Universitá degli Studi Roma Tre
Dipartimento di Informatica e Automazione

Rapporto Tecnico

n.192

Implementazione di un algoritmo distribuito per il sequenziamento di task in un sistema ad agenti

Donato Di Paola

Febbraio 2012

Abstract

In questa relazione viene definito il problema principale e, successivamente, viene descritta la tecnica risolutiva adottata al fine di implementare un algoritmo distribuito per il sequenziamento di task in un sistema ad agenti. In particolare, il problema del sequenziamento di task per un sistema ad agenti viene suddiviso in due sotto-problemi correlati. Il primo é relativo alla definizione delle regole di precedenza tra i task e delle modalitá con cui identificare insiemi di task che servono a raggiungere un obiettivo specifico. Il secondo problema riguarda l'assegnamento di task ad agenti effettivamente in grado di eseguire i task richiesti. In quest'ultimo caso viene posta in evidenza una delle caratteristiche fondamentali dello scenario affrontato, ovvero l'assunzione di una rete di agenti eterogenea. In tale tipo di sistemi gli agenti posseggono capacitá diverse, pertanto l'assegnamento dei task ed il successivo coordinamento durante la fase di esecuzione richiedono il soddisfacimento di vincoli appositamente predisposti. Inoltre, nella presente relazione é illustrata l'architettura che implementa una possibile strategia risolutiva per il problema proposto. Tale architettura permette, attraverso un approccio modulare, di considerare i problemi di assegnamento e sequenziamento come due componenti che, interagendo tra loro, portano ad un miglioramento delle prestazioni globali dell'intero sistema. Infine, l'architettura distribuita proposta nella presente relazione é stata implementata come toolbox nel linguaggio di programmazione Matlab®.

1 Introduction

In this report we will consider the decentralized framework for the control of tasks execution for networks of heterogeneous robotic agents. Assuming a decentralized task assignment algorithm to be available, we will focus on the problem of the decentralized control of tasks execution. This problem concerns the design of a decentralized control system whose objective is to monitor the execution of a set of tasks characterized by precedence constraints, which has to be carried out by a network of agents.

2 Problem Statement

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.

A task can be considered as a generic job or action which can be executed by an agent (e.g. reaching a target, measuring the temperature, finding an object in a real-world environment, etc.). More precisely, a task is the atomic unit of work that an agent can do. Thus, an agent can execute a single task at time. Tasks in T can be subject to two categories of constraints: *assignment* and *precedence constraints*. The assignment constraints, related to each task, define if a task can be assigned to an agent. A task can be assigned to an agent only if the characteristics (skills) of the agent allow the execution of that task. The precedence constraints, related to a group of tasks, define the order of execution of the tasks in T, i.e. one or more tasks may have to be accomplished before another task is allowed to be started.

The network V is *heterogeneous* because each agent can be configured with a different set of capabilities called *skills*. Skills either reflect the structural capabilities of the robotic agent (for instance, some agents fly while others walk because of their locomotion and sensory system) or allow to define abstract categories in order to establish a hierarchy among agents (for instance, some agents might serve as a communication network to the others). The skills owned by an agent define which tasks the agent is able to execute. This means that each agent owns a set of skills to perform a number of tasks and consequently each task requires a set of skills in order to be executed by an agent.

Definition 2.1 (Agent Skill Set)**.** *Given the agent $v_i \in V$, said the S_{v_i} to be the set of skills owned by v_i, we define $S_V = \cup_{v_i \in V} S_{v_i}$ as the set of all skills owned by all agents in V.*

Definition 2.2 (Task Skill Set)**.** *Given the task $t_j \in T$, said S_{t_j} to be the set of skills required by the task t_j to be executed, we define $S_T = \cup_{t_j \in T} S_{t_j}$ as the set of all skills required by all tasks in T.*

It is interesting to notice that the skills can be considered as links between the agent set V and task set T, which allow to define an important property of the task set T.

Definition 2.3 (Feasible Task Set)**.** *Given a set of tasks T, this is said to be feasible for the network V, if the following holds: skills required by all tasks in T are owned by the network V and, for each task in T, exists at least one agent in V which is able to execute that task. Formally, the set T is feasible if:*

$$\begin{aligned} &S_T \subseteq S_V, \\ &\exists\, v_i : S_{t_j} \cap S_{v_i} = S_{t_j} \quad \forall t_j \in T. \end{aligned} \tag{1}$$

This property can be used to verify if a generic set of tasks can be assigned and then executed by a network of agent.

2.1 The Control of Task Execution

We define a *mission* as a group of tasks belonging to the main set T and aiming at a single goal.

Definition 2.4 (Mission)**.** *A mission w is defined by the tuple:*

$$\langle T_w, S_w, X_w, U_w, Y_w \rangle \tag{2}$$

where $T_w \subseteq T$ is the set of tasks involved in the mission w; $S_w \subseteq S_T$ is the set of skills required by the tasks in T_w; X_w is the set of rules which describes the logical links among tasks within the mission w and determines the execution flow of tasks; 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 condition by which the mission w can start (a sensory input, a user command, etc.). Output events in Y_w are symbols used to indicate the completion of the mission w.

If we want to achieve several goals, several missions have to be defined. Thus, it is necessary to introduce the following definition:

Definition 2.5 (Missions Set)**.** *We define $W = \{w_1, \ldots, w_q\}, q \in \mathbb{N}$ as the set of all missions that have to be completed by all agents in the network. As a result, the sets $T_{w_k}, k \in \{1, \ldots, q\}$ are a partition of the set T, that is to say $\cup_{w_k \in W} T_{w_k} = T$ and $\cap_{w_k \in W} T_{w_k} = \emptyset$.*

It is necessary to note that precedence constraints and assignment constraints are captured by the concept of set of rules and by the concept of skills respectively. A more detailed formalization of skills and thus of the heterogeneity of the network will be discussed in the next subsection on task assignment. Now we will instead define the concept of rule.

A rule in a mission can be defined as a correspondence of *causes* and *effects* [**?**]. We define causes as *preconditions* and effects as *postconditions*. For example, given four tasks t_1, t_2, t_3 and $t_4 \in T_w$, for the mission w, if t_3 and t_4 have to be executed in parallel after the completion of t_1 and t_2, we define the rule for this case as follows:

RULE: x_1
PRECONDITION: $\texttt{IsCompleted}(t_1) \wedge \texttt{IsCompleted}(t_2)$
POSTCONDITION: $\texttt{Start}(t_3) \wedge \texttt{Start}(t_4)$

where the symbol $\wedge$ indicates the logical *and*, the function `IsCompleted`$(\cdot)$ is true when the task is completed, and the function `Start`$(\cdot)$ starts the actual execution of a task.

Using the previous definition, the mission w can be expressed in terms of rules with preconditions and postconditions and defined by the elements of the sets in (**??**). More specifically, preconditions are expressed by completed tasks in T_w, required skills in S_w and input events in U_w, while postconditions are expressed by tasks that have to be executed in the set T_w, available skills in S_w and output events in Y_w.

It is important to point out that rules can be defined only as links among tasks on the same mission. In the proposed model logical links among different missions are not allowed. Thus, if two or more missions have to be executed at the same time, the control system must guarantee the parallel execution of tasks from different missions ensuring the validity of constraints among tasks in the same mission. Moreover, tasks are non-preemptive [**?**], i.e. the execution of a task cannot be resumed if it has been interrupted. A task that has been interrupted is considered completed. Given the above definitions, we are now ready to define the problem of the control of task execution as follows:

Problem 2.1 (Control of Task Execution)**.** *Given a heterogeneous network of agents V and a feasible set of tasks T, on which a set of missions W is defined, control the concurrent execution of missions ensuring their accomplishment. In the execution of each mission $w_k \in W$, all tasks in T_{w_k} have to match the precedence constraints defined by rules in X_{w_k} and assignment constraints expressed by skills in S_{w_k} in order to achieve all goals defined by missions in W.*

2.2 The Task Assignment for Heterogeneous Networks

Consider the heterogeneous network of agents $V = \{v_1, \ldots, v_n\}$, the task set $T = \{t_1, \ldots, t_m\}$, and the skill set $S_V = \{s_1, \ldots, s_f\}$, representing all the skills owned by all agents in V, the objective of the task assignment problem for heterogeneous networks is to find an assignment of tasks to agent, considering the ability of agents to perform the tasks, that fulfills all the constraints and maximizes a global reward.

In the classical task assignment problem, an assignment is said to be *free of conflicts* if each task is assigned to no more than one agent at a time. Moreover, the case in which only one task can be assigned to each robot is called *single-assignment* problem, while the more general case in which each robot can handle a sequence of maximum λ tasks is referred to as *multi-assignment* problem. Since each robot is capable of executing at most one task at a time, in the case of multi-assignment, tasks will be executed in an ordered sequence. We will use λ to represent the length of this sequence and suppose that λ is equal for each agents in V. The assignment is said to be *complete* if the number of assigned tasks is equal to the maximum number of *assignable tasks* represented by $m^\star$. In homogeneous networks $m^\star$ can be easily computed as $m^\star = \min\{m, n\lambda\}$. In heterogeneous networks $m^\star$ depends instead on the characteristics of the

available agents as well as on those of the tasks that have to be executed. A way to compute it in heterogeneous networks is given in [**?**].

In the task assignment for heterogeneous networks each task can be assigned only to agents that have the skills needed to perform it, as defined in the set S_V. This type of assignment constraint is taken into account by the *Matrix of Possible Assignments* $\mathbf{\Delta} \in \mathbb{B}^{n \times m}$. The element $\mathbf{\Delta}_{ij} = 1$ indicates that task $t_j \in T$ can be assigned to the robotic agent $v_i \in V$ and 0 otherwise. We suppose that the task set T is feasible for the network V, thus $\mathbf{\Delta}$ always verifies the property (**??**).

The task assignment problem described before can be written as the following binary integer program. Given n agents, with index set $\mathcal{I} = \{1, \ldots, n\}$, and m tasks, with index set $\mathcal{J} = \{1, \ldots, m\}$, and the matrix of possible assignment $\mathbf{\Delta}$, find the decision variables $z_{i,j} \in \mathbb{B}$ such that:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} z_{ij} \tag{3}$$

subject to:

$$\sum_{j=1}^{m} z_{ij} \leq \lambda \qquad \forall i \in \mathcal{I} \tag{4}$$

$$\sum_{i=1}^{n} z_{ij} \leq 1 \qquad \forall j \in \mathcal{J} \tag{5}$$

$$z_{ij} \leq \mathbf{\Delta}_{ij} \qquad \forall (i,j) \in \mathcal{I} \times \mathcal{J} \tag{6}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m} z_{ij} = m^{\star} \tag{7}$$

where $z_{ij} = 1$ if task $j \in \mathcal{J}$ is assigned to agent $i \in \mathcal{I}$ and 0 otherwise and the score value $c_{ij} \geq 0$ represents the reward if agent i performs the task j. The constraints (**??**)-(**??**) express, respectively, the requirements that the number of tasks assigned to a agents should not exceed its capacity λ, that a task should be assigned to no more than one agent, that a task should be assigned only to agents able to perform it and that the number of assigned tasks should be equal to the maximum value $m^{\star}$ for the current task and agent sets.

As stated in the previous subsection, scheduling constraints among tasks in T exist and each agent is able to execute at most one task at a time, thus the order in which the tasks are assigned to each agent has to be defined too. In other words, the goal of the task assignment problem is to determine for each agent $v_i \in V$ an ordered sequence (of length λ) of tasks that will have to be assigned and then executed, this sequence is called path $\mathbf{p}^{[i]}$.

Following the taxonomy for task allocation in the multi-robot systems proposed in [**?**], the problem (**??**)-(**??**) falls into the category of Single-task Robots, Single-robot Tasks, Time-extended Assignment (ST-SR-TA). This means that: each robot is capable of executing at most one task at a time, each task requires

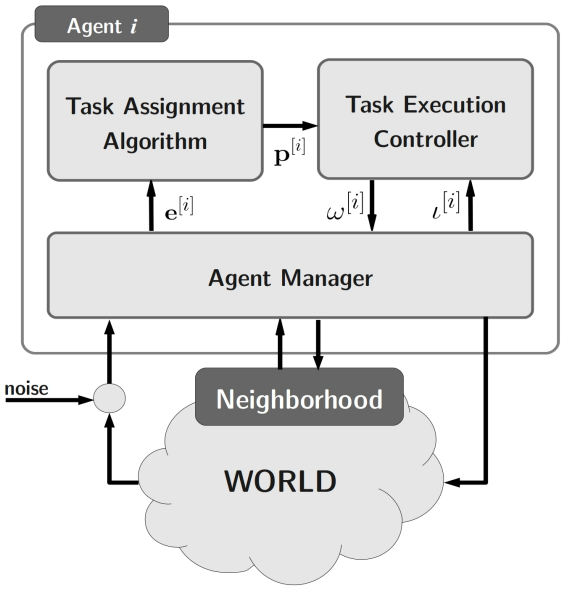

Figure 1: The Diagram of the Decentralized Planning Architecture.

exactly one robot to be executed, and a set of tasks is assigned to a robot as a plan to be executed. Problems in this category have been shown to be NP-hard. Adding the concept of heterogeneity to the classical task assignment problem, we can formulate the problem as follows:

Problem 2.2 (Task Assignment for Heterogeneous Networks)**.** *Given a heterogeneous network of agents V and a set of tasks T, find a solution to the problem (??)-(??) to ensure the best assignment, with respect to a predefined metric, and taking into account the heterogeneity constraints of the network expressed by the skills.*

3 Decentralized Planning Architecture

In this section, we outline the planning architecture which implements the control of task execution and the task assignment in a decentralized structure. The diagram of the architecture, for agent $i \in V$, is depicted in Figure **??**. Looking at the architecture from a global point of view we identify tree main components: the Agent Manager, the Task Execution Controller, and the Task Assignment Algorithm.

3.1 The Agent Manager

All the input and output channels of the planning architecture, for the i-th agent, are managed by the *Agent Manager*. The Agent Manager can be viewed as an interface between the external environment and the internal modules of each agent. This component provides a low-level interface between the decentralized architecture and the environment: it manages the sensory system detecting events which will be sent to the other modules and manages the actuating system making effective the actions obtained from the decision-making components. It is important to note that when an event is detected the Agent Manager sends a message to both the Task Execution Controller and Task Assignment Algorithm, this ensures the coherence of the internal information of these two modules. More specifically, the event $\iota^{[i]}$ is the message sent to the Task Execution Controller and the event $\mathbf{e}^{[i]}$ is the message sent to the Task Assignment Algorithm. The former contains the tasks completed and the events which trigger the missions, the latter determines the trigger of the algorithm based on updated information (updated set T and V). Moreover the Agent Manager manages the communication with the neighbor agents of the agent i in order to make possible the exchange of messages during the communication phases triggered by other modules of the architecture.

3.2 The Task Execution Controller

The *Task Execution Controller* implements the decentralized control strategy presented for the execution of tasks, as the strategy presented in [**?**]. This component contains both static information (matrices and vectors related to the configuration of mission in W and the agent i) and dynamic information (about the status of mission vectors) that are used by the algorithm execution control. Through communication with the Agent Manager the Task Execution Controller receives events ($\iota^{[i]}$) that determine the evolution of missions and returns ($\omega^{[i]}$) output events determined by the control of tasks execution algorithm. These events are translated into commands from the Agent Manager which commands the real execution of tasks in the environment. The behavior of the control algorithm is also a function of the result of task assignment, so there is direct communication between the Task Execution Controller and the Task Assignment Algorithm. More precisely ,the Task Assignment Algorithm, triggered by the event $\mathbf{e}^{[i]}$, sends the new assignment for the agent i (the path $\mathbf{p}^{[i]}$). Since these two modules are updated simultaneously by the Agent Manager, each time the Task Execution Controller calls the Task Assignment Algorithm it will always receive results based on consistent information.

3.3 The Task Assignment Algorithm

The *Task Assignment Algorithm* encapsulates a decentralized task assignment method as [**?**], [**?**] . Given the current information on the agent's state and the current set of tasks the Task Assignment Algorithm returns the task-to-agent

assignment for all agents in the network. Obviously, for the purposes of local control of the agent i is not necessary to know the assignment of all agents, thus only the path $\mathbf{p}^{[i]}$ is sent to the Task Execution Controller. However the global assignment may be used in some particular cases (for example if one or more agents failed) to trigger the task assignment algorithm only on a subset of tasks and not on the entire set of tasks. It is important to note that the Task Assignment Algorithm does not send messages to the Agent Manager, but communicates only with the Task Execution Controller. This means that, from a design point of view, the Task Assignment Algorithm could be viewed as a sub-component of the Task Execution Controller. However, we designed the decentralized planning architecture so that the Task Assignment Algorithm module can be changed easily, for example for a comparison of different distributed task assignment algorithms.