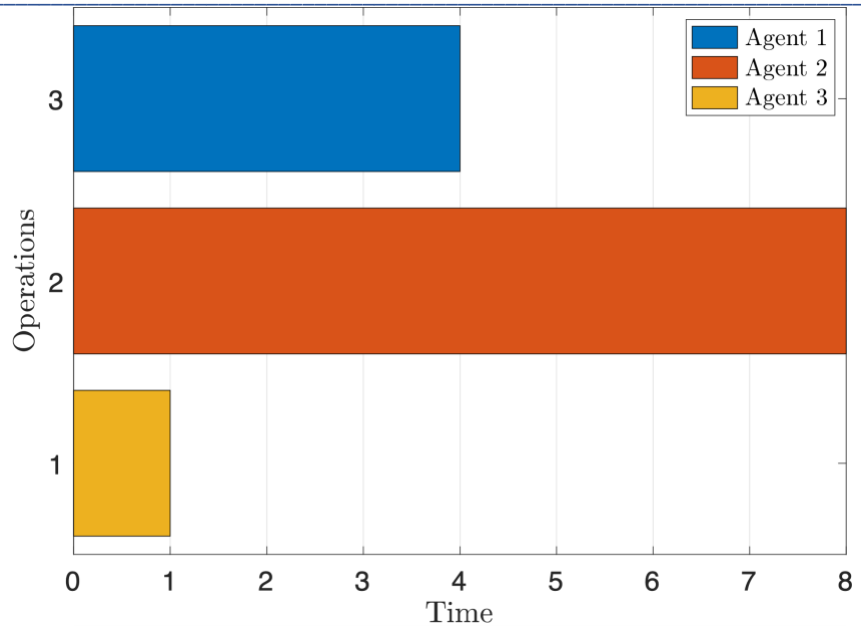

Figure 4 – Gantt diagram for the scenario introduced in Figure 1 and Figure 2.

6.2 Abstract Task Allocation and Scheduling

In this section we illustrate the results of an abstract task allocation and scheduling consisting of a team of 5 agents equipped with devices from a set of 6 elements. There are 10 agronomical operations that need to be allocated and scheduled according to the following precedence constraints:

- Operation o_4 must start after the competition of operation o_6;
- Operation o_5 must start after the competition of operation o_4;
- Operation o_8 must start after the competition of operation o_6;
- Operation o_9 must start after the competition of operation o_8;
- Operation o_{10} must start after the competition of operation o_7.

Figure 5 depicts the operations, the inputs, the outputs, and the rules of the considered scenario as described in the scheduling framework introduced in Section 4.

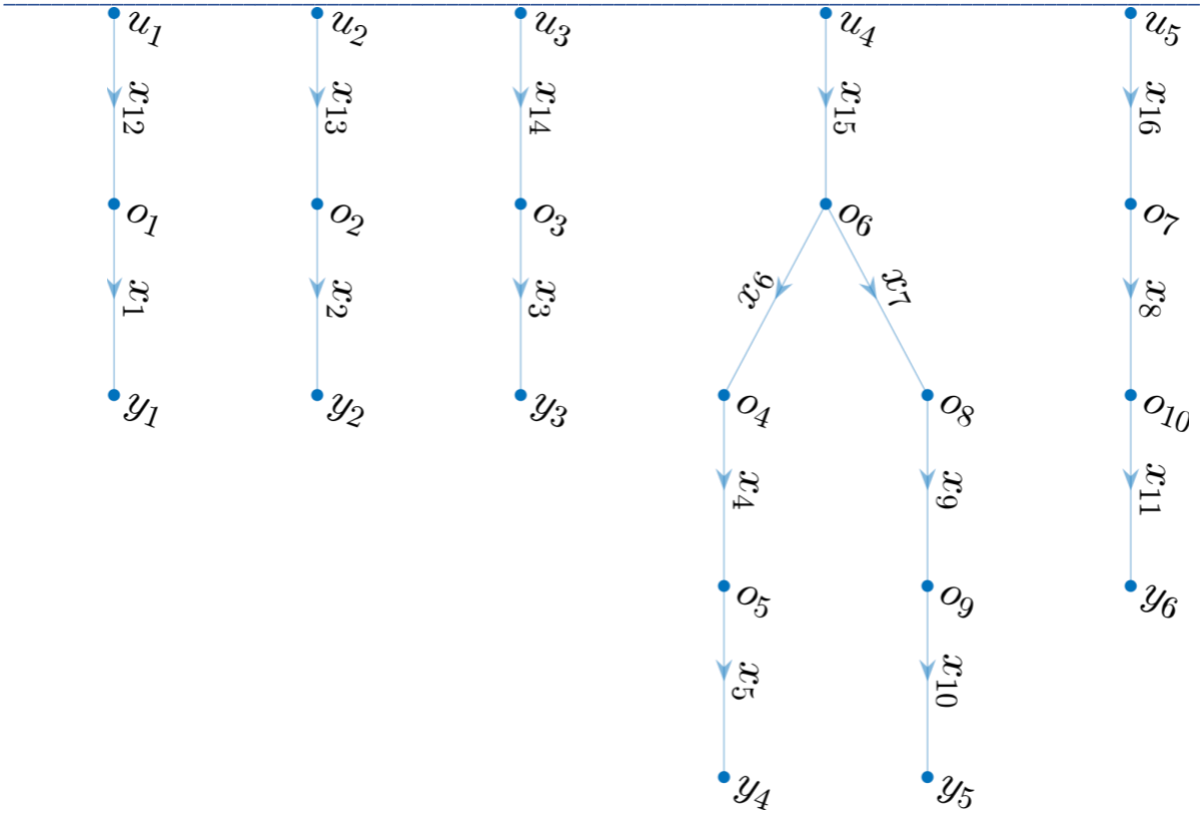

Figure 5 – Graphical description of the precedence constraints on the set of operations considered in the abstract scenario.

Each triplet $(r_i, s_j, o_k) \in E$ has been associated to a processing time $p_{ijk} \in \mathbb{N}$ and a utility u function which is inverse to the processing time p_{ijk} as $u(r_i, s_j, o_k) = \frac{1}{p_{ijk}}$. The inputs are all enabled at the starting time $\tau = 0$, i.e., $\boldsymbol{u}_i(\tau) = 1$ for all $i = 1, \dots, 5$.

Each agent is associated to an independence set $\mathcal{I}_{a_i}$ which for the sake of simplicity is a uniform matroid, i.e., each agent can be assigned to a maximum of k_i operations at the same time.

Algorithm 2 is used to determine the allocation and scheduling of the 10 operations. The results are shown in Table 2. As shown, the precedence constrains are verified since operation o_4 and o_8 starts at time $\tau = 8$ after the completion of operation o_6 at the same time. Similarly, operations o_5, o_9, and o_{10} all starts after the completion of their respective required operation.

Agent	*Device*	*Operation*	*Start Time*	*End Time*	*Processing Time*
3	2	2	0	6	6
5	4	3	0	9	9
2	3	7	0	9	9
1	1	6	0	8	8
4	4	1	0	8	8
3	1	8	8	11	3
5	4	4	8	11	3
2	3	10	9	10	1
1	1	9	11	20	9
4	2	5	11	19	8

Table 2 – Results for the abstract task allocation and scheduling.

Figure 6 depicts the Gantt diagram of the results of the considered scenario in which each color is associated to a different agent. Each row depicts the scheduling of the related operation over the time window $\tau \in [0,20]$. Furthermore, the presence of matroidal constrains can also be seen by the allocations of agent a_5 which can perform operation o_3 and o_4 at the same time.

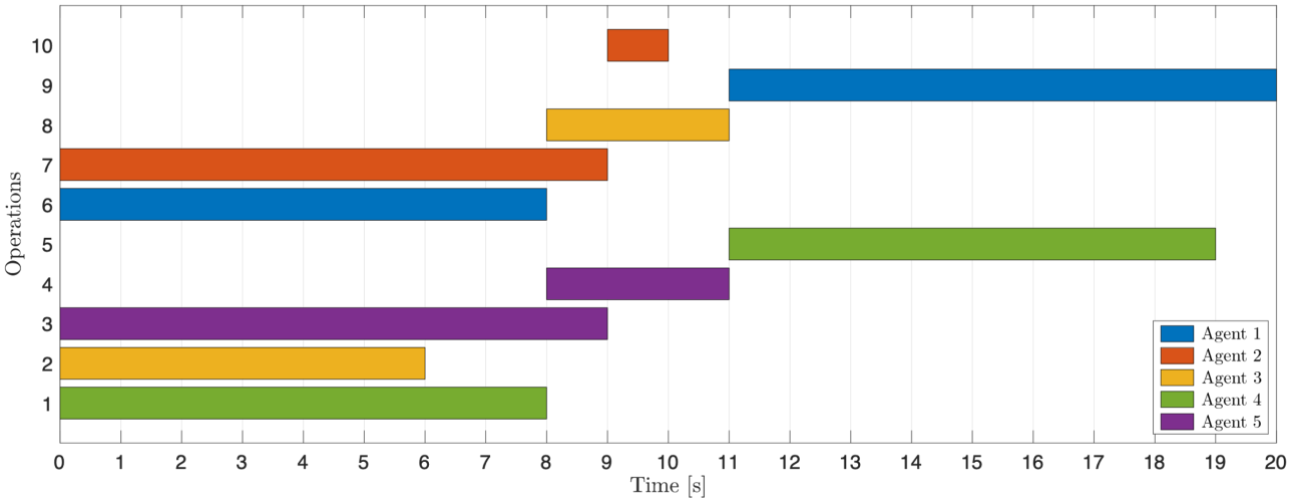

Figure 6 – Gantt diagram for the abstract task allocation and scheduling consisting of 5 agents and 10 agronomical operations.

6.3 Precision Farming Activities Allocation and Scheduling

Motivated by the agronomical activities involved in the project, in this section we illustrate the proposed framework to our agronomical context which involves a total of 4 agents: 2 ground robots, 1 aerial robot, and 1 agronomical expert.

The agronomical operations that we intend to allocate and then schedule, are the following:

- o_1: Tree Geometry Reconstruction;
- o_2: Pest and Disease Management;
- o_3: Water Stress Assessment;
- o_4: Suckers' Detection;
- o_5: Suckers' Management;
- o_6: Bugs' Detection;
- o_7: Bugs' Management;
- o_8: Pruning Management.

The set of agents at our disposal are the following:

- a_1: UGV 1;
- a_2: UGV 2;
- a_3: UAV;
- a_4: Agronomical expert.

The set of devices that can be equipped by the agents are the following:

- d_1: RGB camera;

- d_2: RGBD camera;
- d_3: Multispectral camera;
- d_4: Thermal camera;
- d_5: FARO lidar;
- d_6: Herbicide sprayer;
- d_7: Bugs' traps;
- d_8: Scissors.

The following precedence constraints are considered in our agronomical scenario:

- Suckers' Management (o_5) can be performed only after Suckers' Detection (o_4);
- Pruning Management (o_8) can be performed only after Tree Geometry Reconstruction (o_1);
- Bugs' Management (o_7) can be performed only after Bugs' Detection (o_6).

The list of operations and precedence constraints are shown in Figure 7. The input events are all enabled at the starting time $\tau = 0$, i.e., $\boldsymbol{u}_i(\tau) = 1$ for all $i = 1, \dots, 5$.

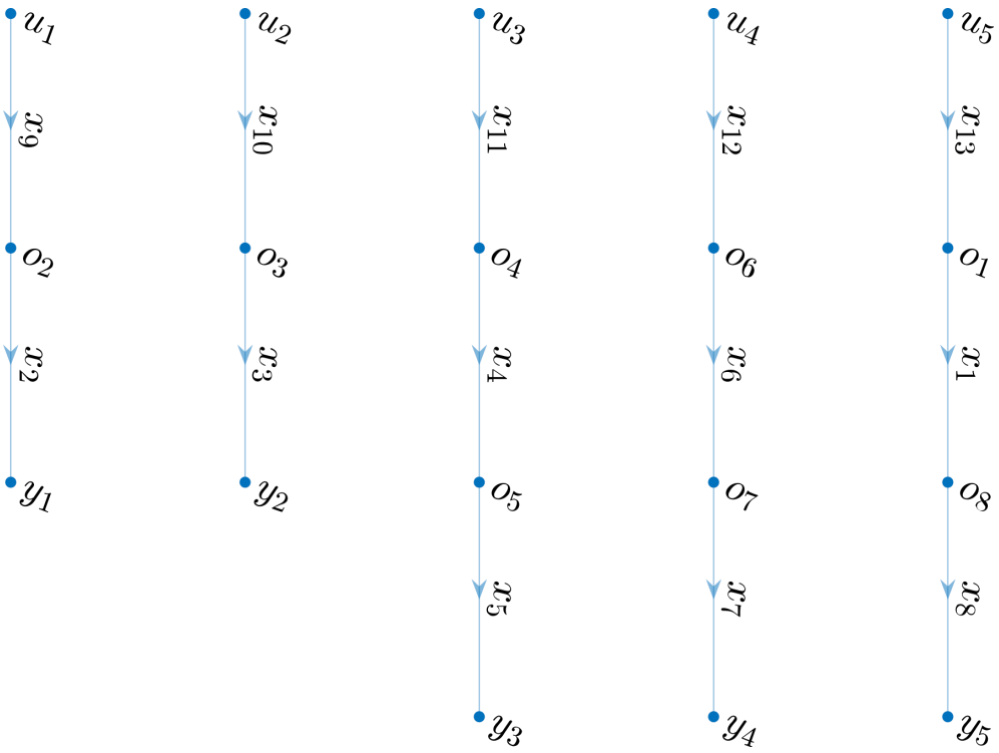

Figure 7 – Graphical representation of the precedence constraints on the set of operations considered scenario.

The independence sets for the agents are defined as follows.

The independence set for UGV 1 is:

$$\mathcal{I}_{a_1} = \{ \quad \{\emptyset\}, \{(a_1, d_1, o_4)\}, \{(a_1, d_2, o_4)\}, \{(a_1, d_1, o_6)\}, \{(a_1, d_2, o_6)\}, \\ \{(a_1, d_1, o_4), (a_1, d_1, o_6)\}, \{(a_1, d_2, o_4), (a_1, d_2, o_6)\}, \\ \{(a_1, d_1, o_1), (a_1, d_3, o_1), (a_1, d_5, o_1)\}, \{(a_1, d_2, o_1), (a_1, d_3, o_1), (a_1, d_5, o_1)\}, \\ \{(a_1, d_1, o_1), (a_1, d_2, o_1), (a_1, d_3, o_1), (a_1, d_5, o_1)\} \quad \}.$$

The independence set for UGV 2 is:

$$\mathcal{I}_{a_2} = \{ \quad \{\emptyset\}, \{(a_2, d_2, o_4)\}, \{(a_2, d_2, o_6)\}, \{(a_2, d_6, o_5)\}, \\ \{(a_2, d_2, o_4), (a_2, d_6, o_5)\}, \{(a_2, d_2, o_6), (a_2, d_6, o_5)\} \quad \}.$$

The independence set for UAV is:

$$\mathcal{I}_{a_3} = \{ \quad \{\emptyset\}, \{(a_3, d_1, o_2)\}, \{(a_3, d_3, o_2)\}, \{(a_3, d_1, o_2), (a_3, d_3, o_2)\}, \\ \{(a_3, d_3, o_3), (a_3, d_4, o_3)\}, \{(a_3, d_1, o_2), (a_3, d_3, o_2), (a_3, d_3, o_3), (a_3, d_4, o_3)\} \quad \}.$$

The independence set for the agronomical expert is:

$$\mathcal{I}_{a_4} = \{ \quad \{\emptyset\}, \{(a_4, d_7, o_7)\}, \{(a_4, d_8, o_8)\} \quad \}.$$

As done in the previous examples, each triplet is associated to a processing time p_{ijk} which defines also its utility $u(a_i, d_j, o_k)$ in the solution. The matrices defining the processing times for each agent are the following:

$$a_1: \begin{bmatrix} 7 & 9 & 4 & 7 & 7 & 5 & 5 & 1 \\ 7 & 2 & 6 & 5 & 4 & 1 & 6 & 6 \\ 3 & 5 & 5 & 6 & 8 & 8 & 2 & 8 \\ 5 & 7 & 9 & 7 & 5 & 4 & 8 & 1 \\ 2 & 1 & 9 & 10 & 6 & 5 & 4 & 7 \\ 1 & 4 & 9 & 9 & 6 & 10 & 2 & 3 \\ 6 & 7 & 5 & 2 & 2 & 9 & 8 & 5 \\ 5 & 3 & 2 & 10 & 4 & 3 & 6 & 1 \end{bmatrix},$$

$$a_2: \begin{bmatrix} 6 & 10 & 9 & 3 & 5 & 5 & 4 & 4 \\ 10 & 8 & 5 & 8 & 7 & 6 & 9 & 8 \\ 4 & 9 & 1 & 2 & 10 & 6 & 7 & 9 \\ 1 & 2 & 3 & 1 & 1 & 8 & 5 & 4 \\ 9 & 9 & 3 & 7 & 6 & 10 & 6 & 4 \\ 10 & 5 & 7 & 4 & 5 & 4 & 10 & 7 \\ 5 & 3 & 10 & 5 & 7 & 8 & 7 & 3 \\ 10 & 6 & 6 & 1 & 9 & 7 & 7 & 1 \end{bmatrix},$$

$$a_3: \begin{bmatrix} 3 & 10 & 1 & 6 & 5 & 2 & 3 & 2 \\ 7 & 7 & 10 & 4 & 3 & 2 & 9 & 5 \\ 6 & 9 & 5 & 8 & 9 & 8 & 8 & 8 \\ 6 & 8 & 2 & 2 & 4 & 1 & 2 & 6 \\ 7 & 10 & 2 & 4 & 10 & 5 & 3 & 2 \\ 6 & 4 & 6 & 2 & 3 & 1 & 7 & 4 \\ 10 & 4 & 7 & 7 & 7 & 10 & 8 & 8 \\ 9 & 1 & 9 & 6 & 8 & 9 & 4 & 5 \end{bmatrix},$$

$$a_4: \begin{bmatrix} 8 & 3 & 10 & 4 & 2 & 3 & 7 & 10 \\ 2 & 1 & 5 & 5 & 9 & 2 & 8 & 8 \\ 7 & 7 & 4 & 10 & 9 & 9 & 9 & 9 \\ 8 & 10 & 5 & 8 & 7 & 8 & 7 & 5 \\ 9 & 6 & 3 & 5 & 9 & 4 & 8 & 4 \\ 3 & 4 & 3 & 10 & 1 & 6 & 2 & 7 \\ 10 & 4 & 9 & 9 & 7 & 3 & 8 & 6 \\ 6 & 5 & 9 & 5 & 9 & 4 & 10 & 1 \end{bmatrix},$$

where each row is associated to a device and each column is associated to an operation. We recall that the processing times are given for all triplets in the ground set but the solution is constructed by first evaluating their feasibility.

The problem is solved applying Algorithm 2. The results of the allocation and scheduling are summarized in Table 3. For example, UGV 1 (agent a_1) is scheduled to perform Bugs' Detection (operation o_6) using RGBD camera (device d_2) at time $\tau = 0$ and Tree Geometry Reconstruction (operation o_1) using FARO lidar sensor (device d_5) at time $\tau = 1$, whereas UGV 2 (agent a_2) is scheduled to perform Suckers' Detection (operation o_4) using RBGD camera (device d_2) at time $\tau = 0$ and Suckers' Management (operation o_5) using the herbicide sprayer (device d_6) at time $\tau = 8$.

Agent	*Device*	*Operation*	*Start Time*	*End Time*	*Processing Time*
1	2	6	0	1	1
3	4	3	0	2	2
2	2	4	0	8	8
1	5	1	1	3	2
4	7	7	1	9	8
3	3	2	2	11	9
2	6	5	8	13	5
4	8	8	9	10	1

Table 3 – Results of the allocation and scheduling of the precision farming activities.

Note that the devices are assumed unique for each agent, i.e., each robot or individual is equipped with its own copy. This implies that no device precedence constraints on the use of the same devices have to be considered in our formulation.

Figure 8 depicts the Gantt diagram of the results of the considered scenario in which each color is associated to a different agent. Each row depicts the scheduling of the related operation over the time window $\tau \in [0,14]$. As shown, the precedence constraints between the pairs (o_4, o_5), (o_6, o_7), and (o_1, o_8) are all satisfied.

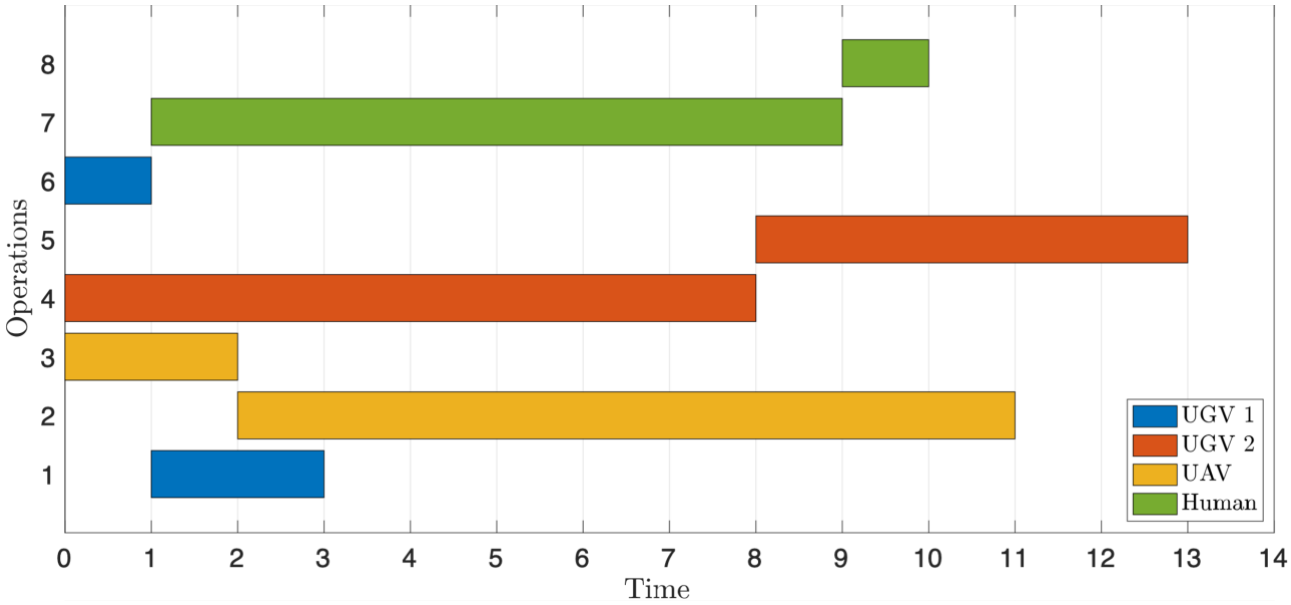

Figure 8 – Gantt diagram for the precision farming activities.

7 References

[1] R. K. Williams, A. Gasparri and G. Ulivi, "Decentralized Matroid Optimization for Topology Constraints in Multi-Robot Allocation Problems," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[2] D. Tacconi and F. Lewis., "A new matrix model for discrete event systems: application to simulation," *IEEE Control Systems Magazine,* vol. 17, no. 5, 1997.

[3] D. D. Paola, A. Gasparri, D. Naso and F. L. Lewis., "Decentralized dynamic task planning for heterogeneous robotic networks," *Autonomous Robots,* vol. 38, 2015.

[4] S. Im, B. Moseley, K. Pruhs and M. Purohit., "Matroid coflow scheduling," in *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, 2019.

[5] L. F. O. Chamon, A. Amice and A. Ribeiro, "Approximately Supermodular SchedulingSubject to Matroid Constraints," *IEEE Transactions on Automatic Control,* 2021.

[6] W. Ouyang, M. S. Obaidat, X. Liu, X. Long, W. Xu and T. Liu, "Importance-Different Charging Scheduling Based on Matroid Theory for Wireless Rechargeable Sensor Networks," *IEEE Transactions on Wireless Communications,* 2021.

[7] J. Oxley, Matroid Theory, Oxford University Press, 1992.

[8] M. L. Fisher, G. L. Nemhauser and L. A. Wolsey., "An analysis of approximations for maximizing submodular set functions – II," 1978.

[9] G. L. Nemhauser, L. A. Wolsey and M. L. Fisher., "An analysis of approximations for maximizing submodular set functions – I," *Mathematical Programming,* vol. 14, no. 1, 1978.