Noname manuscript No.
(will be inserted by the editor)

Decentralized Dynamic Task Planning for Heterogeneous Robotic Networks

Donato Di Paola · Andrea Gasparri · David Naso · Frank L. Lewis

Received: date / Accepted: date

Abstract In this paper, we propose a decentralized model and control framework for the assignment and execution of tasks, i.e. the dynamic task planning, for a network of heterogeneous robots. The proposed modeling framework allows the design of missions, defined as sets of tasks, in order to achieve global objectives regardless of the actual characteristics of the robotic network. The concept of skills, defined by the mission designer and considered as constraints for the mission execution, is exploited to distribute tasks across the robotic network. In addition, we develop a decentralized control algorithm, based on the concept of skills for decoupling the mission design from its deployment, which combines task assignment and execution through a consensus-based approach. Finally, conditions upon which the proposed decentralized formulation is equivalent to a centralized one are discussed. Experimental results are provided to validate the effectiveness of the proposed framework in a real-world scenario.

Keywords Heterogenous Multi-Robot Systems, Task Sequencing, Distributed Cooperation.

D. Di Paola
Institute of Intelligent Systems for Automation (ISSIA), National Research Council (CNR), 70126 Bari, Italy
E-mail: dipaola@ba.issia.cnr.it
A. Gasparri
Engineering Department, University "Roma Tre", 00146 Rome, Italy
E-mail: gasparri@dia.uniroma3.it
D. Naso
Department of Electrical and Electronic Engineering (DEE), Polytechnic of Bari, 70126 Bari, Italy
E-mail: naso@poliba.it
F. L. Lewis
Automation and Robotics Research Institute, University of Texas at Arlington, Fort Worth, TX 76118-7115, USA
E-mail: lewis@uta.edu

1 Introduction

The research on robotic networks (RNs) has been very active for over two decades [16]. RNs are groups of multi-functional robots (often called agents) interconnected by means of a communication network and capable to perform a variety of tasks such as measuring, finding, tracking, following, or manipulating objects. RNs can be formed by identical robots or by robots having different functionalities.

The area of RNs is a challenging domain, rich of multifaceted aspects that can be attacked from many different viewpoints. Typically, the state of a mobile, autonomous robot is a combination of mixed continuous-time (position, battery charge level) and discrete-event (busy/idle, reachable/ unreachable) information [14], [15]. When considering tracking [26], exploration [2], mapping [1], localization or formation [6] problems, RNs present significant affinities with research works in the area of multi-agent systems and distributed control over networks [28], in which problems are amenable to continuous-time or discrete-time models and control laws [6], [20], or probabilistic decision-making algorithms [3]. Besides these issues, when dealing with heterogeneous RNs, i.e., a RN in which each robotic agent might differ in terms of dynamics, and/or sensing and computation capabilities, the control strategy has also to deal with inherently combinatorial and discrete-event driven problems. In particular, while the continuous or discrete time information is mostly restricted within the single robots (e.g., individual trajectory controllers handle the navigation of each robot independently without needing to share information between each other), the RNs as a whole is a discrete-event system, in which the robot operating condition changes, and the corresponding effects occurs only in correspondence of new events (e.g.,

target found, intrusion detected, mission completed). Handling the overall control of the RNs from this higher level viewpoint is difficult, because the control system must simultaneously address task planning, dynamic task sequencing [11], resolution of conflicts for shared resources [12], and event-based feedback control [25], [23].

Several frameworks for modeling, planning and controlling a RN as a discrete event system have been proposed. In this context, the Petri Net (PN) formalism proved to be an effective tool [5], [29], [17]. For example, stochastic PNs are used in [5] to develop a framework which enables modeling a robot task and analyzing its qualitative and quantitative properties. In addition, PNs are combined with Monte Carlo methods in [29] to build a generic hybrid monitoring approach which allows the detection of inconsistencies in the navigation of autonomous mobile robots using online generated models. Furthermore, logical PNs are exploited in [17] to represent the Gantt chart of a sequence of operations to be performed by limited stationary or mobile resources placed at different locations in a fixed sequence, with fixed (or roughly specified within time-windows) service times.

In this work we propose an approach to model, analyze and control a RN using a Matrix-based Discrete Event control Framework (MDEF) [24]. This framework, which has been proven to be equivalent to Petri Nets (PNs), has been applied to a variety of complex, large-scale systems including manufacturing systems [18], material handling systems, and warehouses [13]. However, RNs present substantial differences from the mentioned indoor and structured systems in which the use of a central discrete-event supervisor is an ideal choice. On the other hand, modern communication technologies based on networked environments [27], and the ever-wide availability of computing resource onboard of the robots stimulates the research on decentralization of decision and control functions also at the higher discrete-event supervisory level of the controller. The proposed framework is modular, transparent, and based on a set of compact matrix equations, which can be used to simultaneously model and control the discrete-event dynamics of the RN. In this framework, robots are able to organize, coordinate and synchronize their activities in an autonomous way. The coherence of the information necessary to avoid conflicts and dead-locks is guaranteed by means of a consensus-based mechanism on a restricted set of robot state variables. Furthermore, a decoupling mechanism is proposed to detach the activities plan from its actual execution. This is achieved by dynamically choosing which robot is doing what according to both the network availability and the robots capabilities each time a new event occurs. To the best of our knowledge, this paper is the first effort to provide a thorough rationalization of the multiple decision problems arising in a heterogeneous RN by means of a unifying modeling and decentralized control approach that is formally proven to be equivalent to a network of centrally supervised robots. Notably, some of the aspects covered by this paper were preliminarily discussed in our earlier contributions [8], [7].

To summarize the following contributions are made in this paper:

i) A rigorous and unifying approach which can incorporate any user-defined strategy for the dynamic task planning, i.e. the modeling and the control of the execution of groups of tasks (missions), and for the online task assignment to the heterogeneous robots;

ii) A decoupling mechanism of the structure of missions from the network of agents, by introducing the concept of skills. This mechanism allows to decompose missions into distributed models for the decentralized task execution control and task assignment systems on board each robot;

iii) A real-world implementation of a heterogeneous RN, operating in an indoor environment to underline the feasibility of the proposed decentralized dynamic task planning system.

2 Problem Statement

In this section we define the Dynamic Task Planning (DTP) problem that is solved in this paper. Before formalizing the DTP problem, we introduce some basic definitions.

2.1 Heterogeneity and Mission Structure

Consider a heterogeneous network $V = \{v_1, \ldots, v_n\}$ of n robotic agents and m tasks $T = \{t_1, \ldots, t_m\}$ which have to be accomplished by the network. The network V is referred to as *heterogeneous* because robotic agents might differ in terms of dynamics, and/or sensing and computation capabilities, e.g., agents might be equipped with different sensors, such as short range cameras, laser scanners or sonars, or might have different payloads, or might have different kinematics. The concept of *skills* is introduced to model this heterogeneity. Skills either reflect the structural capabilities of the robotic agent (for instance, agents can be equipped with different sensors, such as short range cameras, laser scanners or sonars, or agents can have different payloads) or allow to define abstract categories in order to establish a hierarchy among agents (for instance, some agents

might serve as a communication network for the others). Thus, we define $S = \{s_1, \ldots, s_p\}$ as the set of all skills owned by all agents in V. Moreover, we define $S_{v_i} \subseteq S$ as the set of skills owned by the agent $v_i \in V$, thus $\cup_{v_i \in V} S_{v_i} = S$. Since the same skill can be owned by different robots in the network, the following condition may occur, $\cap_{v_i \in V} S_{v_i} \neq \emptyset$

In our framework, a *task* is a generic job or action which can be executed by a single robot (e.g. reaching a target, measuring the temperature, finding an object in a real-world environment, etc.). A robot can execute a single task at time and a task have to be executed by a robot at time. This means that we consider the cooperation among robots not for a single task but for a set of tasks with the same goal. This concept will be better explained later in this section. Tasks are non-preemptive [21], i.e. the execution of a task cannot be resumed if it has been interrupted.

Moreover, a task is the atomic unit of work that an agent can execute exploiting its own skills. Thus, given the task $t_j \in T$, we define R_{t_j} as the set of skills required by the task t_j to be executed by an agent. Furthermore, let us define $R = \{s_1, \ldots, s_r\}$ as the set of all skills required by all tasks in T, i.e. $R = \cup_{t_j \in T} R_{t_j}$. Also in this case, the same skill can be required by different tasks, thus we may obtain $\cap_{v_i \in V} R_{v_i} \neq \emptyset$. Therefore, the generic task t_j can be executed by the generic robot v_i if and only if $R_{t_j} \subseteq S_{v_i}$.

In order to define how to group tasks to achieve a single goal, let us now introduce the following concept:

Definition 1 (Mission) A mission w is defined by the tuple $\langle T^w, R^w, X^w, U^w, Y^w \rangle$, where $T^w \subseteq T$ is the set of *tasks* involved in the mission w; $R^w \subseteq R$ is the set of *skills* required by the tasks in T^w; X^w is the set of *rules*, which are symbols that describe the logical links among pairs of tasks, within the mission w. These symbols determine the precedence rules of tasks in the execution flow of the mission w; finally U^w and Y^w are the sets of *inputs* and *output events*, respectively. Input events in U^w are symbols used to indicate the conditions which trigger the start of the mission w (a sensory input, a user command, etc.). Specifically, each symbol in U^w is univocally related to a single task in T^w. Thus, the mission w starts when one or more tasks, related to symbols in U^w, start. Output events in Y^w are symbols used to indicate the completion of the mission w. Also in this case, each symbol in Y^w is univocally related to a single task in T^w. The symbols in Y^w indicate which tasks determine the end of the mission w. Using the link between tasks in T^w and events in U^w and Y^w we can model the relationship between the skills in R^w requested to start and to accomplish a given mission w.

Furthermore, we define with $W = \{w_1 \ldots w_q\}$ the set of the q missions that have to be completed by all the robots in the network V. It is worth to notice that, to maintain the abstraction of the concept of task, the generic task t_j can belong only to a specific mission w_χ, i.e. $t_j \in T^{w_\chi}$ and $t_j \notin T^{w_k}\ \forall\ w_k \in W \backslash \{w_\chi\}$. If the same job, in the real world (for instance, carry an object, take a measurement, build a map, etc.), have to be executed within two or more different missions, that job will be labeled as different tasks in T, one for each mission. Thus, the set T can be defined as $T = \cup_{w_k \in W} T^{w_k}$ and, as a result, we obtain $\cap_{w_k \in W} T^{w_k} = \emptyset$.

Remark 1 (Cooperative Mission Planning) It may be useful to note that in the proposed framework we define the concept of cooperation among robots at a mission level rather than at a task level. Thus, we can obtain cooperative planning designing missions in which each robot executes its own task and all the tasks in the mission start with the same input and the end of all the tasks triggers the mission end.

It is worth noticing that, given the set of missions W and the related sets of rules $\{X^1, \ldots X^w\}$, the proposed framework can model different types of execution flows of tasks in T:

- *Parallel execution*: a rule $x_k \in X^{w_\chi}$, in mission w_χ, determines the start of task $t_j \in T^{w_\chi}$ and, at the same time, a rule $x_l \in X^{w_\gamma}$, in mission w_γ, determines the start of task $t_j \in T^{w_\gamma}$;
- *Ordered execution*: a rule $x_k \in X^{w_\chi}$ determines the start of task $t_j \in T^{w_\chi}$ after the termination of the task $t_k \in T^{w_\chi}$;
- *Synchronized execution*: a rule $x_k \in X^{w_\chi}$ determines the start of multiple tasks in T^{w_χ} after the termination of the task $t_k \in T^{w_\chi}$.

2.2 The Dynamic Task Planning Problem

In this paper, given the generic mission $w \in W$, we assume that tasks in T^w can be subject to two categories of constraints: *precedence* constraints and *assignment* constraints. The precedence constraints define the execution order of tasks, i.e. one or more tasks may have to be accomplished before another task is allowed to be started. The assignment constraints define the task-agent assignment according to the skills required by the task and the skills owned by the agent. Keeping in mind this last concept and given the definitions above, we are now ready to define the key issue addressed in this paper:

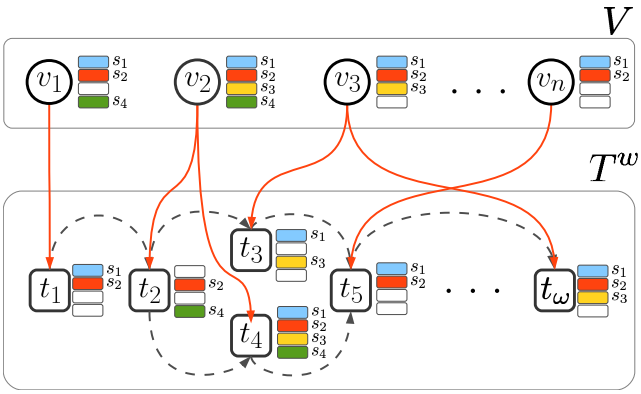


Fig. 1 Abstract representation of the dynamic task planning scenario: we represent the set of agents at the top and the set of tasks, in a mission, with the precedence constraints (gray arrows) at the bottom. The rectangles beside the agents and tasks indicate the skills owned by each agent and required by each task. The orange arrows indicate a possible task to agent assignment.

Problem 1 (Dynamic Task Planning (DTP)) Given a heterogeneous network of robots V and a set of tasks T, on which a set of missions W is defined, design a system which controls missions in W, where the execution is determined by input events in U and output events in Y, ensuring their accomplishment. In the execution of each mission $w_k \in W$, all tasks in T^{w_k} have to match dynamically the precedence constraints defined by rules in X^{w_k} and the assignment constraints expressed by skills in R^{w_k}.

The main objective of this paper is to solve this problem in a decentralized fashion. The DTP problem is defined as "dynamic" because the robots do not have any knowledge about information which change over time, neither on the current situation of the whole network nor on the current state of the mission execution. However, each robot knows a set of static information, i.e. the skills it owns and the structure of the missions that have to be executed. The key idea of the paper is to represent these two different kinds of information by using the concept of skill as a tool to decouple the structure of missions, which contains the static information, from the network of agents, which dynamically executes the missions and fulfills the constraints. In order to solve the DTP problem we propose a decentralized framework which operates in a event-driven fashion. Two different kinds of events are considered: events (more frequents) which trigger the discrete-event control of the mission execution, and events (possibly less frequents) which activate the task assignment algorithm to ensure the assignment constraints expressed by the skills.

The diagram in Fig. 1 gives an overview of the described DTP problem. The circles at the top represent the n robots in V and the rectangles at the bottom represent the set of ω tasks T^w for a generic mission $w \in W$. Tasks are connected by gray dotted arrows which represent precedence constraints. For instance, task t_2 can be executed only after task t_1 is completed, while tasks t_3 and t_4 will start at the same time after the completion of task t_2. The set of skills is represented by a stack of rectangles each one identified by a different color. The set of skills required for the execution of a task R_{t_j} is represented by the stack next to it, while the set of skills owned by a robot S_{v_i} is described by the stack next to it. Note that a white rectangle indicates, for a task the fact that the skill is not required for the execution of the task, while for a robot the fact that the skill is not owned by it. The orange solid arrows from the robots to the tasks indicate a possible assignment. The assignment, driven by a generic objective function (minimize the paths of all robots, minimize the time to visit the task locations, etc.) which determines the best assignment among all possible ones, have to match the constraints between the skills required by the tasks and the skill owned by the robots. For instance, task t_1 can be executed by all robots in the network, while tasks t_4 can be executed only by the robot v_2.

3 Dynamic Task Planning Model

In this section we present the discrete-event mission model of the proposed DTP framework. The key idea is to decouple the design of a set of missions (i.e. how tasks are linked within each mission) from their deployment (i.e. how tasks can be distributed to robots in the network) by introducing the concept of skills. First we describe the centralized mission model, which is independent from the network of robots. Then, we develop an algorithm to decompose the centralized model into local mission models, on the basis of the skills owned by each robot in the network. Furthermore, in the next section, we show how the local model can be used in the cooperative execution of all mission for the entire network.

3.1 Centralized Mission Model

In this subsection we present the Centralized Mission Model (CMM), based on the original matrix-based discrete event model presented in [24]. In this work, we extend this model to deal with a decentralized control scenario by introducing the concept of skills. The reader is referred to the Appendix for an overview on Boolean algebra and matrix operations.

3.1.1 Single Mission Definition

Given the generic mission w belonging to the mission set W, the precedence constraints defined by rules in X^w are represented through the following two sets of Boolean matrices. The set of *preconditions matrices* for each mission $w \in W$ is defined as follows:

Definition 2 (Input matrix) Let $\mathbf{F_u}^w \in \mathbb{B}^{|X^w|\times|U^w|}$ be the boolean input matrix with the entry $\mathbf{F_u}^w_{(j,k)}$ set to "1", if the occurrence of the k-th input of mission w is an immediate prerequisite for the activation of the j-th rule.

Definition 3 (Task-end matrix) Let $\mathbf{F_t}^w \in \mathbb{B}^{|X^w|\times|T^w|}$ be the boolean task-end matrix for the mission w, where an entry $\mathbf{F_t}^w_{(j,k)}$ is set to "1" if the completion of the task k is a necessary prerequisite for the rule j to be fired.

The set of *postconditions matrices* for each mission $w \in W$ is defined as follows:

Definition 4 (Task-start matrix) Let $\mathbf{S_t}^w \in \mathbb{B}^{|T^w|\times|X^w|}$ be the boolean task-start matrix where the entry $\mathbf{S_t}^w_{(j,k)}$ is set to "1" if the task j of mission w starts as the result of the fired rule k.

Definition 5 (Output matrix) Let $\mathbf{S_y}^w \in \mathbb{B}^{|Y^w|\times|X^w|}$ be the boolean output matrix. The entry $\mathbf{S_y}^w_{(j,k)}$ is set to "1" when the j-th output event is a consequence of the activation of the rule k.

Note that, the matrices introduced above provide *static* information, since they do not change over time, concerning the evolution of a generic mission w. By contrast *dynamic* information, such as tasks completion or the occurrence of input and output events, are provided by the following set of vectors, concerning the current iteration τ.

Definition 6 (Input vector) Let $\mathbf{u}^w(\tau) \in \mathbb{B}^{|U^w|}$ be the boolean column vector which describes the occurrence of an input event for the mission w. The element $\mathbf{u}^w_{(j)}(\tau)$ is set to "1" when the j-th triggering event (a threshold on a sensor signal, a user input, etc.) occurs.

Definition 7 (Task vector) Let $\mathbf{t}^w(\tau) \in \mathbb{B}^{|T^w|}$ be the boolean column vector which describes the status of all tasks belonging to mission w. The element $\mathbf{t}^w_{(j)}(\tau)$ is set to "1" if the j-th task is completed and to "0" otherwise.

Definition 8 (Output vector) Let $\mathbf{y}^w(\tau) \in \mathbb{B}^{|Y^w|}$ be the boolean column vector which describes the accomplishment of the mission w. The element $\mathbf{y}^w_{(j)}(\tau)$ is set to "1" when the j-th output event, such as the successfully mission completion, occurs.

The preconditions on completed tasks and the occurrence of input events are linked through rules, i.e., if a given combination of preconditions is verified, then the corresponding rule is fired. The result of the evaluation of logical preconditions for the mission w is summarized by the following boolean vector:

Definition 9 (Rule vector) Let $\mathbf{x}^w(\tau) \in \mathbb{B}^{|X^w|}$ be the boolean column vector which describes the activation of rules for the mission w. The element $\mathbf{x}^w_{(j)}(\tau) = 1$ denotes that rule j is fired, i.e. all the preconditions to rule j are true.

We recall that the key idea proposed in this paper is to decouple agents from tasks by introducing the concept of skills. After defining the task set T^w, which is part of the general mission $w \in W$, we need to define a way to indicate which skills, identified by the elements of the set R^w, among the all possible skills in R are required to perform tasks in T^w. To this end we introduce the following matrix:

Definition 10 (Skill-Task matrix) Let $\mathbf{\Sigma_t}^w \in \mathbb{B}^{|R|\times|T^w|}$ be the skill-task matrix for the mission w. The element $\mathbf{\Sigma_t}^w_{(j,k)} = 1$ indicates that the task k identified by the element $t_k \in T^w$, requires the skill j identified by the element $s_j \in R^w$ among all skills in R.

It is important to note that $\mathbf{\Sigma_t}^w$ can not contain null columns, as this would mean that a task does not require skills to be executed. As a result, we define the following matrix:

Definition 11 (Skill-Rule matrix) Let $\mathbf{\Sigma_x}^w \in \mathbb{B}^{|R|\times|X^w|}$ be the skill-rule matrix for the mission w, obtained by the following expression $\mathbf{\Sigma_x}^w = \mathbf{\Sigma_t}^w \odot \mathbf{S_t}^w$, where $\odot$ is the logical matrix product (see Appendix for further details). The element $\mathbf{\Sigma_x}^w_{(j,k)} = 1$ indicates that the rule k identified by the element $x_k \in X^w$, requires the skill j identified by the element $s_j \in R^w$ among all skills in R.

3.1.2 Multiple-Mission Definition

In order to model the simultaneous evolution of different missions, we consider the set W of the q missions that have to be completed by all robots in the network. Moreover, let us define the following global sets: the global rule set $X = \cup_{w_k \in W} X^{w_k}$, the global input set $U = \cup_{w_k \in W} U^{w_k}$, and the global output set $Y = \cup_{w_k \in W} Y^{w_k}$.

Postconditions and preconditions for all missions in W can be represented by diagonal block matrices, in which a block corresponds to an individual mission. For

instance, the *global input matrix* $\mathbf{F_u} \in \mathbb{B}^{|X|\times|U|}$ is defined as follows:

$$\mathbf{F_u} = \mathtt{diag}(\,\mathbf{F_u}^{w_1}, \mathbf{F_u}^{w_2}, \ldots, \mathbf{F_u}^{w_q}\,),$$

where $\mathbf{F_u}^{w_k}$ corresponds to mission $w_k \in W$. Similarly, we define the *global task-end matrix* $\mathbf{F_t} \in \mathbb{B}^{|X|\times|T|}$, the *global task-start matrix* $\mathbf{S_t} \in \mathbb{B}^{|T|\times|X|}$, and the *global output matrix* $\mathbf{S_y} \in \mathbb{B}^{|Y|\times|X|}$.

In order to complete the multiple -mission model of the missions, let us now introduce the control matrix which allows to control the activation of each rule in X:

Definition 12 (Control matrix) Given the set of rules X, we define $\mathbf{F_{u_c}} \in \mathbb{B}^{|X|\times|X|}$ as the control matrix that is the matrix $\mathbb{I}^{|X|\times|X|}$, i.e. the identity matrix of size $|X|$. This matrix allows to control the activation of each rule in X as we will show later.

All the matrices defined above can be denoted using the following compact notation. We define the *precondition matrix* $\mathbf{F}$ as follows:

$$\mathbf{F} = [\mathbf{F_u} \quad \mathbf{F_t} \quad \mathbf{F_{u_c}} \quad \mathbf{F_y}], \tag{1}$$

where $\mathbf{F_y} \in \mathbb{B}^{|X|\times|Y|}$ is a null matrix introduced for dimensional constraints. In the same way, we define the *postcondition matrix* $\mathbf{S}$ as follows:

$$\mathbf{S} = [(\mathbf{S_u})^{\mathrm{T}} \quad (\mathbf{S_t})^{\mathrm{T}} \quad (\mathbf{S_{u_c}})^{\mathrm{T}} \quad (\mathbf{S_y})^{\mathrm{T}}]^{\mathrm{T}}. \tag{2}$$

where $\mathbf{S_u} \in \mathbb{B}^{|U|\times|X|}$ and $\mathbf{S_{u_c}} \in \mathbb{B}^{|X|\times|X|}$ are null matrices introduced for dimensional constraints and the superscript T indicates the operation of transposition.

Given q missions in W, we define the *global skill-task matrix* $\mathbf{\Sigma_t} \in \mathbb{B}^{|R|\times|T|}$ and the *global skill-rule matrix* $\mathbf{\Sigma_x} \in \mathbb{B}^{|R|\times|X|}$ as:

$$\mathbf{\Sigma_t} = [\ \mathbf{\Sigma_t}^{w_1} \quad \mathbf{\Sigma_t}^{w_2} \quad \ldots \quad \mathbf{\Sigma_t}^{w_q}\ ],$$

$$\mathbf{\Sigma_x} = [\ \mathbf{\Sigma_x}^{w_1} \quad \mathbf{\Sigma_x}^{w_2} \quad \ldots \quad \mathbf{\Sigma_x}^{w_q}\ ].$$

Note that these last two matrices are defined differently from the previous ones, since the number of rows of $\mathbf{\Sigma_t}$ and $\mathbf{\Sigma_x}$ is equal to the number of global skills in R.

Furthermore, a set of global vectors is introduced. Let us define the *global rule vector* $\mathbf{x}(\tau) \in \mathbb{B}^{|X|}$ as the vector which represents the state of all rules belonging to all missions in W, by stacking together the q rule vectors as follows:

$$\mathbf{x}(\tau) = [\ (\mathbf{x}^{w_1})^{\mathrm{T}}(\tau) \quad (\mathbf{x}^{w_2})^{\mathrm{T}}(\tau) \quad \ldots \quad (\mathbf{x}^{w_q})^{\mathrm{T}}(\tau)\ ]^{\mathrm{T}}. \tag{3}$$

Similarly, we define the following global vectors: the *global task vector* $\mathbf{t}(\tau) \in \mathbb{B}^{|T|}$, the *global input vector* $\mathbf{u}(\tau) \in \mathbb{B}^{|U|}$, and the *global output vector* $\mathbf{y}(\tau) \in \mathbb{B}^{|Y|}$.

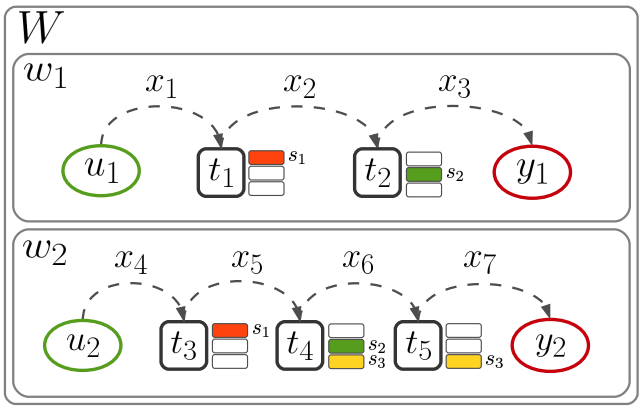


Fig. 2 An illustrative example: the mission set W composed of two missions $\{w_1, w_2\}$, the tasks set $T = \{t1, t2, t3, t4, t5\}$, and the all skills required by all tasks $R = \{s1, s2, s3\}$.

An Illustrative Example - Part I

In order to highlight the enhancements of the proposed Centralized Mission Model (CMM) for heterogeneous networks, we provide the following illustrative example. A schematic representation of two missions, w_1 and w_2, is shown in Fig. 2.

Following the composition rules presented above and deriving the matrices of the missions w_1 and w_2 from the diagram, the following sets of global matrices are obtained. The *preconditions matrix* $\mathbf{F}$ is obtained by combining, as in the eq. (1), the following matrices:

$$\mathbf{F_u} = \begin{bmatrix} \mathbf{F_u}^{w_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{F_u}^{w_2} \end{bmatrix} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{array} \begin{array}{c} \begin{array}{cc} u_1 & u_2 \end{array} \\ \begin{bmatrix} 1 & \\ 0 & 0 \\ 0 & \\ & 1 \\ & 0 \\ 0 & 0 \\ & 0 \end{bmatrix} \end{array},$$

$$\mathbf{F_t} = \begin{bmatrix} \mathbf{F_t}^{w_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{F_t}^{w_2} \end{bmatrix} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{array} \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{bmatrix} 0 & 0 & & & \\ 1 & 0 & & 0 & \\ 0 & 1 & & & \\ & & 0 & 0 & 0 \\ & & 1 & 0 & 0 \\ 0 & & 0 & 1 & 0 \\ & & 0 & 0 & 1 \end{bmatrix} \end{array},$$

$$\mathbf{F_{u_c}} = \begin{bmatrix} \mathbb{I}^{3\times3} & \mathbf{0} \\ \mathbf{0} & \mathbb{I}^{4\times4} \end{bmatrix}, \quad \mathbf{F_y} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}.$$

Similarly, the *postconditions matrix* $\mathbf{S}$ is obtained by combining, as in the eq. (2), the following matrices:

$$\mathbf{S_t} = \begin{bmatrix} \mathbf{S_t}^{w_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{S_t}^{w_2} \end{bmatrix} = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & & & & \\ 0 & 1 & 0 & & 0 & & \\ & & & 1 & 0 & 0 & 0 \\ & 0 & & 0 & 1 & 0 & 0 \\ & & & 0 & 0 & 1 & 0 \end{bmatrix} \end{array},$$

Algorithm 1 Global matrices partition procedure

Input: $\mathbf{F}_*, \mathbf{S}_*, \boldsymbol{\varphi}^{[i]}$
Output: $[\mathbf{F}_*{}^{[i]}, \mathbf{S}_*{}^{[i]}]$

for $j \leftarrow 1 : |\boldsymbol{\varphi}^{[i]}|$ **do**
  if $\boldsymbol{\varphi}^{[i]}_{(j)} = 1$ **then**
    $\mathbf{F}_*{}^{[i]} \leftarrow \mathbf{F}_{*(j,h)} \quad \forall h \in \{1, \dots, |\mathbf{F}_*|^c\}$
    $\mathbf{S}_*{}^{[i]} \leftarrow \mathbf{S}_{*(h,j)} \quad \forall h \in \{1, \dots, |\mathbf{S}_*|^r\}$
  end if
end for
for $k \leftarrow 1 : |\mathbf{F}_*{}^{[i]}|^c$ **do**
  if $(\mathbf{F}_*{}^{[i]}_{(j,k)} = 0$ **and** $\mathbf{S}_*{}^{[i]}_{(k,j)} = 0) \; \forall j \in \{1, \dots, |\boldsymbol{\varphi}^{[i]}|\}$ **then**
    $\mathbf{F}_*{}^{[i]} \leftarrow \texttt{delc}(\mathbf{F}_*{}^{[i]}, k)$
    $\mathbf{S}_*{}^{[i]} \leftarrow \texttt{delr}(\mathbf{S}_*{}^{[i]}, k)$
  end if
end for

$$\mathbf{S_y} = \begin{bmatrix} \mathbf{S_y}^{w_1}\,\mathbf{0} \\ \mathbf{0}\,\mathbf{S_y}^{w_2} \end{bmatrix} = \begin{matrix} y_1 \\ y_2 \end{matrix} \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ 0 & 0 & 1 & & & \mathbf{0} & \\ & \mathbf{0} & & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{S_u} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \quad \mathbf{S_{u_c}} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}.$$

The global matrix which describes the heterogeneity of the tasks is defined as

$$\mathbf{\Sigma_t} = [\mathbf{\Sigma_t}^{w_1}\ \mathbf{\Sigma_t}^{w_2}] = \begin{matrix} s_1 \\ s_2 \\ s_3 \end{matrix} \left[\begin{array}{cc|ccc} t_1 & t_2 & t_3 & t_4 & t_5 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{array}\right],$$

and the matrix which describes the heterogeneity of the rules as $\mathbf{\Sigma_x} = [\mathbf{\Sigma_x}^{w_1}\ \mathbf{\Sigma_x}^{w_2}]$.

Moreover, the global vectors associated to missions w_1 and w_2 are defined as: $\mathbf{u}(\tau) \in \mathbb{B}^2$, $\mathbf{t}(\tau) \in \mathbb{B}^5$, $\mathbf{y}(\tau) \in \mathbb{B}^2$, and $\mathbf{x}(\tau) \in \mathbb{B}^7$. Furthermore, at the initial conditions ($\tau = 0$) all these vectors have all entries equal to zero.

3.2 Decentralized Mission Model

In this subsection, we illustrate how to decompose the proposed CMM in order to obtain a decentralized set of Local Mission Models (LMMs) suitable for a distributed implementation.

Given the network of robots V, we introduce the following vector to describe the relationship between the generic agent v_i and the rules in X which define the missions in W.

Definition 13 (Rule-enable vector.) Let $\boldsymbol{\varphi}^{[i]} \in \mathbb{B}^{|X|}$ be the boolean column vector indicating which rules in X the agent v_i is able to fire. The element $\boldsymbol{\varphi}^{[i]}_{(j)}$ is set to "1" if the j-th rules can be fired by v_i.

3.2.1 Local Matrices

Given the rule-enable local vector $\boldsymbol{\varphi}^{[i]}$ is now possible to partition the set of missions in order to obtain only the tasks that the agent $v_i \in V$ is able to perform. Thus, each pair of matrices of the global model, denoted by $(\mathbf{F}_*, \mathbf{S}_*)$, where $*$ can be $\mathbf{u}$, $\mathbf{t}$, $\mathbf{u_c}$ or $\mathbf{y}$, is decomposed by each single agent v_i.to obtain the local pair of matrices, denoted by $(\mathbf{F}_*^{[\mathbf{i}]}, \mathbf{S}_*^{[\mathbf{i}]})$ to describe only tasks, control inputs, and input and output events related to the agent v_i. In other words, each agent removes from the matrices all the information related to skills that it does not own. This procedure is described in Algorithm 1, assuming that the operators $|\cdot|^c$ and $|\cdot|^r$ return the number of columns and rows of the evaluated matrix, respectively. Moreover, functions $\texttt{delc}(\mathbf{M}, j)$ and $\texttt{delr}(\mathbf{M}, j)$ erase both the j-th column and the j-th row of the matrix $\mathbf{M}$.

Thus, the following local matrices of the missions model are obtained for each agent $v_i \in V$:

Definition 14 (Local input matrix) Let us define the local input matrix $\mathbf{F_u}^{[i]}$ as a sub-matrix of $\mathbf{F_u}$ obtained erasing the j-th row of $\mathbf{F_u}$ where $\boldsymbol{\varphi}^{[i]}_{(j)} = 0$.

Definition 15 (Local task-end matrix) Let us define the local task-end matrix $\mathbf{F_t}^{[i]}$ obtained erasing the j-th row of $\mathbf{F_t}$ where $\boldsymbol{\varphi}^{[i]}_{(j)} = 0$.

Definition 16 (Local task-start matrix) Let us define the local task-start matrix $\mathbf{S_t}^{[i]}$ obtained erasing the j-th row of $\mathbf{S_t}$ where $\boldsymbol{\varphi}^{[i]}_{(j)} = 0$.

Definition 17 (Local output matrix) Let us define the local output matrix $\mathbf{S_y}^{[i]}$ as a sub-matrix of $\mathbf{S_y}$ obtained erasing the j-th column of $\mathbf{S_y}$ where $\boldsymbol{\varphi}^{[i]}_{(j)} = 0$.

Moreover, we also obtain the local control matrix $\mathbf{F_{u_c}}^{[i]}$ and $\mathbf{F_y}^{[i]}$, $\mathbf{S_u}^{[i]}$, and $\mathbf{S_{u_c}}^{[i]}$ which are null matrices introduced for dimensional constraints as in the global model (eq. 1-2).

3.2.2 Local Vectors

To complete the definition of the local mission model of the agent $v_i \in V$ also the global vectors have to be partitioned. The *local rule vector* $\mathbf{x}^{[i]}(\tau)$ is obtained by resizing the global rule vector $\mathbf{x}(\tau)$ according to the dimension of the matrices $\mathbf{S}^{[i]}$ and $\mathbf{F}^{[i]}$. Also, the set of local vectors has to be resized according to the local matrices dimension. Moreover, as for the matrix set a vector to control the activation of rule has to be defined:

Definition 18 (Control vector) Let $\mathbf{u_c}^{[i]}(\tau) \in \mathbb{B}^{|X|}$ be the boolean column vector which inhibits or allows

the firing of a rule in X. The element $\mathbf{u_c}^{[i]}_{(j)}(\tau)$ is set to "1" when the firing of the j-th rule, at the discrete event iteration (τ), is allowed.

In particular, local vectors can be put together in a novel local vector, called *local marking vector* which can be represented as follows at the discrete event iteration τ:

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

where $\mathbf{u}^{[i]}(\tau)$ is the *local input vector* (whose length is equal to the number of columns of $\mathbf{F_u}^{[i]}$), $\mathbf{t}^{[i]}(\tau)$ is the *local task vector* (whose length is equal to the number of columns of $\mathbf{F_t}^{[i]}$), $\mathbf{u_c}^{[i]}(\tau)$ is the *local control vector* (whose length is equal to the number of columns of $\mathbf{F_{u_c}}^{[i]}$) and $\mathbf{y}^{[i]}(\tau)$ is the *local output vector* (whose length is equal to the number of rows of $\mathbf{S_y}^{[i]}$).

3.2.3 Task Duration

It is important to note that the local mission model described above does not take into consideration the tasks execution time. Thus, we need to introduce the concept of *duration of a task*. By denoting with τ the discrete event iteration, let us split the local marking vector $\mathbf{m}^{[i]}(\tau)$ into two vectors of the same dimensions in order to take into account the duration of tasks:

$$\mathbf{m}^{[i]}(\tau) = {}^{\bullet}\mathbf{m}^{[i]}(\tau) + \mathbf{m}^{\bullet\,[i]}(\tau),$$

where ${}^{\bullet}\mathbf{m}^{[i]}(\tau)$ is the *precondition part* of the marking vector, which can be trigger the activation of one or more rules, defined as:

$${}^{\bullet}\mathbf{m}^{[i]}(\tau) = [\mathbf{u}^{[i]}(\tau)^{\mathrm{T}}\ {}^{\bullet}\mathbf{t}^{[i]}(\tau)^{\mathrm{T}}\ \mathbf{u_c}^{[i]}(\tau)^{\mathrm{T}}\ \mathbf{y_0}^{[i]}(\tau)^{\mathrm{T}}]^{\mathrm{T}}, \quad (4)$$

and $\mathbf{m}^{\bullet\,[i]}(\tau)$ is the *postcondition part* of the marking vector, which represents the consequence of the activation of one or more rules, defined as:

$$\mathbf{m}^{\bullet\,[i]}(\tau) = [\mathbf{u_0}^{[i]}(\tau)^{\mathrm{T}}\ \mathbf{t}^{\bullet\,[i]}(\tau)^{\mathrm{T}}\ \mathbf{u_{c_0}}^{[i]}(\tau)^{\mathrm{T}}\ \mathbf{y}^{[i]}(\tau)^{\mathrm{T}}]^{\mathrm{T}}. \quad (5)$$

The vectors $\mathbf{y_0}^{[i]}(\tau)$, $\mathbf{u_0}^{[i]}(\tau)$, and $\mathbf{u_{c_0}}^{[i]}(\tau)$ are suitable null vectors and ${}^{\bullet}\mathbf{t}^{[i]}(\tau)$ and $\mathbf{t}^{\bullet\,[i]}(\tau)$ are the vectors of tasks completed and tasks to be started, respectively.

In order to maintain the global identification of tasks and events and to allow agents to exchange this information, a function to map the local indexes into the global ones must be introduced for both ${}^{\bullet}\mathbf{m}^{[i]}(\tau)$ and $\mathbf{m}^{\bullet\,[i]}(\tau)$ vectors. To this end, we define the following vector.

Definition 19 (Marking Index vector) Let us define $\boldsymbol{\mu}^{[i]}(\tau) \in \{1, \ldots, |\mathbf{m}(\tau)|\}^{|\mathbf{m}^{[i]}(\tau)|}$ as the *marking index vector*, where the expression $\boldsymbol{\mu}^{[i]}_{(j)}(\tau) = k$ means that the value in the position j in the local vector ${}^{\bullet}\mathbf{m}^{[i]}(\tau)$ (or equivalently $\mathbf{m}^{\bullet\,[i]}(\tau)$) has to be mapped to the k-th position of the global vector ${}^{\bullet}\mathbf{m}(\tau)$ (or equivalently $\mathbf{m}^{\bullet}(\tau)$). Where ${}^{\bullet}\mathbf{m}(\tau)$ is the precondition part of the global marking vector and $\mathbf{m}^{\bullet}(\tau)$ is the postcondition part of the global marking vector.

3.3 Analysis of The Distribution of the Computational Load

Given the decomposition mechanism described above, it is possible to devise a metric to study the relationship between the set of missions and the network of agents in charge to execute them in term of computational load. We have considered the global and local precondition and postcondition matrices, ($\mathbf{F}$, $\mathbf{S}$) and ($\mathbf{F}^{[i]}$, $\mathbf{S}^{[i]}$), respectively. The number of elements the matrix $\mathbf{F}$ (which is the same of matrix $\mathbf{S}$), given by $\alpha = |\mathbf{x}(\tau)| \cdot |\mathbf{m}(\tau)|$, can give a metric to quantify the computational load of each LMM in the decentralized implementation, because, as we will see in Section 4, the Centralized Task Execution (CTE) algorithm performs the main equations using the $\mathbf{F}$ and $\mathbf{S}$ matrices. In the same way, for each agent v_i we can define the local metric for matrix $\mathbf{F}^{[i]}$ (or equivalently $\mathbf{S}^{[i]}$) such as $\alpha_{v_i} = |\mathbf{x}^{[i]}(\tau)| \times |\mathbf{m}^{[i]}(\tau)|$. Thus we can define the *distribution degree* of the computational load for the agent v_i as:

$$\delta_{v_i} = \frac{\alpha_{v_i}}{\alpha}. \quad (6)$$

Moreover we can evaluate the *mean distribution degree* of the computational load across the network as:

$$\langle \delta \rangle = \frac{1}{n} \sum_{i=1}^{n} \frac{\alpha_{v_i}}{\alpha}. \quad (7)$$

Obviously, this metrics depends on the set of skills of each robot, thus the more the network is homogenous, the closer the value of $\langle \delta \rangle$ is to 1. Conversely, the more the network is heterogeneous, the closer the value of $\langle \delta \rangle$ is to $1/\alpha$. This metric will be used to illustrate the benefit of the distribution of the proposed approach in the implementation of a real-world scenario in Section 5.

An Illustrative Example - Part II

In this subsection we describe the decomposition of a mission set for one agent. Consider the mission set W

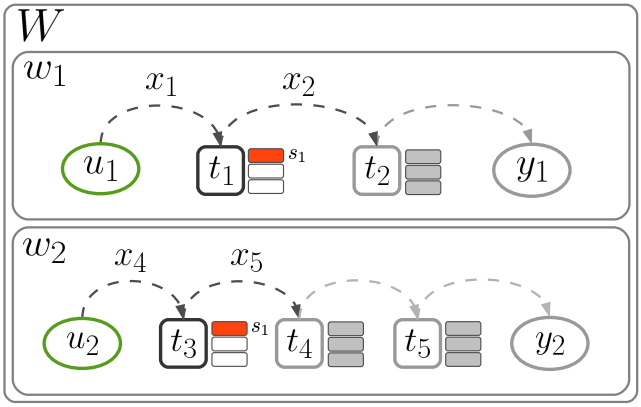


Fig. 3 Partition of missions: the same mission set W of Fig. 2 where only rules, tasks, and skills belonging to agent $v_i \in V$, with set skill $S_{v_i} = \{s1, s2\}$, are highlighted.

as in the first part of this example, depicted in Fig. 2. We consider, the two missions $\{w_1, w_2\}$, with a global set of skills $S = \{s_1, s_2, s_3\}$, and a network V in which the agent v_1 owns the set of skills $S_{v_1} = \{s_1\}$. Due to the skill set S_{v_1}, not all the tasks can be executed by the agent v_1, as reported in Fig. 3. This could represent a scenario where there are two tasks, the first requires the usage of a short range camera while the second requires the combined use of a short range camera and a laser scanner. Clearly, a robotic agent equipped only with a short range camera cannot execute the second task by itself. Thus, given the precondition matrix $\mathbf{F}$, the postcondition matrix $\mathbf{S}$, and using the partition procedure described in Algorithm 1, we can obtain the set of local matrices. For instance, consider the mission w_2 and associated matrix ${\mathbf{F_t}}^{w_2}$, we obtain the local matrix ${\mathbf{F_t}}^{[1]^{w_2}}$ for the agent v_1 as follows:

$$
{\mathbf{F_t}}^{w_2} = \begin{array}{c} \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{array} \begin{array}{c} \begin{array}{ccc} t_3 & t_4 & t_5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \Rightarrow \quad {\mathbf{F_t}}^{[1]^{w_2}} = \begin{array}{c} \\ x_3^{[1]} \\ x_4^{[1]} \end{array} \begin{array}{c} \begin{array}{c} t_2^{[1]} \end{array} \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{array},
$$

and, at the same time, we also obtain the matrix ${\mathbf{S_t}}^{[1]^{w_2}}$,

$$
{\mathbf{S_t}}^{w_2} = \begin{array}{c} \\ t_3 \\ t_4 \\ t_5 \\ t_6 \end{array} \begin{array}{c} \begin{array}{ccc} x_3 & x_4 & x_5 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{array} \Rightarrow \quad {\mathbf{S_t}}^{[1]^{w_2}} = t_2^{[1]} \begin{array}{c} \begin{array}{cc} x_3^{[1]} & x_4^{[1]} \end{array} \\ \begin{bmatrix} 1 & 0 \end{bmatrix} \end{array}.
$$

We can notice that, for mission w_2 , tasks t_4 and t_5 cannot be executed by agent v_1. Thus the matrices ${\mathbf{F_t}}^{[1]^{w_2}}$ and ${\mathbf{S_t}}^{[1]^{w_2}}$ are related only to the task t_3, which is indicated, locally for agent v_i, with the label $t_2^{[1]}$. Similarly for mission w_1 only the task t_1 can be executed, thus this is indicated locally as $t_1^{[1]}$. The correspondence between global and local indices is represented in the marking index vector $\boldsymbol{\mu}^{[1]}(\tau)$, which is defined after the decomposition procedure, when matrices $\mathbf{F}^{[1]}$, $\mathbf{S}^{[1]}$, and the precondition ${}^{\bullet}\mathbf{m}^{[1]}(\tau)$ and postcondition $\mathbf{m}^{\bullet\,[1]}(\tau)$ marking vectors, associated to missions, are properly obtained. In this example, for robot v_1, the distribution degree (equation 6) is:

$$
\delta_{v_1} = \frac{\alpha_{v_1}}{\alpha} = \frac{4 \cdot 8}{7 \cdot 16} = \frac{32}{112} = 0.28.
$$

This means that by means of the decentralization procedure the computational load of the agent v_1 is approximatively 0.3% of the global (centralized) computational load.

4 Dynamic Task Planning Control

In this section, given the LMM, introduced in the previous section, we develop a decentralized dynamic task planning solution to control the execution of tasks, matching the precedence constraints, and to online assign the tasks to the robots in the network. Furthermore, we prove that the evolution of the consensus-based distributed algorithm is equivalent to the evolution of the centralized one, under the assumption of equivalence of the solution of the centralized and distributed task assignment algorithm. To this end, first we introduce the centralized algorithm for the control of task execution, then we propose the decentralized dynamic task planning algorithm.

4.1 Centralized Dynamic Task Planning

In this subsection, first some fundamental concepts of the centralized Matrix-based Discrete Event control Framework (MDEF) proposed in [24], are reviewed. This formalism is used as the basis of the algorithm which ensures the correct execution of tasks. Then, the main equations of the centralized task planning algorithm are described.

4.1.1 Matrix-based Discrete-Event control Framework

The control of task execution, as a part of the dynamic task planning framework, can be viewed as a dynamical system where inputs are represented by events that have been triggered and tasks that have been completed, while outputs are represented by tasks that have to be executed and user outputs. In order to describe how the dynamical system evolves and outputs are obtained, let us now introduce some useful concepts.

Definition 20 (Logical State Equation) The evolution of the internal state of this control system, i.e.,

Algorithm 2 Centralized Task Execution at iteration τ

Input: $[\mathbf{u}(\tau), {}^{\bullet}\mathbf{t}(\tau), \mathbf{z}(\tau)]$
Output: $[\mathbf{t}^{\bullet}(\tau), \mathbf{y}(\tau)]$
1: $\mathbf{u_c}(\tau) \leftarrow ((\mathbf{S_t})^{\mathrm{T}} \odot h(\mathbf{z}(\tau) > 0))$
2: ${}^{\bullet}\mathbf{m}(\tau) \leftarrow (\mathbf{u}(\tau), \mathbf{u_c}(\tau), {}^{\bullet}\mathbf{t}(\tau))$
3: $\mathbf{x}(\tau) \leftarrow \neg(\mathbf{F} \odot \neg\, {}^{\bullet}\mathbf{m}(\tau))$
4: UPDATE$({}^{\bullet}\mathbf{m}(\tau), \mathbf{m}^{\bullet}(\tau))$
5: $\mathbf{m}^{\bullet}(\tau) \rightarrow (\mathbf{t}^{\bullet}(\tau), \mathbf{y}(\tau))$

how logical values of rules are calculated, at iteration τ, is obtained by means of the following *logical state equation*:

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

where $\neg$ and $\odot$ are the unitary operation `complement` and logical matrix product, respectively (see Appendix for further details).

In [24], an important relationship between the Matrix-based Discrete Event control Framework (MDEF) and the Petri Nets (PNs) has been proven. Indeed, the same holds for the proposed dynamic task planning framework for heterogeneous robots.

Proposition 1 (MDEF - PNs Equivalence) *Let us define the set of transitions $\mathcal{X}$ containing elements of vector $\mathbf{x}$, and the set of places $\mathcal{A}$ containing elements of vectors $\mathbf{u}$, $\mathbf{t}$, $\mathbf{u_c}$, and $\mathbf{y}$. Thus the tuple $(\mathcal{X}, \mathcal{A}, \mathbf{F}^T, \mathbf{S})$ is a PN [19] equivalent to the centralized model described in Section 3.1, where matrices $\mathbf{F}$ and $\mathbf{S}$ are the global preconditions matrix and the global postconditions matrix, respectively. Moreover, the marking transition of the equivalent PN can be obtained using the PN transition equation:*

$$\mathbf{m}(\tau + 1) = \mathbf{m}(\tau) + [\mathbf{S}^T - \mathbf{F}]^T \mathbf{x}(\tau). \tag{9}$$

where $\mathbf{m}(\tau)$ is the global marking.

A complete dynamical description of the system can be then achieved by combining the logical state equation (8) with the PN transition equation (9), where the vector $\mathbf{x}(\tau)$ is exactly the global rule logical vector defined in equation (3).

4.1.2 Centralized Dynamic Task Planning

At this point, according to Definition 20 and the CMM representation presented in the previous section, we present the Centralized Dynamic Task Planning (CDTP) algorithm. This algorithm requires a Centralized Task Assignment (CTA) algorithm to obtain the preliminary data for the CTE algorithm. Note that, the description of the particular implementation of the CTA algorithm goes beyond the scope of this paper. Therefore we describe only the CTE algorithm assuming the following requirements for the CTA algorithm: if an event occurs, i.e. a new mission start event or a completion of a task, the CTA algorithm is triggered. The produced assignment is stored in the vector $\mathbf{z}(\tau) \in V^{|T|}$, where if the generic task t_j is assigned to robot v_i, at the discrete event iteration τ, we have $\mathbf{z}(\tau)_{(j)} = i$. This information will be used to trigger the execution of the assigned tasks as follows.

As a first step of the CTE algorithm, described in Algorithm 2, the control vector $\mathbf{u_c}(\tau)$ is updated on the basis of the CTA (line 1 of Algorithm 2), where the function $h(\cdot)$ produced a vector of the same length of $\mathbf{z}$, where the element is 1 if the inequality is verified and 0 otherwise. Thus the marking vector ${}^{\bullet}\mathbf{m}(\tau)$ is updated (line 2 of Algorithm 2). The activation of rules is calculated through the *logical state equation* in (8), by using the updated vector ${}^{\bullet}\mathbf{m}(\tau)$ (line 3 of Algorithm 2). After the rule update is executed, which indicates which task can start and if a mission is accomplished, the precondition part and the postcondition part of the marking vector are updated (line 4 of Algorithm 2). The marking vector ${}^{\bullet}\mathbf{m}(\tau)$ is updated as:

$${}^{\bullet}\mathbf{m}(\tau + 1) = {}^{\bullet}\mathbf{m}(\tau) - \mathbf{F}^T \mathbf{x}(\tau). \tag{10}$$

Note that, this equation updates the elements of ${}^{\bullet}\mathbf{m}(\tau)$ vector if a rule is fired, that is to say that if an event occurs. Postconditions are determined by updating the $\mathbf{m}^{\bullet}(\tau)$ vector as follows:

$$\mathbf{m}^{\bullet}(\tau + 1) = \mathbf{m}^{\bullet}(\tau) + \mathbf{S}\,\mathbf{x}(\tau). \tag{11}$$

Note that, this equation updates the elements of $\mathbf{m}^{\bullet}(\tau)$ vector to activate tasks or to produce mission completion events. If a rule is fired the relative position of the vector is updated, $\mathbf{m}^{\bullet}_{(j)}(\tau) = 1$. At the end, new tasks are triggered $\mathbf{t}^{\bullet}(\tau)$ or missions are completed $\mathbf{y}(\tau)$ according to the information in $\mathbf{m}^{\bullet}(\tau)$ (line 5 of Algorithm 2).

4.2 Decentralized Dynamic Task Planning

In this subsection we present the Decentralized Dynamic Task Planning (DDTP) algorithm. The proposed DDTP algorithm is composed by two parts, running at different time scale. The first part is the Decentralized Task Execution (DTE) control, which guarantees that the precedence constraints among tasks are satisfied. The second part is the Decentralized Task Assignment (DTA) algorithm, which ensures the matching of tasks and robots in order to maximize a given

Algorithm 3 Decentralized Task Execution at iteration τ

Input: $[\mathbf{u}(\tau)^{[i]}, {}^{\bullet}\mathbf{t}(\tau)^{[i]}, \mathbf{z}(\tau)^{[i]}]$
Output: $[\mathbf{t}^{\bullet}(\tau)^{[i]}, \mathbf{y}(\tau)^{[i]}]$
1: $\mathbf{u_c}^{[i]}(\tau) \leftarrow ((\mathbf{S_t}^{[i]})^{\mathrm{T}} \odot h(\mathbf{z}^{[i]}(\tau) = i))$
2: ${}^{\bullet}\mathbf{m}^{[i]}(\tau) \leftarrow (\mathbf{u}^{[i]}(\tau), \mathbf{u_c}^{[i]}(\tau), {}^{\bullet}\mathbf{t}^{[i]}(\tau))$
3: ${}^{\bullet}\widehat{\mathbf{m}}^{[i]}(\tau) \leftarrow$ Local2Global$({}^{\bullet}\mathbf{m}^{[i]}(\tau), \boldsymbol{\mu}^{[i]})$
4: **for** $k \leftarrow 1 : \mathcal{D}(\mathcal{G}(\tau))$ **do**
5: $\quad {}^{\bullet}\widehat{\mathbf{m}}^{[i]}(k+1) \leftarrow \oplus_{j \in \mathcal{N}_i} {}^{\bullet}\widehat{\mathbf{m}}^{[j]}(k)$
6: **end for**
7: ${}^{\bullet}\widetilde{\mathbf{m}}^{[i]}(\tau) \leftarrow$ Global2Local$({}^{\bullet}\widehat{\mathbf{m}}^{[i]}(\tau), \boldsymbol{\mu}^{[i]})$
8: $\mathbf{x}^{[i]}(\tau) \leftarrow \neg(\mathbf{F}^{[i]} \odot \neg\, {}^{\bullet}\mathbf{m}^{[i]}(\tau))$
9: Update$({}^{\bullet}\mathbf{m}^{[i]}(\tau), \mathbf{m}^{\bullet\,[i]}(\tau))$
10: $\mathbf{m}^{\bullet\,[i]}(\tau) \rightarrow (\mathbf{t}^{\bullet\,[i]}(\tau), \mathbf{y}^{[i]}(\tau))$

score. Note that, the mission models decomposition introduced in Section 3.2 is instrumental for the development of the DDTP algorithm described in the following. Indeed, this emphasizes how important the concept of skills is to develop a decentralized framework suitable for handling the RN heterogeneity.

4.2.1 Decentralized Task Execution

The DTE control algorithm starts when an event is sensed over the network by the agent $v_i \in V$, e.g. an event triggered by a sensor of the completion of a task. The Algorithm 3 shows the evolution of DTE at the discrete event iteration τ.

If the agent v_i senses an external event or completes a task, then it will update its local vectors $\mathbf{u}^{[i]}(\tau)$ or ${}^{\bullet}\mathbf{t}^{[i]}(\tau)$, respectively. Moreover, each agent v_i updates the information related to the local control vector $\mathbf{u_c}^{[i]}(\tau)$ by using the result of the DTA, through the vector $\mathbf{z}^{[i]}(\tau)$ that indicates what task is assigned to agent v_i at the discrete event iteration τ (line 1 of Algorithm 3). The role of the task assignment algorithm and the vector $\mathbf{z}^{[i]}(\tau)$ will be clarified in the following. Thus, the vector ${}^{\bullet}\mathbf{m}^{[i]}(\tau)$ is updated accordingly (line 2). The new value of the vector ${}^{\bullet}\mathbf{m}^{[i]}(\tau)$ is mapped into ${}^{\bullet}\widehat{\mathbf{m}}^{[i]}(\tau)$, the local copy of the global preconditions marking vector, using the *marking index vector* $\boldsymbol{\mu}^{[i]}$ (line 3). Thus, the agent v_i starts a synchronization mechanism to reach an agreement on ${}^{\bullet}\widehat{\mathbf{m}}^{[i]}(\tau)$. This is achieved by exploiting the Abstract Consensus described in [9, Theorem 1] (Algorithm 3, line 5). It is worth noticing that the agreement procedure is performed in a different time scale k and the convergence is reached in exactly $\mathcal{D}(\mathcal{G}(\tau))$ steps, according to the Abstract Consensus convergence property.

Then each agent v_i updates the vector ${}^{\bullet}\widetilde{\mathbf{m}}^{[i]}(\tau)$, using the result of the agreement protocol through the *marking index vector* $\boldsymbol{\mu}^{[i]}$. At this step all the agents in V share the same information about internal and external events (line 7). Thus, each agent v_i computes the rules to be activated $\mathbf{x}^{[i]}(\tau)$, according to ${}^{\bullet}\mathbf{m}^{[i]}(\tau)$, using the local *logic state equation* (line 8 of Algorithm 3). After the rule update, each agent v_i computes the local precondition part ${}^{\bullet}\mathbf{m}^{[i]}(\tau)$ and the local postcondition part $\mathbf{m}^{\bullet}(\tau)$ of the marking vector as follows (line 9):

$$ {}^{\bullet}\mathbf{m}^{[i]}(\tau+1) = {}^{\bullet}\mathbf{m}^{[i]}(\tau) - \mathbf{F}^{[i]^{\mathrm{T}}}\,\mathbf{x}^{[i]}(\tau), \tag{12} $$

$$ \mathbf{m}^{\bullet\,[i]}(\tau+1) = \mathbf{m}^{\bullet\,[i]}(\tau) + \mathbf{S}^{[i]}\,\mathbf{x}^{[i]}(\tau). \tag{13} $$

Thus, agent v_i can start the execution of new tasks $\mathbf{t}^{\bullet\,[i]}(\tau)$ or it sends a message about the completion of one or more missions $\mathbf{y}^{[i]}(\tau)$, according to the information in $\mathbf{m}^{\bullet\,[i]}(\tau)$ (line 10).

4.2.2 Theoretical Properties of the DTE

In this subsection we show the convergence property of the DTE control algorithm and we prove that, given the set of local controllers obtained by the decomposition performed by Algorithm 1 and assuming that the results of the centralized and decentralized task assignment algorithm are equivalent, the Algorithm 3 is equivalent of the Centralized Task Execution (Algorithm 2) described in Section 4.A.

Since the DTE algorithm (Algorithm 3) can be viewed as a sequence of updates, its convergence depends only upon the agreement procedure. Thus, we can provide the following convergence condition.

Lemma 1 (Convergence of DTE) *Given the network $V = \{v_1, \ldots, v_n\}$ of heterogeneous agents when each agent $v_i \in V$ performs the Algorithm 3, at discrete event iteration τ, and assuming the communication graph $\mathcal{G}(\tau)$ connected, the Algorithm 3 ends in $\mathcal{O}(\mathcal{D}(\mathcal{G}(\tau)))$ steps.*

Proof The proof of this lemma comes from the fact that Algorithm 3 is a sequence of statements along with the agreement procedure in lines 4–6. Thus, according to [9, Theorem 1], if the graph $\mathcal{G}(\tau)$ is connected, the procedure converges in $\bar{k} \leq \mathcal{D}(\mathcal{G}(\tau))$ steps. As a consequence Algorithm 3 ends in $\mathcal{O}(\mathcal{D}(\mathcal{G}(\tau)))$ steps.

In order to prove the equivalence between the distributed formulation and the centralized one, let us now introduce the concept of *consistency* in a set of local vectors $\{\mathbf{a}^{[i]}(\tau)\}$, $i = \{1, \ldots, n\}$ with respect to another vector $\mathbf{a}(\tau)$, as follows:

Definition 21 A set of local vectors $\{\mathbf{a}^{[i]}(\tau)\}$ with $i = \{1, \ldots, n\}$ is said to be *consistent*, with respect to another vector $\mathbf{a}(\tau)$, if the following holds:

$$ \mathbf{a}(\tau) = \widehat{\mathbf{a}}^{[1]}(\tau) \oplus \widehat{\mathbf{a}}^{[2]}(\tau) \oplus \ldots \oplus \widehat{\mathbf{a}}^{[n]}(\tau) \tag{14} $$

where $\widehat{\mathbf{a}}^{[i]}(\tau)$ is the global map of the local vector $\mathbf{a}^{[i]}(\tau)$ by means of the index vector $\boldsymbol{\mu}^{[i]}$ and $\oplus$ is the logical `or` operator.

In the following theorem the conditions under which the decentralized formulation is equivalent to the centralized one are given.

Theorem 1 *Given the network $V = \{v_1, \ldots, v_n\}$ of heterogeneous agents and assuming we have a decentralized and centralized task assignment algorithms capable to produce the same assignment, the decentralized formulation is equivalent to the centralized one if local initial conditions $\{{}^{\bullet}\mathbf{m}^{[i]}(0), \mathbf{m}^{\bullet\,[i]}(0)\}$, with $i = \{1, \ldots, n\}$ are consistent with the global ones $\{{}^{\bullet}\mathbf{m}(0), \mathbf{m}^{\bullet}(0)\}$.*

Proof In order to prove the theorem, it must be shown that the evolution of the eq. (12) and eq. (13) for the decentralized scenario is consistent with the evolution of eq. (10) and eq. (11) for the centralized scenario. To this end, by recalling that the matrices of the local models are obtained by removing rows and columns from the matrices of the centralized model, it is sufficient to show that the local variables ${}^{\bullet}\widetilde{\mathbf{m}}^{[i]}(\tau)$, $\mathbf{m}^{\bullet\,[i]}(\tau)$ are consistent with the global ones ${}^{\bullet}\mathbf{m}(\tau)$, $\mathbf{m}^{\bullet}(\tau)$. As a consequence, this would imply that the vectors $\{\mathbf{z}^{[i]}(\tau)\}$, with $i = \{1, \ldots, n\}$ are consistent with $\mathbf{z}(\tau)$ on the assumption of the equivalence of the task assignment. At this point, being all the variables affecting the evolution of the decentralized scenario consistent with the variables affecting the evolution of the centralized scenario, their evolution can not differ. Thus, the decentralized control of task execution can only be completely equivalent to the centralized one.

Let us now investigate the relationship between $\{{}^{\bullet}\widetilde{\mathbf{m}}^{[i]}(\tau)\}$, with $i = \{1, \ldots, n\}$ and ${}^{\bullet}\mathbf{m}(\tau)$. To this end, let us assume $\{{}^{\bullet}\mathbf{m}^{[i]}(0)\}$, with $i = \{1, \ldots, n\}$ to be consistent with ${}^{\bullet}\mathbf{m}(0)$, that is:

$$ {}^{\bullet}\mathbf{m}^{[i]}(0) =^{\bullet} \widehat{\mathbf{m}}^{[1]}(0) \oplus {}^{\bullet}\widehat{\mathbf{m}}^{[2]}(0) \oplus \ldots \oplus^{\bullet} \widehat{\mathbf{m}}^{[n]}(0) \quad (15) $$

with $\oplus$ the logical `or` operator. Now, let us assume a certain event to be triggered at time $\tau = 1$. According to the decentralized control of task execution algorithm, the agent v_i sensing this event first updates its vector ${}^{\bullet}\mathbf{m}^{[i]}(1)$ and then starts the abstract consensus over ${}^{\bullet}\widehat{\mathbf{m}}(1)$, obtained by means of the marking index vector $\boldsymbol{\mu}^{[i]}$. In a centralized scenario, the triggering event is simply registered into the vector ${}^{\bullet}\mathbf{m}(1)$. However, according to [9, Theorem 1], ${}^{\bullet}\mathbf{m}(1)$ also represents the steady state of the abstract consensus. Therefore, the local vectors $\{{}^{\bullet}\widetilde{\mathbf{m}}^{[i]}(\tau)\}$, with $i = \{1, \ldots, n\}$ and ${}^{\bullet}\mathbf{m}(\tau)$ are still consistent at time $\tau = 1$. The same reasoning can be iterated over time for any step τ.

Let us now investigate the relationship between $\{\widetilde{\mathbf{m}}^{\bullet\,[i]}(\tau)\}$, with $i = \{1, \ldots, n\}$ and $\mathbf{m}^{\bullet}(\tau)$. To this end, let us assume $\{\mathbf{m}^{\bullet\,[i]}(0)\}$, with $i = \{1, \ldots, n\}$ to be consistent with $\mathbf{m}^{\bullet}(0)$, that is:

$$ \mathbf{m}^{\bullet\,[i]}(0) = \widehat{\mathbf{m}}^{\bullet\,[1]}(0) \oplus \widehat{\mathbf{m}}^{\bullet\,[2]}(0) \oplus \ldots \oplus \widehat{\mathbf{m}}^{\bullet\,[n]}(0) \quad (16) $$

with $\widehat{\mathbf{m}}^{\bullet\,[i]}(0)$ the global version of the local vector $\mathbf{m}^{\bullet\,[i]}(0)$ obtained by means of the marking index vector $\boldsymbol{\mu}^{[i]}$. To prove the local vectors $\{\mathbf{m}^{\bullet\,[i]}(\tau)\}$, with $i = \{1, \ldots, n\}$ and $\mathbf{m}^{\bullet}(\tau)$ to be consistent for any step τ, it is sufficient to notice that their evolution is not related to external events but only to $\{{}^{\bullet}\widetilde{\mathbf{m}}^{[i]}(\tau)\}$, with $i = \{1, \ldots, n\}$ and ${}^{\bullet}\mathbf{m}(\tau)$ respectively. Therefore, if we assume the $\{\mathbf{m}^{\bullet\,[i]}(\tau)\}$, with $i = \{1, \ldots, n\}$ and $\mathbf{m}^{\bullet}(\tau)$ to be consistent at time $\tau = 0$, it is sufficient to guarantee their consistence over time for any τ. Note that the dependence on $\{\mathbf{z}^{[i]}(\tau)\}$, $i = 1, \ldots, n$ and $\mathbf{z}(\tau)$ is not considered on the basis of the task assignment equivalence, being their evolution related to the evolution of $\{{}^{\bullet}\widetilde{\mathbf{m}}^{[i]}(\tau)\}$, with $i = \{1, \ldots, n\}$ and ${}^{\bullet}\mathbf{m}(\tau)$ respectively, and thus following the same argument.

It may be useful to remark that Theorem 1 only guarantees that the decentralization of the framework does not directly determine a change in the performance of the RN, because assuming that a decentralized decision algorithm that makes the same choices of a centralized decision algorithm exists, the dynamics of the decentralized and centralized RNs would be the same. However, the existence of such an equivalent algorithm is not a necessary prerequisite for the decentralized framework to work properly. Rather, the theorem should be viewed as a means to ensure that if the dynamics of the decentralized RN is different from that of a centralized RN, this occurs only due to the (often unavoidable) differences between the decision algorithms.

4.2.3 Decentralized Task Assignment Requirements

In this subsection, we present how the task assignment algorithm is embedded in the DDTP framework. As described above, the DTE algorithm needs the support of a task assignment algorithm, the DTA. This algorithm is necessary to prevent situations of *assignment conflict*, i.e. the flow of execution of multiple missions can lead to situations where a single robot has to perform two or more tasks at the same time. Thus, the DTA algorithm has to satisfy these requirements:

Definition 22 (DTA Requirements) Given the task set $\bar{T} \subseteq T$, which contains the only $\bar{m}$ tasks which have to be executed at iteration τ, and set of n robots V, the objective of the DTA algorithm is to find a conflict-free

assignment of tasks in $\bar{T}$ to each agent $v_i \in V$, considering its skills S_{v_i}, that maximizes a global reward.

Differently from the CTA, where the assignment is instantaneous, in the DTA we need the assignment represented by the sequence of tasks $\mathbf{p}^{[i]} \in \bar{T}^{\lambda}$ called *path*, where λ is a predefined length, that represents the execution order of tasks assigned to v_i. Thus, at each discrete event iteration τ, we obtain a set of paths, $\{\mathbf{p}^{[1]}, \dots \mathbf{p}^{[n]}\}$, one for each agent. Using its sequence of tasks, each agent v_i can build the *local first-task-assignment* vector $\mathbf{z}^{[i]}(\tau) \in V^{|T|}$, used in the Algorithm 3, as follows: the element which corresponds to the first task of $\mathbf{p}^{[i]}$ is set to "1", i.e $\mathbf{z}^{[i]}_{(\mathbf{p}^{[i]}_{(1)})}(\tau) = i$, and it is erased from the path. Using this strategy, the information of the sequence of tasks that agent v_i will execute is stored in $\mathbf{p}^{[i]}$, while the current task, updated at iteration τ, is stored in $\mathbf{z}^{[i]}(\tau)$.

Different implementations of the DTA can be chosen, according to the particular scenario in which the generic objective function (minimize the paths of all robots, minimize the time to visit the all task locations, maximize the number of items collected, etc.) is considered.

Remark 2 (Deadlock-Free Execution) Generally, in systems of the control of task execution, deadlocks on shared resources can occur [18]. This situation may be prevented by using particular techniques which perform off-line analysis of the relationships between the resources (robots) and tasks that use them. The main reason behind the occurrence of a deadlock is that the execution flow of tasks and the assignment of robots to tasks are represented in the same static formalism. In our framework the assignment of tasks to robots is not given a priori, thus the problem of the assignment is solved online, both in the case of centralized and DDTP, guaranteeing the conflict-free assignment and a deadlock-free task execution.

5 Implementation

The objective of this section is to provide an insight on the major aspects concerning the implementation of the proposed DDTP framework. For the sake of simplicity, an example based on few missions, tasks and robot skills is considered. Note that, missions are executed in parallel in order to emphasize the framework capability to handle the concurrent execution of tasks and the sharing of resources.

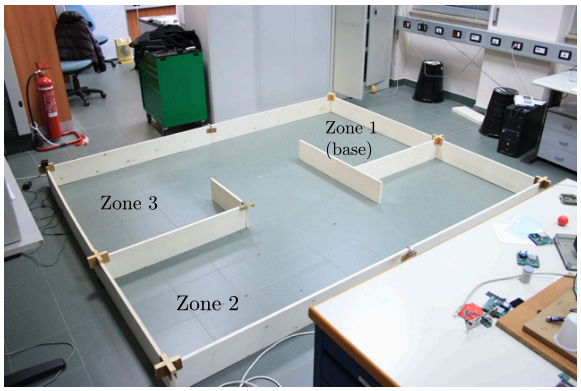


Fig. 4 The arena used for the experimental validation.

5.1 The Testbed

The testbed used for the experiments has the following characteristics: open software and hardware platform; small-size mobile robots; low cost; robust, flexible and scalable software. The main components of the system are the arena, i.e., the physical environment where the robots operate, and the workstation, i.e., the computer in charge of monitoring and logging data from robots through the communication with the DDTP architecture implemented on-board each robot.

The arena is a bounded and controlled environment in which the robots, wirelessly connected, can move to perform the desired tasks. In our implementation, the arena is a part of an indoor environment with boundaries and movable objects (i.e., small wooden items), which can be arranged to create walls and divide the environment in different zones. A short-range wireless connection is adopted to link the robot among themselves and the network with the workstation. The latter runs a software architecture, which is in charge of monitor the connections among robots in the network, using appropriate communication libraries. Furthermore, no a priori knowledge about the environment configuration is required by the control system. Fig. 4 shows the arena that has been used for the experimental validation of the proposed decentralized planning architecture. This is a rectangular area (2.5m $\times$ 3m) divided in three zones. Robots which at the beginning are grouped in the zone 1 (base-station), are supposed to perform two patrolling missions, respectively in zone 2 and zone 3. Briefly speaking, these missions involve the exploration of the assigned zones along with the detection (if any) of an event of interest. If the event is detected, one or more missions can be triggered, and thus different tasks have to be carried out. In order to perform these missions, robots are required to have several skills. In this experiments, no robot is assumed to be equipped with a sufficient set of skills to perform any

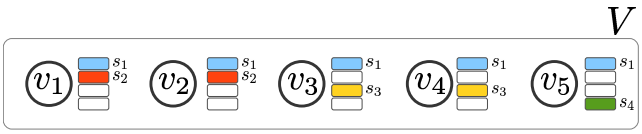


Fig. 5 Definition of the robotic network used in the experiments: the five robots with their skills represented by the stack of rectangles.

mission by itself. Differently, each robot has a subset of the required skills, possibly overlapping for redundancy. Therefore coordination among robots is mandatory for the correct execution of these missions. Experiments have been carried out by exploiting the mobile robotic platform SAETTA [22].

5.2 Network and Missions Setup

We consider a network composed by five SAETTA units $V = \{v_1, v_2, v_3, v_4, v_5\}$ as represented in Fig. 5. The RN is heterogeneous because each robot has a different sensor equipment apart from the the infrared-based system required for the navigation (skill s_1). In particular, robots $\{v_1, v_2\}$ are provided with a buzzer to fire an alarm (skill s_2); robots $\{v_3, v_4\}$ are equipped with a camera (skill s_3) for object recognition and finally robot $\{v_5\}$ is equipped with a light emitter (skill s_4). As a result the set of skills owned by robots is defined as $S = \{s_1, s_2, s_3, s_4\}$.

This configuration of skills allows the robots in V to perform the following tasks:

- *Patrolling* $\{t_1, t_5\}$: an infrared-based system for the detection of unexpected objects in the scene. This task requires the skill $\{s_1\}$;
- *Alarm Firing* $\{t_2, t_6\}$: an acoustic feedback to the user as a result of an event detection. This task requires the skills $\{s_1, s_2\}$;
- *Object Recognition* $\{t_3, t_7\}$: a vision-based system for the identification of specific objects. This task requires the skills $\{s_1, s_3\}$;
- *Environment Lighting* $\{t_4, t_8\}$: a lighting system based on a LED emitter to improve the recognition capability of the vision-based system. This task requires the skills $\{s_1, s_4\}$.

This tasks are organized in two types of mission: the *Surveillance* mission and the *Target Analysis* mission. Fig. 6 provides a diagram of the two types of missions. The experiment involves the execution of two instances of each mission, a pair performed in zone 2 and the other one performed in zone 3 of the arena. Formally, the set of mission $W = \{w_1, w_2, w_3, w_4\}$, contains w_1 and w_3 as the first and the second instances of the mission type Surveillance, with $T^{w_1} = \{t_1, t_2\}$ and

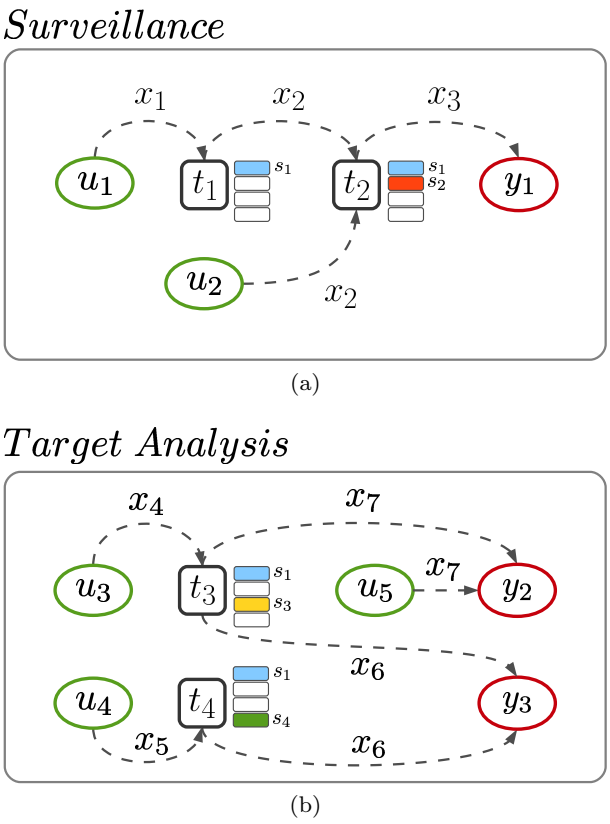


Fig. 6 Diagram of the two type of missions where labels refer to the an instance for each kind of mission, namely Surveillance mission (w_1) and Target Analysis mission (w_2). Note that, the same label is used for all the events, e.g. input event and/or completion of a task, which are precondition to fire another even, e.g. output event and/or task start.

$T^{w_3} = \{t_5, t_6\}$, while w_2 and w_4 denote the first and the second instances of the mission type Target Analysis, with $T^{w_2} = \{t_3, t_4\}$ and $T^{w_4} = \{t_7, t_8\}$. Notice that for missions w_2 and w_4 (Target Analysis) a race condition might arise for the shared resource, i.e., the environmental lighting which requires a light emitter owned only by the robot v_5.

Given the diagram of Fig. 6, we can derive the matrices related to the CMM for the proposed experiment. The sets $\{\mathbf{F_u}, \mathbf{F_t}, \mathbf{F_{u_c}}, \mathbf{F_y}\}$ and $\{\mathbf{S_u}, \mathbf{S_t}, \mathbf{S_{u_c}}, \mathbf{S_y}\}$ are the block-matrices describing respectively the set of pre-conditions and post-conditions for the overall experiment. It should be noticed that each of these matrices is a block-diagonal matrix composed of 4 blocks. This is due to the fact that two instances of each mission are considered for the experiment, thus we obtain:

$$\mathbf{F_u} = \mathtt{diag}(\mathbf{F_u}^{w_1}, \mathbf{F_u}^{w_2}, \mathbf{F_u}^{w_3}, \mathbf{F_u}^{w_4}).$$

In the same way the diagonal block matrices $\mathbf{F_t}$, $\mathbf{S_t}$, and $\mathbf{S_y}$ are built, while $\mathbf{F_{u_c}}$ is the 14×14 identity matrix, and $\mathbf{F_y}$ and $\mathbf{S_u}$ are null matrices. In particular,

the following matrices has been considered for missions w_1 and w_3:

$$\mathbf{F_u}^{w_1} = \mathbf{F_u}^{w_3} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \qquad \mathbf{F_t}^{w_1} = \mathbf{F_t}^{w_3} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix},$$

$$\mathbf{S_t}^{w_1} = \mathbf{S_t}^{w_3} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}^T, \qquad \mathbf{S_y}^{w_1} = \mathbf{S_y}^{w_3} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T,$$

and for missions w_2 and w_4:

$$\mathbf{F_u}^{w_2} = \mathbf{F_u}^{w_4} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{F_t}^{w_2} = \mathbf{F_t}^{w_4} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix},$$

$$\mathbf{S_t}^{w_2} = \mathbf{S_t}^{w_4} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^T, \qquad \mathbf{S_y}^{w_2} = \mathbf{S_y}^{w_4} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}^T.$$

Thus, we can derive the matrices related to the LMM. Note that, three different sets of matrices are considered to describe the different capabilities of the sets of robots $\{v_1, v_2\}$, $\{v_3, v_4\}$ and $\{v_5\}$. Furthermore, it should be noticed that also in the LMM formulation the matrix $\mathbf{F_{u_c}}^{[i]}$ is the identity matrix, and matrices $\mathbf{F_y}^{[i]}$, $\mathbf{S_u}^{[i]}$, and $\mathbf{S_{u_c}}^{[i]}$ are structurally null. As a consequence, only the set of matrices $\{\mathbf{F_u}^{[i]}, \mathbf{F_t}^{[i]}\}$ and $\{\mathbf{S_t}^{[i]}, \mathbf{S_y}^{[i]}\}$ are reported in the following.

`Agents` $\{v_1, v_2\}$`:`

$$\mathbf{F_u}^{[i]} = \begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \\ 0&0&0&0 \end{bmatrix}, \quad \mathbf{F_t}^{[i]} = \begin{bmatrix} 0&0&0&0 \\ 1&0&0&0 \\ 0&1&0&0 \\ 0&0&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{bmatrix},$$

$$\mathbf{S_t}^{[i]} = \begin{bmatrix} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&0&1 \end{bmatrix}^T, \quad \mathbf{S_y}^{[i]} = \begin{bmatrix} 0&0 \\ 0&0 \\ 1&0 \\ 0&0 \\ 0&0 \\ 0&1 \end{bmatrix}^T.$$

`Agents` $\{v_3, v_4\}$`:`

$$\mathbf{F_u}^{[i]} = \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix}, \quad \mathbf{F_t}^{[i]} = \begin{bmatrix} 0&0&0&0&0&0 \\ 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&1 \end{bmatrix},$$

$$\mathbf{S_t}^{[i]} = \begin{bmatrix} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&1 \\ 0&0&0&0&0&0 \end{bmatrix}^T, \quad \mathbf{S_y}^{[i]} = \begin{bmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 1&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&1&0&0 \\ 0&0&0&0 \\ 0&0&1&0 \\ 0&0&0&0 \\ 0&0&0&1 \end{bmatrix}^T.$$

`Agent` $\{v_5\}$`:`

$$\mathbf{F_u}^{[i]} = \begin{bmatrix} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \\ 0&0&0&0&0&0 \end{bmatrix}, \quad \mathbf{F_t}^{[i]} = \begin{bmatrix} 0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&1 \end{bmatrix},$$

$$\mathbf{S_t}^{[i]} = \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0 \end{bmatrix}^T, \quad \mathbf{S_y}^{[i]} = \begin{bmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 1&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&1&0&0 \\ 0&0&0&0 \\ 0&0&1&0 \\ 0&0&0&0 \\ 0&0&0&1 \end{bmatrix}^T.$$

Computational Cost

In order to evaluate the feasibility as well as the effectiveness of the proposed DDTP architecture, we investigate, using the tool defined in Section 3.3, the computational cost of the LMM formulation versus the centralized one.

As far as the computational complexity is concerned, let us recall that for the decentralized formulation, only a set of reduced-order matrices are considered. As explained in Section 4 this is related to the fact that in the local model, only tasks which can be effectively executed by an agent are described. Indeed, the more the network is heterogeneous the smaller the order of the matrices is. In the proposed scenario, we obtain the following *distribution degree* (equation 6) for each robot:

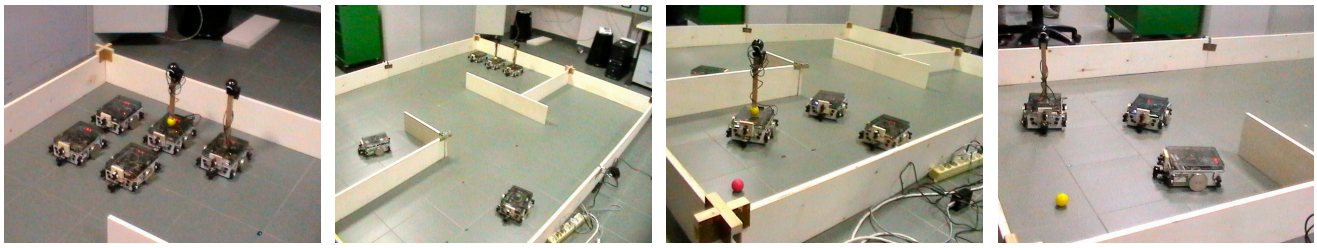
$$\delta_{v_1} = \delta_{v_2} = 0.18, \quad \delta_{v_3} = \delta_{v_4} = 0.52, \quad \delta_{v_5} = 0.52.$$

Thus, the whole robotic network has a *mean distribution degree* of $\langle\delta\rangle = 0.4$. This means that in average the computational load of each agent is approximatively 40% of the global (centralized) computational load. Nevertheless, it should be noticed that, this is only a (positive) side-effect of the proposed approach, being the main objective of this paper the decentralization of the DTP framework.

5.3 Experimental Results

In this subsection, the execution of an experiment of the network V which execute the four missions in W is presented. Note that to properly execute these missions each robot was implementing the algorithm [10] for estimating its location within the arena.

Fig. 8 shows the time trace of the experiment obtained by exploiting the data provided by the logging system of the robotic network. In particular, at time



(a) (b) (c) (d)

Fig. 7 Four different moments of the experiment. a) Robots grouped in the base-station. b) Robots v_1 and v_2 are reaching the areas to be monitored. c) Object detection has been performed in zone 2. d) Object detection has been performed in zone 3.

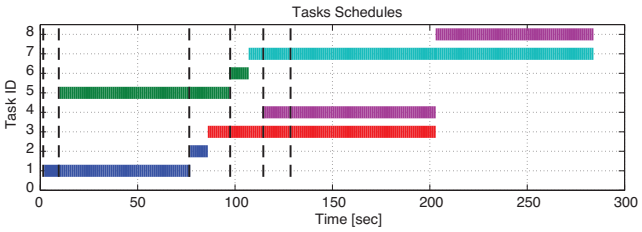


Fig. 8 Time trace of the experiment. Horizontal lines represent task duration while vertical (dashed) lines represent the time instant when an input event is triggered.

$\tau = 0$ all the robots are grouped in Zone 1 (Fig. 7-a), while at time $\tau = 1$ the first mission is triggered and the robot v_1 starts moving toward Zone 2 (task t_1) . Similarly, as shown in Fig. 7-b, at time $\tau = 9$ the second mission is triggered and the robot v_2 starts moving toward Zone 3 (task t_5). At time $\tau = 76$ an event is detected by the robot v_1 and thus the alarm is fired to provide an acoustic feedback to the user (task t_2). Furthermore, as a consequence of the alarm firing the *object recognition* task is fired at time $\tau = 77$ (task t_3). As a result, the robot v_3 starts moving towards Zone 2. At time $\tau = 114$ the input event *lighting request* is fired by the robot v_3 as it realizes the scene requires a better illumination (task t_4). Since the robot v_5, the only one equipped with a LED emitter, is available, the task can be started right away at time $\tau = 115$. In the meanwhile, at time $\tau = 97$ an event is detected by the robot v_2 and thus another alarm is fired to provide an acoustic feedback to the user (task t_6). In addition, as a consequence of the alarm firing the *object recognition* task is fired at time $\tau = 108$ (task t_7). As a result, being the robot v_4 available the task can be started and the robot v_4 starts moving towards Zone 3. At time $\tau = 127$ robot v_4 fires the *lighting request* event for Zone 3 as it realizes the scene requires a better illumination (task t_8). Note that, since the lighting resource owned only by the robot v_5 is not available at the moment of the request, the second mission is paused. It will be resumed at time $\tau = 203$, as the first mission ends at time $\tau = 202$ (Fig. 7-c), robot v_5 becomes available again and it can start moving towards Zone 3. Finally mission two ends at time $\tau = 283$ (Fig. 7-d).

5.4 Complexity Analysis of Overall Framework

In this subsection the number of messages required to be exchanged each time an event is triggered is investigated. For the sake of the complexity analysis, let us recall that three different kind of events should be considered: (i) mission started, (ii) task ended, (iii) robot added/removed. Note that, while the DecentralizedTask Execution (DTE) has to be executed each time an event is triggered, the Decentralized Task Assignment (DTA) algorithm is required only if a robot is added or removed, a mission is started or the path $\mathbf{p}^{[i]}$ of a robot is empty. In particular, while the first two events can be neglected as they happen rarely, the third one actually determines the frequency by which the DTA algorithm is run. Let us now review the complexity of the two algorithms to determine the dominant operation. As far as the abstract consensus algorithm is concerned, the CTE algorithm takes at most $\mathcal{D}(\mathcal{G}(k))$ steps to converge, therefore since each robot broadcast a message at each iteration the number of messages exchanged is of the order $\mathcal{O}\left(n\,\mathcal{D}(\mathcal{G}(k))\right)$. Regarding the DTA algorithm, we implements the CBBA algorithm [4], which satisfies the requirements given in Definition 22. It takes at most $\min\{\bar{m}, n\lambda\}\,\mathcal{D}(\mathcal{G}(k))$ steps to converge ([4, Theorem 1]), thus the number of messages to be exchanged is of the order $\mathcal{O}\left(n\,\min\{m, n\lambda\}\,\mathcal{D}(\mathcal{G}(k))\right)$. Clearly between the two algorithms, the DTA turns out to be dominant in terms of number of exchanged messages. Interestingly enough, if we consider only the fact a task is ended as a triggering event the analysis, the average complexity of the overall framework becomes a design parameter as the frequency by which a bundle becomes empty linearly depends on its length λ. Therefore, the larger the bundle is the less frequently the DTA algorithm is run.

However, it should be noticed that although a larger bundle implies a lower number of messages to be exchanged, it also implies that the assignment tends to a static one. This suggests that a proper tuning of the parameter λ is usually case-dependent.

6 Conclusion

In this paper a novel decentralized framework for the dynamic task planning problem for robotics networks has been proposed. The presented framework allows the robots to organize, coordinate and execute groups of tasks in an autonomous way. A decoupling mechanism of the structure of missions from the network of agents, by introducing the concept of skills has been proposed. This mechanism allows to decompose missions into distributed models for the decentralized dynamic task planning and execution control on board each robot. Furthermore, the correctness of the distributed task allocation and execution is guaranteed by proving the equivalent evolution of the proposed framework respect to its centralized version.

Future work will be focused on the extension of the proposed decentralized framework to robotic networks with a time-varying number of units, e.g., due to failures and/or temporary lack of connectivity.

A Boolean Algebra and Matrix Operations

A Boolean Algebra $\{\mathbb{B}, \otimes, \oplus, \neg, 0, 1\}$ is a six-tuple consisting of a set $\mathbb{B}$ called `universe`, equipped with two binary operations $\otimes$ called `and`, $\oplus$ called `or`, a unary operation $\neg$ called `complement` and two elements 0 and 1, such that the following axioms hold: associativity, commutativity, absorption, distributivity and complements.

Let us define a general $n \times m$ logical matrix as $\mathbf{M} \in \mathbb{B}^{n\times m}$, where $\mathbb{B}$ is the boolean set $\{0, 1\}$ and let us introduce the logical matrix product as follows:

Definition 23 (Logical Matrix Product) Let us consider two logical matrices $\mathbf{A} \in \mathbb{B}^{n\times m}$, $\mathbf{B} \in \mathbb{B}^{m\times p}$. The *logical matrix product* $\mathbf{C} \in \mathbb{B}^{n\times p}$ can be defined as: $\mathbf{C} = \mathbf{A} \odot \mathbf{B}$, where each element $\mathbf{C}_{i,j}$ is defined as:

$$\mathbf{C}_{i,j} = \bigoplus_{r=1}^{m} \mathbf{A}_{i,r} \otimes \mathbf{B}_{r,j}$$

for each pair (i, j) with $i = \{1, ..., n\}$ and $j = \{1, ...p\}$.

Heres an example to clarify the above definition:

Example

Given the Boolean matrix $\mathbf{A} \in \mathbb{B}^{2\times 2}$, and the Boolean column vector $\mathbf{b} \in \mathbb{B}^2$, defined as follows:

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

the logical matrix product $\mathbf{c} \in \mathbb{B}^2$, is given by

$$\begin{aligned} \mathbf{c} &= \mathbf{A} \odot \mathbf{b} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \odot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} (1 \otimes 1) \oplus (0 \otimes 0) \\ (0 \otimes 1) \oplus (1 \otimes 0) \end{bmatrix} = \begin{bmatrix} 1 \oplus 0 \\ 0 \oplus 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \end{aligned}$$

References

1. Aragues, R., Cortes, J., Sagues, C.: Distributed consensus on robot networks for dynamically merging feature-based maps. IEEE Transactions on Robotics **28**(4), 840–854 (2012)
2. Brass, P., Cabrera-Mora, F., Gasparri, A., Jizhong, X.: Multirobot tree and graph exploration. Robotics, IEEE Transactions on **27**(4), 707–717 (2011). DOI 10.1109/TRO.2011.2121170
3. Burgard, W., Moors, M., Stachniss, C., Schneider, F.: Coordinated multi-robot exploration. Robotics, IEEE Transactions on **21**(3), 376 – 386 (2005). DOI 10.1109/TRO.2004.839232
4. Choi, H., Brunet, L., How, J.: Consensus-based decentralized auctions for robust task allocation. IEEE Transactions on Robotics **25**(4), 912 – 926 (2009)
5. Costelha, H., Lima, P.: Robot task plan representation by petri nets: modelling, identification, analysis and execution. Autonomous Robots **33**(4), 337–360 (2012). DOI 10.1007/s10514-012-9288-x. URL `http://dx.doi.org/10.1007/s10514-012-9288-x`
6. Defoort, M., Floquet, T., Kokosy, A., Perruquetti, W.: Sliding-mode formation control for cooperative autonomous mobile robots. IEEE Transactions on Industrial Electronics **55**(11), 3944 –3953 (2008). DOI 10.1109/TIE.2008.2002717
7. Di Paola, D., Gasparri, A., Naso, D., Lewis, F.: Decentralized discrete-event modeling and control of task execution for robotic networks. In: Decision and Control (CDC), 2012 IEEE 51st Annual Conference on, pp. 7346–7351 (Dec.). DOI 10.1109/CDC.2012.6426687
8. Di Paola, D., Gasparri, A., Naso, D., Ulivi, G., Lewis, F.L.: Decentralized task sequencing and multiple mission control for heterogeneous robotic networks. In: Proc. IEEE International Conference on Robotics and Automation (2011)
9. Fagiolini, A., Pellinacci, M., Valenti, G., Dini., G., Bicchi, A.: Consensus-based distributed intrusion detection for multi-robot systems. In: IEEE International Conference on Robotics and Automation, 2008. ICRA 2008., pp. 120–127 (2008). DOI 10.1109/ROBOT.2008.4543196
10. Gasparri, A., Prosperi, M.: A bacterial colony growth algorithm for mobile robot localization. Autonomous Robots **24**(4), 349–364 (2008). DOI 10.1007/s10514-007-9076-1
11. Gerkey, B.P., Matarić, M.J.: A formal analysis and taxonomy of task allocation in multi-robot systems. The Intl. J. of Robotics Research **23**(9), 939 – 954 (2004)
12. Giordano, V., Ballal, P., Lewis, F., Turchiano, B., Zhang, J.: Supervisory control of mobile sensor networks: math formulation, simulation, implementation. IEEE Transactions on Systems, Man and Cybernetics, Part B **36**(4), 806 – 819 (2006)
13. Giordano, V., Jing, B.Z., Naso, D., Lewis, F.: Integrated supervisory and operational control of a warehouse with a

matrix-based approach. IEEE Transactions on Automation Science and Engineering **5**(1), 53 –70 (2008). DOI 10.1109/TASE.2007.891472

14. Huq, R., Mann, G., Gosine, R.: Behavior-modulation technique in mobile robotics using fuzzy discrete event system. IEEE Transactions on Robotics **22**(5), 903 –916 (2006). DOI 10.1109/TRO.2006.878937
15. Ji, M., Sarkar, N.: Supervisory fault adaptive control of a mobile robot and its application in sensor-fault accommodation. IEEE Transactions on Robotics **23**(1), 174 –178 (2007). DOI 10.1109/TRO.2006.889481
16. Jones, C.V., Matarić, M.J.: Behavior-based coordination in multi-robot systems. In: S. Ge, F. Lewis (eds.) Autonomous Mobile Robots: Sensing, Control, Decision-Making, and Applications. Marcel Dekker, Inc. (2005). URL `http://robotics.usc.edu/publications/466/`
17. Meyer, W., Drathen, A.: Collaboration and collision functions for plan-based and event-driven mission control. In: D. Yang (ed.) Informatics in Control, Automation and Robotics, *Lecture Notes in Electrical Engineering*, vol. 133, pp. 503–510. Springer Berlin Heidelberg (2012). DOI 10.1007/978-3-642-25992-0_69. URL `http://dx.doi.org/10.1007/978-3-642-25992-0_69`
18. Mireles, J., Lewis, F.: Deadlock analysis and routing on free-choice multipart reentrant flow lines using a matrix-based discrete event controller. In: Proceedings of the 41st IEEE Conference on Decision and Control, 2002, vol. 1, pp. 793 – 798 vol.1 (2002). DOI 10.1109/CDC.2002.1184602
19. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4), 541–580 (1989)
20. Pallottino, L., Scordio, V., and Frazzoli, A.B.E.: Decentralized cooperative policy for conflict resolution in multivehicle systems. IEEE Transactions on Robotics **23**(6), 1170 –1183 (2007). DOI 10.1109/TRO.2007.909810
21. Pinedo, M.L.: Scheduling: Theory, Algorithms, and Systems. Springer (2008)
22. Rocco, M.D., Gala, F.L., Ulivi, G.: Saetta: A small and cheap mobile unit to test multirobot algorithms. IEEE Robotics Automation Magazine (2012). DOI 10.1109/MRA.2012.2185991. Available online
23. Song, M., Tarn, T., Xi, N.: Integration of task scheduling, action planning, and control in robotic manufacturing systems. Proceedings of the IEEE **88**(7), 1097 –1107 (2000). DOI 10.1109/5.871311
24. Tacconi, D., Lewis, F.: A new matrix model for discrete event systems: application to simulation. IEEE Control Syst. Mag. **17**(5), 62 – 71 (1997)
25. Tiehua, C., Sanderson, A.: Task sequence planning using fuzzy petri nets. IEEE Transactions on Systems, Man and Cybernetics **25**(5), 755 –768 (1995). DOI 10.1109/21.376489
26. Wang, Z., Gu, D.: Cooperative target tracking control of multiple robots. IEEE Transactions on Industrial Electronics **59**(8), 3232 –3240 (2012). DOI 10.1109/TIE.2011.2146211
27. Zavlanos, M., Pappas, G.J.: Distributed connectivity control of mobile networks. IEEE Transactions on Robotics **24**(6), 1416 – 1428 (2008)
28. Zhang, H., Lewis, F., Qu, Z.: Lyapunov, adaptive, and optimal design techniques for cooperative systems on directed communication graphs. IEEE Transactions on Industrial Electronics **59**(7), 3026 –3041 (2012). DOI 10.1109/TIE.2011.2160140
29. Zouaghi, L., Alexopoulos, A., Wagner, A., Badreddin, E.: Mission-based online generation of probabilistic monitoring models for mobile robot navigation using petri nets. Robotics and Autonomous Systems **62**(1), 61 – 67 (2014). DOI http://dx.doi.org/10.1016/j.robot.2012.07.012. URL `http://www.sciencedirect.com/science/article/pii/S0921889012001157`