

Assignment of Dynamically Perceived Tasks by Token Passing in Multi-Robot Systems

A. Farinelli, L. Iocchi, D. Nardi, V. A. Zuparo
Dipartimento di Informatica e Sistemistica
University of Rome "La Sapienza"
Via Salaria 113, 00198 Rome, Italy
E-mail: last-name@dis.uniroma1.it

Abstract

The problem of assigning tasks to a group of robots acting in a dynamic environment is a fundamental issue for a Multi Robot System (MRS) and several techniques have been studied to address this problem. Such techniques usually rely on the assumption that tasks to be assigned are inserted into the system in a coherent fashion. In this work we consider a scenario where tasks to be accomplished are perceived by the robots during mission execution. This issue has a significative impact on the task allocation process and, at the same time, makes it strictly dependent on perception capabilities of robots. More specifically, we present an asynchronous distributed mechanism based on Token Passing for allocating tasks in a team of robots.

We tested and evaluated our approach by means of experiments both in a simulated environment and with real robots; our scenario comprises a set of robots that must cooperatively collect a set of objects scattered in the working environment. Each object collection task requires the cooperation of two robots. The experiments in the simulation environment allowed us to extract quantitative data from several missions and in different operative conditions and to characterize in a statistical way the results of our approach, especially when the team size increases.

Index Terms

Multi-Robot Systems, Coordination, Task Assignment.

I. INTRODUCTION

Cooperation among robots is nowadays regarded as one of the most challenging and critical issues towards fieldable robotic systems. Indeed, not only do the robots need to take into account the presence of other robots, but they also need to cooperate in order to improve the performance of the whole system [1]–[4]. In this article we are concerned with systems composed by multiple cooperating robots, i.e. teams of robots.

A central problem for achieving cooperation in Multi Robot Systems is *Task Assignment*, i.e. the problem of decomposing the task faced by the system into smaller sub-tasks, and to ensure that they can be accomplished by individual robots without interference and, more generally, with better performance.

Task Assignment has been deeply investigated in both Multi Agent Systems (MAS) and Multi Robot Systems (MRS) [5]–[11], and several successful approaches have been proposed. However, the growing complexity of applications makes it desirable to improve current approaches to Task Assignment in order to suitably deal with more and more challenging requirements: dynamic task evolution, strict bounds on communication and constraints among tasks to be executed. But, most notably, in real world applications involving MRS tasks to be assigned cannot be inserted into the system in a centralized fashion: they are perceived by each robot during mission execution. Moreover, noisy and limited perception of the surrounding environment not only influences the performance of individual robotic agents, but it affects the cooperation among robots. Specifically, *perception capabilities significantly impact on the task allocation process*. For example, let us consider a task where robots need to correctly identify objects of similar shape and color. This requires to consider, in the data association process, properties that can change with time (such as object position in the working environment). Consequently, errors are

frequent and can easily lead to conflicts in the task allocation process. Conflicts on the tasks to be accomplished can obviously cause serious inefficiencies of the overall system.

Perception capabilities thus play a fundamental role in every application involving multiple robots and significantly constrain the design of the Task Assignment technique. Task Assignment approaches developed for MAS (such as for example [5], [12], [13]) usually do not take into account the perception capabilities of agents involved in the task allocation process; such techniques are therefore not directly applicable to our reference scenarios. As for MRS, previous approaches to Task Assignment include: i) Sequential Task Assignment ([14], [15]), where tasks are allocated to robots sequentially as the tasks enter the system; ii) Iterative Task Assignment ([16], [17]), where all tasks present in the system are allocated from scratch at each time step; iii) Reactive Task Assignment [11], where each member of the team decides whether to engage itself in a task, without (re)-organizing the other member activities. While in all these works the perception capabilities of robots influence the technique for Task Assignment, the conflicts in the allocation process due to dynamic task generation are not explicitly taken into account. More generally, the assumptions that often underlie the work on Task Assignment are that either the tasks are defined statically (when application domains are simple enough) or that they can be unambiguously detected by a team member. A notable exception is the work by Zlot et al. [10], where a team of robots involved in an exploration task dynamically exchanges points of interest to be visited. The system is able to avoid conflicts arising from dynamically perceived tasks through an auction mechanism based on broadcast communication.

A reference application in which Task Assignment and dynamic perception play a crucial role is mobile heterogeneous robots forming a sensor network. These are multi-robot systems formed by heterogeneous robots (with different kinds of sensors on board) that perform an exploration task to examine/analyze a set of objects scattered in the environment. Tasks (i.e. objects to be analyzed) are discovered during the mission execution, and dynamic task assignment must be used to improve the performance of the system. The object might need to be analyzed with different sensors (mounted on different robots) at the same time, thus tasks might be tied by execution constraints. An example of such reference scenario can be found in the field of rescue robotics, where the goal is to find victims in a post-disaster scenario and to measure many parameters about the victims. Here approaches based on several small robots with different sensors and limited computation and network capabilities are more suitable than big and powerful robots.

Since this is a very complex scenario, in this article we refer to a simpler test-bed which encapsulates all the interesting features for coordination. In particular, we focus on multi-robot foraging for two main reasons: i) it is a well-known test-bed for Multi-Robot systems [1], ii) it can well represent all the main coordination issues required in the described scenario. Multi-robot foraging requires robots to detect several objects scattered in the environment and to collect them in a desired position. In addition, we have defined an application scenario for multi-robot foraging in which the collecting of each object requires that exactly two robots help each other to grab it (a helper robot and a collector robot). After the grabbing phase, only one robot is needed to transport the object. The number and position of the objects in the environment are not known, thus enforcing task discovery through perception. Moreover, objects are identical, hence they can only be distinguished by their position in the environment; therefore robots must consider, in the data association process, properties that change over time.

Summarizing, from the coordination perspective our scenario has the following characteristics: i) tasks are discovered and created during mission execution; ii) tasks may require multiple agents to perform them, and such agents must synchronize their actions; iii) agents may perform one or more tasks, but within resource limits; iv) too many agents fulfilling the same task lead to conflicts that need to be avoided; v) properties that distinguish tasks can vary over time. This provides a challenging scenarios for the Multi-Robot system as compared with previous MRS test beds [1].

In this article we focus on the integration of dynamic task discovery within a distributed Task Assignment method based on Token Passing (extending our previous work [18]). We present an approach to distributed Task Assignment that allows for dynamically assigning perceived tasks in a team of robotic agents, while avoiding general types of conflicts among team members, arising when tasks have properties that change over time. More specifically, the coordination system presented here takes into consideration changes in the environment by directly

perceiving them and by minimizing communication requirements.

Most previous approaches to Task Assignment in MRS address particular types of conflicts that arise through the allocation process using specific procedures that are related to the application domain. Moreover, none of the previous approaches explicitly address the problem of minimizing the communication overhead to guarantee a conflict-free allocation.

The basic idea of our approach is based on a Token Passing technique [19] that has recently been adopted for Task Assignment in MRS [18]. Token Passing embodies several features that make it suitable not only to handle conflicts arising from perception, but also to embody constraints on task execution. Moreover, Token Passing can be implemented with limited communication requirements.

Tokens are used to represent tasks that must be executed by the agents. Each team member creates, executes and propagates these tokens based on its knowledge of the environment. The basic approach relies on the assumption that one token is associated to every task to be executed and that the token is maintained only by the agent that is performing such a task. If the agent is not in the condition of performing the task, it can decide to pass the token on to another team member. Token Passing assigns tasks using only a broad knowledge of team mates, sharing a minimal set of information among team members. The approach ensures that task allocation is highly reactive and requires low communication.

To apply Token Passing in a system composed by physical robots, we introduce the new concept of *dynamic token generation*. In fact, tokens are not statically predefined, but generated on-line during mission execution as result of robot perceptions. An asynchronous distributed algorithm is used to detect and solve conflicts due to simultaneous or erroneous task perception by several robots. Our approach guarantees a conflict-free allocation of exactly n agents for each task.

We have implemented and tested the proposed approach with a team of AIBO robots collecting colored balls scattered in the environment. Moreover, in order to present a quantitative analysis of our approach, we have used a simulation environment that emulates MRS behaviors in dynamic environments and allows to run extensive experiments. The reported results show that the proposed approach can allocate exactly 2 robots to each task, avoiding possible conflicts with other team mates and maintaining a very low communication bandwidth.

The remainder of this article is structured as follows. The next section presents a formal definition of the Task Assignment problem. Section III describes the Token Passing approach to Task Assignment. Section IV illustrates how dynamic task perception impacts on Task Assignment. Section V describes our proposed technique for handling conflicts arising from the dynamic generation of tokens. Section VI presents the implementation and results obtained from a foraging task with AIBO robots, as well as a quantitative analysis of the performance of the algorithm in a simulation framework. In Section VII we address related work and put the proposed approach in perspective with the state of the art in the field. Finally, Section VIII draws conclusions and sketches future work.

II. THE TASK ASSIGNMENT PROBLEM

The problem of assigning a set of tasks to a set of robots can be easily framed as a Generalized Assignment Problem (GAP) [20]. However, while GAP is well defined for a static environment, where agents and tasks are fixed and capabilities and resources do not depend on time, in multi-robot applications a problem with the defined parameters changing with time must be solved. Indeed, several methods for Dynamic Task Assignment implicitly take into consideration such an aspect: solutions that consider the dynamics of the world are proposed and Task Allocation methods that approximate solutions of the GAP problem at each time step are derived [9]–[11].

The GAP formulation fails to model all the relevant aspects for our interest domains. In particular, it does not consider two main issues: i) tasks to be accomplished can be tied by constraints, ii) the set of tasks is not known a priori when the mission starts, but it is discovered and dynamically updated during task execution.

We will use the following notation: $E = \{e_1, \dots, e_n\}$ denotes the set of robots. While in general robots involved in the Task Assignment process can also vary over time, in this contribution we focus on a predefined static set of robots.

We will denote tasks by $\tau^{[t_s, t_e]}$, where $[t_s, t_e]$ is the time interval in which the task is present in the system. We denote with Γ_t the set of tasks which are present at time t , i.e. $\Gamma_t = \{\tau^{[t_s, t_e]} \mid t_s \leq t \leq t_e\}$, and with $m(t) = |\Gamma_t|$.

Since values characterizing a task τ may vary over time, we use τ_k^t to denote the status of task τ_k at time t . However, in the following, time specifications for the tasks will be dropped when not relevant. For example, in a collecting scenario in which at a given time t two objects must be collected, $\Gamma_t = \{collect(O_1), collect(O_2)\}$, with $m(t) = 2$. Each task is composed by a set of roles or operations $\tau_i = \{r_1, \dots, r_k\}$, satisfying the following properties: i) $\forall i, j \quad i \neq j \Rightarrow \tau_i \cap \tau_j = \emptyset$; ii) $|\tau_i^t| = c \quad \forall t \in [t_s, t_e]$. For example, the task $collect(O_1)$, which represents the collection of object O_1 , includes two roles or operations: $grab(O_1)$, which represents the role/operation of grabbing object O_1 , and $support(O_1)$, which represents the role/operation of supporting the grabbing action of object O_1 . We finally define the set of all possible roles at time t as $R_t = \bigcup_{i=1}^{m(t)} \tau_i^t$.

Notice that each role can comprise a set of sub-roles and so on; for the sake of simplicity, we consider only two levels of the possible hierarchy, i.e. tasks which are divided in roles. Hence, for the coordination process, roles can be considered as atomic actions. Each robot has different capabilities for performing each role and different resources available.

Moreover we define for each $r \in R_t$, the set of all roles constrained to r as $C_r \subseteq R_t$. While in general constraints can possibly be of several types (AND, OR, XOR), in this article we focus only on AND constraints. Thus, C_r represents the set of roles that must be executed concurrently by different agents. The properties of each constrained set C_r are: i) $r \in C_r$; ii) $r' \in C_r \rightarrow r \in C_{r'}$. Non-constrained roles are determined by $|C_r| = 1$. For example, in the above mentioned collecting scenario we have $C_{grab(O_1)} = C_{support(O_1)} = \{grab(O_1), support(O_1)\}$, imposing an AND constraint on the two roles of grabbing and supporting an object.

A set of roles C_r subject to AND constraints must be performed simultaneously by $|C_r|$ teammates. For example, if at a given time t there are m objects to be collected then such a situation can be represented with m sets C_{r_1}, \dots, C_{r_m} , where $C_{r_j} = \{grab(O_j), support(O_j)\}$. Notice that if a role r is unconstrained, $C_r = \{r\}$.

We express the capabilities and the resources depending on time with $Cap(e_i, r_j, t)$, $Res(e_i, r_j, t)$, where $Cap(e_i, r_j, t)$ represents the reward for the team when robot e_i performs role r_j at time t , $Res(e_i, r_j, t)$ represents the resources needed by e_i to perform r_j at time t . Finally, $e_i.res(t)$ represents the available resources for e_i at time t . For example, in the previously mentioned collection scenario, if we consider a homogeneous team, capabilities of robots to perform the roles $grab(O)$ or $collect(O)$ depend on time needed by a robot to reach the location of object O . This measure depends on the robot's position which is a function of time, therefore the capability for a robot to perform a role varies with time. Since a robot can accomplish only one role at a time, the resource needed by a robot to accomplish a role $Res(e_i, r_j, t)$ can be considered as a boolean value representing whether the robot is already performing another role at time t . Thus, in this example, the resource of each robot $e_i.res(t)$ corresponds to the current state of the robot: *free* if the robot is not performing any role, or *busy* if the robot is already performing a role.

A dynamic allocation matrix, denoted by A_t , is used to establish the Task Assignment; in A_t , $a_{e_i, r_j, t} = 1$ if the robot e_i is assigned to the task r_j at time t , and 0 otherwise. Consequently, the problem is to find a dynamic allocation matrix that maximizes the following function

$$f(A_t) = \sum_t \sum_i \sum_{r_j \in R_t} Cap(e_i, r_j, t) \times a_{e_i, r_j, t} \quad (1)$$

subject to:

$$\forall t \forall r_j \in R_t \sum_i \sum_{r_k \in C_{r_j}} a_{e_i, r_k, t} = |C_{r_j}| \vee \sum_i \sum_{r_k \in C_{r_j}} a_{e_i, r_k, t} = 0 \quad (2)$$

$$\forall t \forall i \sum_{r_j \in R_t} Res(e_i, r_j, t) \times a_{e_i, r_j, t} \leq e_i.res(t) \quad (3)$$

$$\forall t \forall r_j \in R_t \sum_i a_{e_i, r_j, t} \leq 1 \quad (4)$$

It is important to notice that this problem definition allows for solutions that can oscillate between different allocations that have the same value of $f(A_t)$. Such oscillations can also happen when noisy perception affects computation of the capabilities. This can be avoided by taking into account in the implementation of $Cap(e_i, r_j, t)$ the cost of interrupting a task for switching to another.

III. TOKEN PASSING APPROACH TO TASK ASSIGNMENT

The problem of Task Assignment presented in Section II can be successfully addressed by a Token Passing approach. The main idea of the Token Passing approach is to regulate access to task execution, through the use of tokens. Each token represents a task, and only the agent currently holding the token can execute the corresponding task. Following this approach, the communication needed to guarantee that each task is performed by one agent at time is dramatically reduced.

Token Passing approach to Task Assignment in MAS is a totally distributed method, which can allocate tasks with AND constraints and has been tested in several environments and working conditions. Results show that this method gives good performance, while dramatically reducing the communication overhead [19]. Next in this section we describe the basic process for token management described by Scerri et al. [19].

Tokens represent tasks to be executed and are exchanged through the system in order to collect information and to allocate the tasks to the agents. The basic Token Passing approach assumes coherent knowledge about tasks to be accomplished and thus tokens are univocally associated to tasks when inserted in the system.

When an agent receives a token, it decides whether to perform the task associated to it or to pass the token on to another agent. This decision is taken based only on local information: each agent follows a greedy policy, i.e. it tries to maximize its utility, given the tokens it can currently access, its resource constraints and a broad knowledge on team composition. The ability of the team to assign tasks is related to the computation of the capabilities $Cap(e_i, r_j, t)$. Tasks are executed by the agent that has the corresponding token only if this capability is higher than a given threshold. This threshold can be computed in different ways depending on the scenario. For example, if tasks are a priori known, this threshold can be fixed before inserting the token, or it can be established by the first agent receiving the token based on its local information.

If the capability of the agent is higher than the required threshold, the agent considers the possibility to allocate this task to itself. Otherwise, the agent adds some information about the task in the token and then sends the token to another agent. The token stores the list of agents that have already refused the task, in this way, when an agent passes a token away can choose an agent that has not previously discarded it.

Thresholds guide the search towards good solutions for the allocation problem. While such mechanism cannot give guarantees concerning the optimality of the solutions found, it has been experimentally shown that it can consistently increase the algorithm performance [19]. Under the assumptions that capabilities to perform different tasks are independent and uniformly distributed among agents, thresholds which give best performance for the method can be analytically computed. In particular, when the number of tasks to be executed during the mission remains constant, the best threshold to be used over all the mission execution can be computed in advance, while when the number of tasks to be executed changes consistently, thresholds should be computed according to the current operative situations (e.g. task number, agent number, etc.).

When tasks are constrained, they are composed by roles to be simultaneously executed. In this case, tokens are associated to the roles in the tasks. When considering constrained tasks, assignments based on thresholds on the agent capabilities will lead to potential deadlocks or inefficiencies. For example, consider two roles, r_j and r_k , that need to be simultaneously performed. When a teammember a accepts role r_j , it may reject other roles that it could potentially perform. If there is no teammember currently available to perform role r_k , a will wait and will not be assigned to another role. Thus, an explicit enforcement of the AND constraints among roles is needed.

The general idea is to use potential tokens to represent roles that are tied by AND constraints. Potential tokens retain agents: when an agent receives a potential token, it can perform other roles (i.e. the potential token does not impact on the current resource load of the agent). A manager agent exists for each group of ANDed roles. When

enough agents have been retained for the task execution, the manager agent sends a *lock* message to each of the retained agents. When the *lock* message arrives, the retained agent should start the execution of the role, possibly releasing the current role and sending away the related token. The choice on which role(s) should be stopped is performed based on a greedy local policy. If the role to be stopped is a constrained role, the agent will inform the task manager and the allocation process for that role will be restarted. This mechanism for allocating AND constrained roles has been tested and validated in several domains and operative conditions [19].

To further clarify the token-based assignment, let us consider the following situation: two tasks τ_1, τ_2 and three agents e_1, e_2, e_3 . The task τ_1 comprises one role r_1 while τ_2 comprises two roles r_2 and r_3 tied by an AND constraint. Suppose agent e_2 is handling roles r_2 and r_3 and it is not capable of performing them. Suppose agent e_1 is retained for role r_2 , while no one else is retained for role r_3 . Finally, suppose agent e_1 receives a token for role r_1 and is capable of performing the role. The agent e_1 will thus keep the token and start performing role r_1 . If at this point agent e_3 considers itself retained for role r_3 , it will notify that to agent e_2 (which is the task manager). Agent e_2 will send a lock message to both agent e_1 and agent e_3 . Agent e_3 will start performing role r_3 , and agent e_1 will refute the role r_1 , sending the token to another agent and start executing role r_2 . In this way, the execution of the roles will correctly meet the AND constraint between roles r_2 and r_3 .

IV. DYNAMIC TASK PERCEPTION

In the Task Assignment problem formulated so far the knowledge robots have about tasks that have to be performed has not yet been addressed. However, in MRS, tasks to be performed are often perceived from the environment, through robot perception capabilities. Therefore, a characterization of the task knowledge that robots have is useful for a more precise formulation of coordination in MRS.

Robots have only a local view of the environment, thus we define two sets: $LOT_{i,t}$ (Locally Observed Tasks), which is the set of tasks locally known to robot i at time t , and $GOT_t = \bigcup_i LOT_{i,t}$ (Globally Observed Tasks), as the union of the locally observed tasks from each robot i at time t . The $LOT_{i,t}$ is built by each robot based on its local perception, thus we can write $LOT_{i,t} = Mem(LOT_{i,t-1}, O(e_i, t))$, with $O(e_i, t)$ being a function that models the robot perception and Mem a function that integrates information over time.

More specifically, $O(e_i, t) : E \times Time \rightarrow \mathcal{P}(\Gamma_t)$ returns a set $\{\tau_k^t\}$ of tasks visible to the robot at time t . Furthermore, the Mem function integrates observations over time adding newly discovered tasks and managing tasks in the Locally Observed Tasks set. To this end, for each task, the robot evaluates whether the task is *active*, meaning that it has not been allocated to any robot yet, or not. Active tasks determine the creation of new tokens that are inserted in the system.

Finally, we define $LRS_{i,t} = \{r_j | a_{e_i, r_j, t} = 1\}$ (Local Roles Set) and $GRS_t = \bigcup_i LRS_{i,t}$ (Global Role Set) as the roles currently assigned to robots. The global constraints (4) defining a non conflicting allocation can be then expressed as

$$\forall t \bigcap_i LRS_{i,t} = \emptyset \quad (5)$$

It is important to notice here that assuming that a robot can reliably distinguish between *active* and *non-active* tasks is quite a strong assumption. For example, in our application scenario, objects are similar in shape and color and the distinction can be made only by using the object's absolute position in the environment (that is indeed affected both by perception and localization errors). The *accuracy* in object absolute positioning of the robots determines the minimum distance between two objects (and hence tasks) that the system is able to distinguish.

V. DISTRIBUTED CONFLICT DETECTION AND RESOLUTION

The Token Passing approach presented in Section III is based on the assumption that one token is associated to every task to be executed and that the token is maintained by the agent that performs such a task, or is passed to another agent. This assumption holds when tokens are inserted into the system in a coherent fashion. Under this assumption the algorithm ensures that no conflict arises among agents, (i.e. two agents trying to execute the same

role). However, when tasks are perceived and tokens are generated by agents during mission execution, conflicts on task execution may arise. In fact, several agents may perceive the same task, and an uncontrolled number of tokens can be created, leading too many agents to execute the same role. In the following, we present an approach that ensures that exactly n agents will participate in the same task simultaneously.

To prevent possible conflicts that may arise during mission execution, we have to guarantee that only one token is created for each role. Our main idea is to use broadcast messages only to maintain token coherence among agents. For example, when an agent perceives an object, it records this information in a local structure and broadcasts the presence of the object to all its teammates. Whenever an agent accomplishes a task, it broadcasts to the entire team the task termination, and each of the team members removes the tokens referring to the accomplished task from its local structures.

By the use of this approach conflicting tokens can still be created for three main reasons: i) *Simultaneous task discovery*: two agents e_1 and e_2 perceive a new task τ , creating a set of tokens $Tk(t,1) \dots Tk(t,s)$ exactly at the same time, so that both agents will have different tokens referring to the same task. ii) *Message asynchrony*: messages are not guaranteed to arrive in a predefined order; if a robot observes a new object, it creates a new token for a specified role and decides to pass the token on. If the token reaches a team member before the broadcast message which signals the new task discovery, the team member can decide to perform the role possibly conflicting with other teammates. iii) *Errors in active object detection*: if the observation function fails in the recognition of an active task, a team member can decide to allocate itself to a role that is already being carried on by someone else.

Algorithm 1: Tokens Coherence Maintenance

ONPERCREIVED($task$)

- (1) **if** ($task \notin KTS$)
- (2) $KTS = KTS \cup task$
- (3) $annMsgS = annMsgS \cup \{task, MyId\}$
- (4) $TkS = TkS \cup Tk(task)^1 \cup \dots \cup Tk(task)^s$
- (5) BCASTSEND(msg(Announce, $task$))

ONTASKACCOMPLISHMENT($task$)

- (1) $ATS = ATS \cup task$
- (2) BCASTSEND(msg(AccomplishedTask, $task$))

MSGANALYSIS(msg)

- (1) **if** msg is AccomplishedTask
- (2) $ATS = ATS \cup msg.task$
- (3) **if** msg is Announce
- (4) **if** ($msg.task \notin KTS$)
- (5) $KTS = KTS \cup \{msg.task\}$
- (6) $annMsgS = annMsgS \cup \{msg.task, msg.senderId\}$
- (7) **else**
- (8) $AnnItem = GETANNOUNCE(Msg.Task)$
- (9) **if** $AnnItem.senderId \leq msg.senderId$
- (10) $ITS = ITS \cup \{AnnItem.task, AnnItem.agentId\}$
- (11) UPDATE(AnnMsgS, msg)
- (12) **else**
- (13) $ITS = ITS \cup \{msg.task, msg.senderId\}$

In the following, we describe our approach to avoid conflicts during mission execution and further details of the Token Passing process. Simultaneous task perception and message asynchrony are addressed using a distributed approach for conflict detection and a global policy that gives a preference criterion among all agents, the simplest one being a fixed priority. The pseudo-code for the distributed conflict detection approach is reported in Algorithm 1. This algorithm uses the following data structures: i) Known Task Set (KTS), which contains at each time step all the tasks that have been perceived by all agents; ii) Accomplished Task Set (ATS), which contains at each time step all the tasks that have been accomplished by all the agents; iii) Invalid Task Set (ITS), which contains at each time step the tasks that the agent considers invalid along with the information on the agent that created the task; iv) Announced Message Set (annMsgS), which is a set of announced messages received. annMsgS is updated in order to store at each time step and for each announced task the announce message received by the highest priority teammate, and is used to decide whether an announced task should be considered invalid; v) Token Set (TkS), which is the set of tokens that the agent currently holds. Each of these data structures is local to an agent. Messages sent among agents have four fields: (1) *type* denotes the type of the message; (2) *task* contains information about the perceived task (e.g. object position), which is valid when *type* is *announce* or *accomplishedTask*; (3) *token* (valid only when the message is a token) contains information about the token (e.g. task type, identification number, visited agents etc.), (4) *senderId* is an identifier for the robot that sent the message.

Whenever a new perception is received, a broadcast message for the task discovery is stored in the *annMsgS* (procedure *OnPercReceived*, line 3) and then is sent to all teammates (line 5). Whenever a task is accomplished, an accomplished task message is sent broadcast (procedure *OnTaskAccomplishment*, line 2). In the *MsgAnalysis* procedure, if a received message is an *AccomplishedTask* message, the agent adds the task to its *ATS*; if the message is an *Announce* message, the agent checks whether the task has been already announced by checking its *KTS* (line 4). If the task is not present in its *KTS*, it adds the task in the *KTS* and inserts the corresponding announce message in its *annMsgS*; if the task was already present in the *KTS*, the agent detects a conflict Using *annMsgS* (procedure *MsgAnalysis*, line 8) it checks whether the invalid task is the new announced task or the one previously received and, consequently, updates the *annMsgS* and the *ITS*. Each robot periodically removes all tasks which are present in the *ATS* and in the *ITS* from the tokens it currently holds.

An example of execution for our distributed conflict detection method is reported in Figure 1. Numbers on arrows correspond to time steps at which messages are received by agents. The square represents an object, and arrows starting from the square represent object perception. For each data structure, in addition to the information it contains, we report the time step at which the information is inserted, for clarity of explanation. In the situation represented in Figure 1 Object *O* is correctly perceived as an active task at the same time (time step 1) by agent *A1* and agent *A2*. Both agents insert the task in their *KTS* and the corresponding message in their *annMsgS*, which contains the task created for the object and their respective identifiers. They both send an announce message in broadcast. Let us assume that the announce message of agent *A2* is delayed for two time steps and reaches its destination at time step 4. Meanwhile, the announce message of agent *A1* reaches agent *A3* and agent *A2* at time step 2. Agent *A3* does not have task $T(o)$ in its *KTS*. Thus, it adds the task to its *KTS* and the announce message $\langle T(o), A1 \rangle$ to its *annMsgS*. Agent *A2* already has task $T(o)$ in its *KTS*. Thus it detects the conflict and, since the priority of agent *A1* is lower, it inserts $\langle T(o), A1 \rangle$ inside its *ITS* and leaves its *annMsgS* unchanged. At time step 4 the announce message $announce(T(o), A2)$ reaches agent *A1* and agent *A3*; both the agents identify the conflict and, consequently, update their *annMsgS* and their *ITS*. All agents correctly converge toward the same *ITS* without the need of any additional communication.

A. Avoiding failures in active task recognition

The possibility of failures in active task recognition is clarified by the following example. Suppose a robot e_i is allocated to a role $grab(O_i)$ and it performs the transporting phase of the role. As a consequence of the action of robot e_i , the position information of object O_i will change; therefore a second robot e_j , which detects the object

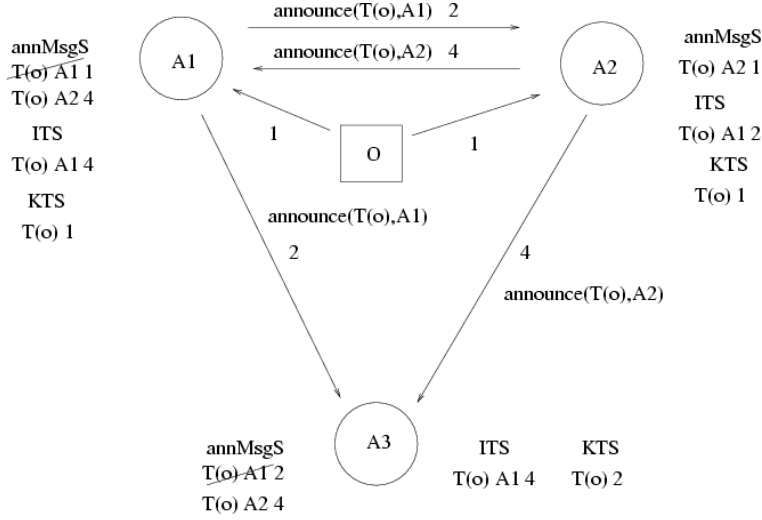


Fig. 1. Distributed conflict detection

O_i in a new position, could incorrectly conclude that the object it is observing is a new object O'_i and start the allocation of a new task referring to this object. This will clearly lead to a conflict in the allocation process, which will result in conflicts between two team members trying to grab an object, which is already being transported by e_i . We refer to this problem as false *active* task detection (see section II).

Notice that this problem can be partially avoided, by tracking the objects that robots perceive. Techniques to address the problem of distributed multi-object tracking have been successfully applied in MRS (e.g., [21], [22]). However, such solutions generally require a central unit to process data coming from the robotic agents, while, for our MRS application domain, centralization of all information is not feasible; in fact, a centralized approach would require a prohibitive amount of communication even for relatively small size teams; moreover, centralization would introduce a single point of failure, thus severely limiting the robustness of the overall system. Finally, the tracking algorithm may fail for particular configurations of the working environment (e.g. due to possible occlusions of robot sensors), and situations like the one discussed above may still arise.

Conflicts arising from this task recognition failures are very hard to solve at low-level. In fact, the process of understanding that a teammate carries a detected object requires sophisticated perception and localization capabilities, which cannot be taken for granted. In particular, to address such situations, robots should not only be able to recognize their teammates, but should also be able to monitor the teammates' actions to understand whether a robot near an object actually transports the object or simply passes by the object.

On the other hand, our approach to Task Assignment can successfully address such conflicting situations. The solution we propose involves the possibility for a collector robot to detect and invalidate tasks incorrectly created by teammates. Let us consider again the above described scenario. When robot e_j starts the allocation of a new task for object O'_i , following our strategy, it will announce the creation of the task to its teammates (through a broadcast message). Eventually, robot e_i will receive the announce message for the creation of a new task for the object O'_i . e_i can detect that the new task is actually being performed by itself. By comparing the position information of the object it is currently transporting O_i and object O'_i , and declares the new created task as invalid. e_i will then announce the task invalidation to all its teammates sending an *InvalidTask* message.

As already mentioned, the ability of correctly solving all the conflicts is obviously related to perception and localization accuracy of the robots. When all the objects (tasks) are sufficiently distant from each other (i.e. the distance between them is greater than the detection accuracy), the system always produces correct solutions. There are also cases in which the system is still able to determine a correct allocation although the objects are actually

closer than the perception accuracy. For example, if two objects are too close, the system will first create a single task associated to one of them, ignoring the other. When the first object has been collected, subsequent observations of the second object will start the creation of another task. However, correct task execution is not guaranteed in general, e.g. when the system is not able to correctly invalidate conflicting tasks.

B. Discussion

We now show that the conflict resolution method proposed in this article ensures conflict free allocations of agents to roles. More precisely, we want to ensure that, at each time step, the intersection of the Local Role Sets of all agents is empty as stated in property 5. In fact, the main focus of this work, with respect to previous approaches using Token Passing [19], is the detection and resolution of conflicts in the allocation process.

The first step in order to detect conflicts is to identify tasks unambiguously. In general, we identify tasks through a set of properties. We distinguish two situations: in the first one, all the properties that we use to identify an object do not vary over time. In this case, each robot has the required knowledge to correctly identify objects during the mission execution. In the second situation, a subset of the properties of an object may vary because an agent operates on that object. In this case, only the agent that is operating on the object is able to perform a correct identification for that object.

We will assume perfect and instantaneous communication in order to prove correctness and then we will show that if there is an upper bound for network delays, which depends on the application domain, we are able to maintain a weaker form of correctness.

Theorem 1 *Algorithm 1 provides solutions which satisfy (5) under the following assumptions: i) communication is perfect and instantaneous, ii) agents can correctly distinguish the perceived properties of different objects.*

Proof: (by contradiction) Equation (5) states that $\forall t \bigcap_i LRS_{i,t} = \emptyset$. Let us assume that (5) is violated for a task. This means that $\exists \bar{t} \bigcap_i LRS_{i,\bar{t}} \neq \emptyset$ and thus there would be more than a robot assigned to a role for a task at time \bar{t} .

This implies that at least two tokens for the same role have been produced and thus the same task has been perceived and announced by two different robots. Let us consider that the two robots i and j ($i \neq j$) perceived the same task τ_k at times t_1 and t_2 where $t_1, t_2 \leq \bar{t}$. Moreover, let us consider that the task τ_k comprises only one role r_k :

$$\exists \bar{t}, i, j \mid LRS_{i,\bar{t}} \cap LRS_{j,\bar{t}} \neq \emptyset \Rightarrow \quad (6)$$

$$\exists r_k \mid r_k \in LRS_{i,\bar{t}} \wedge r_k \in LRS_{j,\bar{t}} \Rightarrow \quad (7)$$

$$Tk(\tau_k) \in TkS_{i,\bar{t}} \wedge Tk(\tau_k) \in TkS_{j,\bar{t}} \Rightarrow \quad (8)$$

$$\exists t_1, t_2 < \bar{t} \mid O(e_i, t_1) = \tau_k^{t_1} \wedge \forall t \in [t_1, \bar{t}] (\tau_k^t, i) \notin ITS_{t,i} \wedge O(e_j, t_2) = \tau_k^{t_2} \wedge \forall t \in [t_2, \bar{t}] (\tau_k^t, j) \notin ITS_{t,j} \quad (9)$$

We can distinguish two cases:

- 1) $t_1 < t_2$ (or $t_1 > t_2$)

For this case we have two sub-cases:

- All the properties of the perceived task τ_k remain constant from t_1 to t_2 . The robot which perceives the task at time t_2 , i.e $O(e_j, t_2) = \tau_k^{t_2}$, would have already received the announce message for it at time t_1 . In this case, it could not have announced the task (and generated the role tokens) because it would have already been notified through an announcement. In particular, the announcement from e_i of τ_k implies that:

$$\forall t > t_1 \tau_k \in KTS_{j,t} \Rightarrow \quad (10)$$

$$Tk(\tau_k) \notin TkS_{j,t} \quad (11)$$

Since 11 contradicts 8, the assumption is refuted.

- Some of the properties of task τ_k changed from t_1 to t_2 . This implies that there exists a robot that operated on the object related to this task. Let i denote this robot. At time t_2 robot j perceives the object that robot i is operating on as a new task $\tau_k^{t_2}$. Therefore, robot j will broadcast the announce of the creation of the new task $\tau_k^{t_2}$ at time t_2 . All robots will receive the announce message, therefore robot i will recognize that the announced object is the one it is acting upon. Robot i will perform the correct association considering how its behavior has affected the properties of task $\tau_k^{t_1}$. Robot i will send an invalid task message for $(\tau_k^{t_2}, j)$ in broadcast at the same time t_2 . All robots will receive the invalid task message at time t_2 , and therefore $\forall t \geq t_2 (\tau_k^t, j) \in ITS_{j,t}$, which contradicts 9, and therefore the assumption.
- 2) $t_1 = t_2 = t$. The robots will both announce the task τ_k and both receive the announcement of the other agent. In this case, the robot that has a lower priority would not have generated the tokens relative to the roles for accomplishing the task.

More formally:

$$i \text{ sends a broadcast announce message for } \tau_k \text{ at time } t \wedge \quad (12)$$

$$j \text{ sends a broadcast announce message for } \tau_k \text{ at time } t \wedge \quad (13)$$

$$ORD(i) > ORD(j) \Rightarrow \quad (14)$$

$$(\tau_k, j) \in ITS_{t,j} \text{ (MsgAnalysis, Line 10)} \wedge (\tau_k, j) \in ITS_{t,i} \text{ (MsgAnalysis, Line 13)} \quad (15)$$

where $ORD(i) > ORD(j)$ indicates that i precedes j in the global robot ordering. Since 15 contradicts 9, the assumption is refuted. ■

In real robot scenarios network delays occur. In this case, there will be a time interval Δt during which property 5 will be violated. However, after Δt our approach is able to detect and solve the conflict.

Lemma 1 *In presence of network delays bounded to a given time interval Δt_b , Algorithm 1 guarantees that each conflict is solved in a finite amount of time. More formally:*

$$if \exists t, i, j, k \mid r_k \in LRS_{t,i} \cap LRS_{t,j} \text{ then } \exists \Delta t \mid \forall t' \geq t + \Delta t \ r_k \notin LRS_{t',i} \cap LRS_{t',j} \quad (16)$$

with $\Delta t_b \leq \Delta t$.

The proof for this lemma closely follows the line of Theorem 1 and thus we do not report it here. The above lemma states that if a conflict occurs for a given task, at a given time, this conflict will be eventually detected and solved after a time interval that is at most Δt . Notice that Δt is dependent on the network delay and on the type of conflict. If the conflict to be solved requires a task invalidation message, the delay will be higher since more communication is required. The network delay is required to be bounded by a given time interval. This is needed because properties of objects can change, thus the communication delays should be small enough to allow agents to correctly identify objects when properties change. The time interval Δt_b depends on the particular application domain and is related to the minimum time interval needed to change the object properties. The higher it is such time interval, the higher can be Δt_b .

Definition 1 *A method for conflict detection is weakly correct for a given domain if conflicts do not affect correctness of task execution.*

Definition 2 *The recovery time $t_r(r)$ for a role $r \in R$ is the maximum time before the execution of a role r modifies the properties of the tasks in the environment.*

Given the above definitions and the lemma, we can formulate the following theorem:

Theorem 2 For a given domain if $\forall r \in R \Delta t_b \leq \Delta t < t_r(r)$, then Algorithm 1 is weakly correct.

Proof: The proof of this theorem directly follows from Lemma 1. ■

The presented proofs show that, under the specified assumptions, our approach provides correct allocation, because it detects and solves all possible conflicting tokens. Moreover, the algorithm has the property that at least one token is created for each role and only one token remains in the system (after conflicts have been solved) until the task is completed. Therefore, conflicts are actually solved by removing from the system only those tokens that are considered responsible for the conflict, leaving exactly one token for each role. The above results rely on the assumption that messages cannot be lost. This is a strong assumption which cannot be given for granted in several realistic scenarios. However, since our approach is specifically designed to minimize the communication overhead, we can afford the use of a reliable protocol (such as TCP or UDP with acknowledgment) for message exchange. Another important issue is the possibility of robot loss due to temporary robot disconnections or to permanent robot failures. Currently, the above results do not hold in case of robot loss, however, if the communication infrastructure could give a signal of possible robot loss (e.g. a threshold on the message re-transmission number), the algorithm could be extended to take such situations into account.

As previously mentioned, detected conflicts are solved by using a static fixed priority defined among agents. Notice that any policy that gives a global ordering of team members, and that does not require further communication, can be used in place of fixed priority as a global preference criterion. Another option could be to invalidate tasks, which have been created more recently. This, however, would require to have a synchronized clock among agents. In any case, setting a static fixed priority among agents can obviously result in non-optimal behavior of the team; for example, assuming that $Cap(e_1, r_k, t) > Cap(e_2, r_k, t)$ following a static priority based on id, we yield to the less capable agent the access to the task r_k . While in principle the difference among capabilities can be unbounded, in practice, when tasks are discovered using perception capabilities, agents perceive tasks when they are close to the object location, (e.g. if two robots perceive the same object their distance from the object is comparable). Therefore, the loss of performance due to the use of a fixed priority is limited.

VI. EXPERIMENTS AND RESULTS

A. Implementation on AIBO robots

We implemented and tested the described method on the Sony AIBO robots¹. Our application scenario is formed by a set of robots that need to perform a synchronized operation on a set of similar objects scattered in the environment. In particular, the robots have to collect a set of identically colored balls. Each ball, to be correctly transported, requires one robot to grab it, by blocking it with its head (*collector robot*).

However, since one robot is not completely reliable in the grabbing phase, we require a second robot (*supporting robot*) to help in the grabbing phase by pushing the ball onto the chest of the collector. Once the ball has been grabbed, the supporting robot moves away from the ball and is ready to be allocated to a different task (i.e. collecting another ball), while the collector robot can transport the grabbed ball to the target area.

Each robot knows its teammates and is able to communicate with them through a wireless device. However, robots know neither the number of objects scattered in the environment, nor their initial positions.

In order to successfully operate in this scenario, four main components are required: 1) self-localization, 2) path planning and obstacle avoidance, 3) object recognition and 4) action execution and synchronization. To put emphasis on coordination issues, we have used an engineered environment, borrowed from the RoboCup Legged League², which is a rectangular field with a set of landmarks in known positions, where every landmark is clearly distinguishable by its color. In this way, it is possible to implement an effective landmark-based localization method (e.g., [23], [24]).

¹See website <http://www.dis.uniroma1.it/~farinell/video/CoopForaging-commentary.wmv> for a video of the experiment.

²See website <http://www.openr.org/robocup>

In order to minimize interference between robot movements, we implemented a hybrid path-planner. At a high level we have a wavefront path planner. The environment is represented as a grid, where there is a goal cell and a set of obstacle cells. These obstacles are set as the elements of the KTS (see section V) in order to avoid collisions with balls that have not been collected yet. Teammates are not modeled at this level for two main reasons: i) the difficulties of evaluating the position of other robots by using the robot sensors, ii) the need to minimize communication overhead (maintaining information on teammate positions requires additional communication).

The high level path planner thus determines the next cell to move to. Then a reactive behavior allows for the robot to reach the target cell and deals with the possible presence of close robots detected with the on-board proximity laser sensors.

The identification of the objects to collect is again based on colors, but since there are many objects with the same color in the environment, also their absolute position in the field must be devised. Note that this information is subject to errors due to noisy perception and inaccurate self-localization, which can lead to false positives in task detection.

To avoid this problem, we filter the absolute position of objects based on their relative distance, thus reducing the probability of false positives. Moreover, we use active perception to reduce errors in object perception: while moving to the expected position, the robot interleaves object tracking with landmark pointing for improving self-localization: in this way, it can better determine if it is the one associated with its task.

More specifically, assuming that both localization error and perception errors are bounded, then also the global object positioning error is bounded. Therefore, we adopt a simple temporal filter, that averages over time global object positions acquired by the robots, thus increasing robustness to noise. This ensures that, if each pair of objects are at a distance greater than the typical global positioning errors, then there will be no errors in the task assignment due to noisy perception and localization. In our experiments we have placed balls to be collected at least 0.5 meter away from each other. However, it is worth reminding that objects are not static, but are moved by the collector robots. This may cause additional errors in object positioning and consequently additional possible conflicts, even when perception errors are bounded.

Although the above described approach makes the system more robust to false positives, they cannot be eliminated. In presence of false positives in task detection ad hoc solutions must be designed. Invalidating tasks when a robot reaches a target and the object is not there may help; however, this may lead to invalidate correct tasks in cases of localization errors. In our implementation we invalidate tasks only for a limited amount of time. In this way, if a task is incorrectly invalidated, it will be eventually accomplished. In any case, we cannot completely avoid perception errors to determine mission failures. In our experiments mission failures due to large and continuous perception errors rarely occurred.

Finally, action execution is accomplished through a reactive module in charge of controlling the execution of pre-defined plans, identifying action failures during their execution, and activating corresponding recovery procedures. Action synchronization has been realized through explicit communication, by exchanging synchronization information among robots.

With the basic features described above, we have been able to implement and test the coordination algorithm presented in this article. Such an implementation on AIBO robots has shown the feasibility of the approach on robots with very limited resources that affect perception and localization capabilities.

We run two sets of experiments, all of them have been performed under the same initial configuration, but in the second set we forced conflicts, due to simultaneous task discovery, by simulating network delays. We use the same initial configuration to study the behavior of the systems when conflicts arise. Results from these experiments are reported in Table I.

In particular, we have a low communication overhead in the former set where no conflicts are generated. In fact, we have a constant number of 9 broadcast messages and an average of 83.7 point to point messages. Finally, each task is accomplished in an average time of about 70 sec.

In the latter set where we forced a fixed number of conflicts, similar results were found. In particular, there were constantly 15 broadcast messages and an average of 118.2 point to point messages for each experiment.

	Peer to Peer Msg	BroadCast Msg	Conflicts	Avg Time for task execution
First Set	83.7	9	0	68.09
Second Set	118.2	15	6	67.93

TABLE I

DATA EXTRACTED FROM TEN EXPERIMENTS ON AIBO ROBOTS

Extra messages in this case were used to detect and solve conflicts. Since conflicts are detected and solved before invalid tasks are started, the conflicts do not significantly impact the overall performance of the system.



Fig. 2. Our experimental scenario

B. Experiments in a simulated environment

To perform experiments with a larger team of robots and to have a larger data set to analyze, we evaluated our approach also in a simulated environment. To this end, we have used a general robot development framework for realizing the robotic agents [25] and implemented an environment manager to manage interaction between robots and objects and among robots. This framework provides an infrastructure for developing *modules* that are functional components and for connecting them to *drivers* that are interfaces to hardware components. The modularity of the framework allows for connecting functional modules to either drivers operating real hardware devices or drivers simulating them, while maintaining the same architectural connection of modules and drivers. More specifically, we have realized (or reused) the main functional modules for perception, localization, navigation, action execution, planning and coordination and connected them to drivers simulating vision, network communication a non-holonomic robotic platform and a simple grabbing device. The environment manager is in charge of handling interactions between robots and objects and among robots in the environment. It provides each robot with information about the position of objects and other robots, according to its perception capabilities. Moreover, it receives information on robot positions and actions to manage interaction with the objects (e.g. grabbing of the objects, bumping of objects into other objects or robots into each other). This process is connected only with the specific drivers, thus being transparent to all the functional modules. In fact, the modularity of the framework allows for directly using some of the developed modules on the robotic platforms.

Finally, a graphical application is used to display the status of the robots and of the environment. This setting provides us with all the necessary features that are required to validate the approach and evaluate the system performance.

C. Performance Evaluation

For a quantitative evaluation of our approach, we have identified a set of initial configurations varying the number of robots involved in the foraging task.

We let the system run until all the tasks have been accomplished and then we measure the time needed to accomplish the task and the number of exchanged messages. Note that, the simulation system introduces perception errors that lead to conflicts as discussed in Section III. Moreover, in the experimental setting we enforce the AND constraint in the grabbing phase, ensuring that each grabbing task is performed by exactly two robots which correctly synchronize their actions. Therefore, each task can be accomplished only if conflicts are correctly solved and robots synchronize their actions. All performed experiments have terminated, thus showing that the algorithm can successfully manage conflicts and that AND constrained roles do not cause deadlocks, always converging to a valid allocation. Moreover, we verified the statistical relevance of our results, using the Student's t -Test [26].

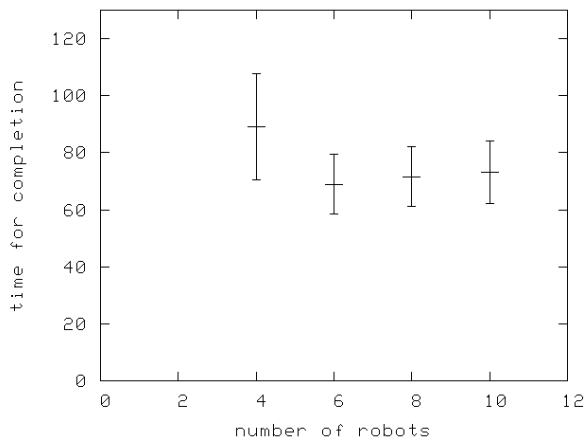


Fig. 3. Average time for completion measured in second, x-axis is number of robots; results averaged over 20 simulations

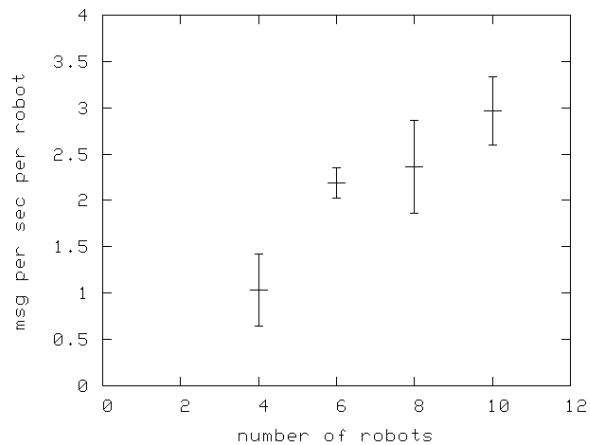


Fig. 4. Messages exchanged per second per robot, x-axis is number of robots; results averaged over 20 simulations

We performed two classes of experiments. In the first class we varied the number of robots with a fixed number of tasks and environment dimension. In the second class we varied the number of robots and number of tasks keeping the ratio tasks/robots constant.

D. Results with a fixed number of objects

In Figures 3 and 4 we report the results obtained by varying the number of robots for a fixed number of tasks. We varied the number of robots from 4 to 10 and fixed the number of objects to be collected to 3. In the experiments all robots can perceive all objects, thus generating the highest number of possible conflicts. Figure 3 shows interesting behavior with respect to time for completion: the time needed by robots to accomplish all tasks decreases when the number of robots increases, but when too many robots are present in the working environment, performance slightly degrades due to spatial conflicts. The results suggest that there is an optimal number of robots to accomplish the task beyond which, adding more robots to the system decreases the overall performance, if the number of tasks and the size of the working environment remain fixed. For this particular situation the Student's t -Test shows that the probability that the two distributions derive from different experimental configurations is very high when comparing 4 robot case with the other combinations, but it decreases significantly when comparing 6 robots with 8 and 8 with 10. This confirms the intuition that adding more robots to a team of 6 robots does not impact the performance in terms of average time for completion.

Figure 4 reports the number of messages exchanged per robot per second. The number of messages increases with respect to the team size for two main factors: i) because of a higher number of conflicts; ii) because of the presence of cycles of not allocated tokens. In fact, when tokens are refused by all the robots, they wait a given amount of time and then they are reinserted into the system.

However, results show that the number of messages exchanged per second is very small. Consider that coordination methods based on iterative Task Assignment (e.g., [16], [17]) for a similar application with n robots and m tasks would require nm messages for each robot at each time step (i.e., for ten robots and three tasks it would be 30 messages per robot per time step, with respect to 3 needed by our method).

E. Results with varying object number

Figure 5 and 6 report results obtained varying the number of tasks. We kept a constant ratio between the number of robots and the number of objects present in the environment, scaling the environmental size accordingly. In particular, we use n objects and $2n$ robots, varying n from 3 to 6. The working environment is a square with a side of $1000*n$ mm and the collect area is a square with a side of $100*n$ mm located in the center of the working field. Objects and robots are scattered in the environment with a uniform random distribution.

The main goal of this experiment is to study how the message load and time for completion change when the system scales in size.

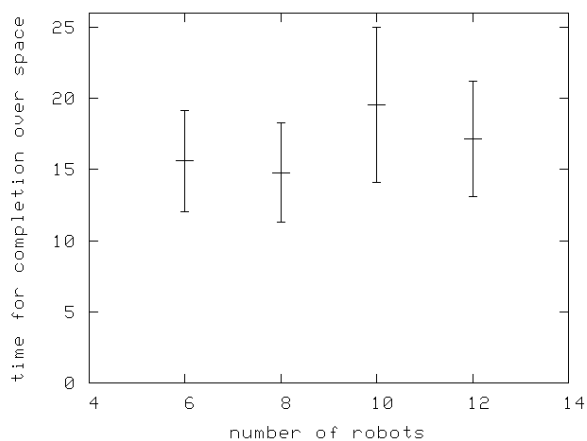


Fig. 5. Average time for completion divided by space, x-axis is number of robots; results averaged over 20 simulations

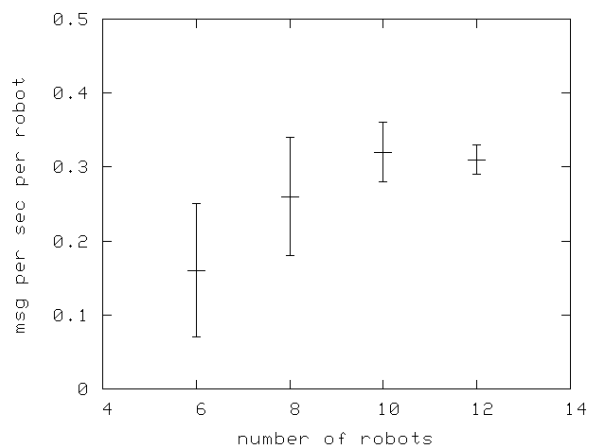


Fig. 6. Messages exchanged per second per robot, x-axis is number of robots; results averaged over 20 simulations

In Figure 5 we report the number of robots on the x axis and the average time for completion divided by the side of the working region on the y axis . We consider such a performance measure to discount the extra amount of time needed by robots to travel from their positions to the objects, which clearly increases with the environment size. Results show that our performance measure is bounded for all the considered situations, suggesting that such measure does not increase scaling the system size (i.e., number of tasks and number of robots). This shows that our approach is able to scale with the system size, ensuring a correct and quick allocation from small up to medium size teams.

In Figure 6 we report all messages exchanged by the coordination algorithm for our experimental scenario. In this case, we measured the number of messages exchanged for each robot and for each task divided by the time needed to accomplish all tasks. In a Token Passing approach without broadcast communication, such a measure would have been constant with respect to the number of robots. In fact, once all tokens are inserted into the system, at each time step each robot sends a point to point message for all unallocated tokens it holds. The behavior shown

in Figure 6 characterizes the ability of the system to send broadcast messages only when needed. The results are comparable to the ones obtained with market based approaches (e.g., [10], [27]) and significantly better than the ones based on iterative broadcast communication (e.g., [16], [17]). This is explained by the fact that our approach sends broadcast messages only for solving conflicts, market based approaches use broadcast for the bidding process at each task assignment, while the iterative methods send broadcast at every cycle.

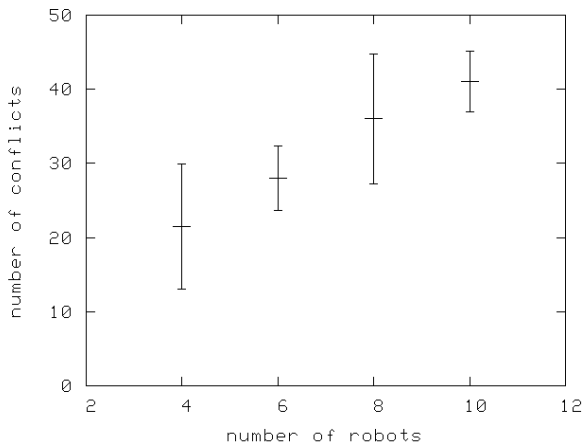


Fig. 7. Number of confl with fix task number

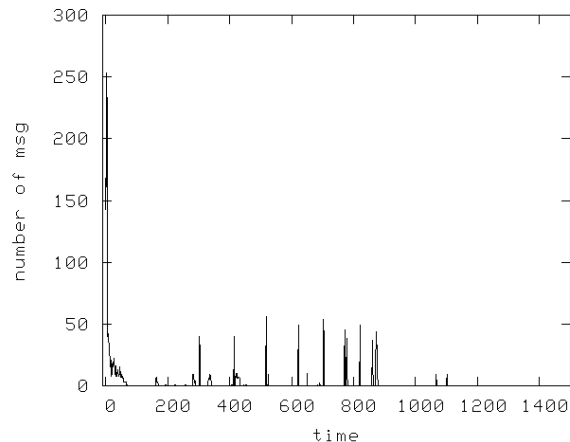


Fig. 8. Bandwidth variation with time.

Figure 7 reports the number of conflicts detected and solved by each robot during the simulations. The measure includes conflicts which arise from simultaneous perception and from incorrect active task perceptions. As shown in the figure, for all situations we detect and solve a relevant number of conflicts. Results reported here refer to the situations with a fixed number of objects and varying robots. The number of conflicts increases with the number of robots. This is because when the number of robots increases, there is a higher chance for simultaneous task detection and incorrect active task perception. Similar behavior is obtained varying the object number.

Finally, in Figure 8 we report how the communication bandwidth used by the coordination algorithm changes with time. The x-axis reports the simulation cycles, while the y-axis reports the number of total messages present in the communication channel for each simulation cycle. The communication channel presents a burst of messages for few simulation cycles and is almost empty for most of the others. In particular, there is a big burst of messages in the first cycles due to the several tasks which are simultaneously perceived by all robots. Once conflicts are solved, the used bandwidth dramatically decreases. Such results show that our approach is able to dynamically use the communication channel, requiring bandwidth only when it is needed for the coordination algorithm.

VII. RELATED WORK

In order to put the proposed approach in a proper perspective with respect to the literature on Task Assignment and cooperation in MRS, it is worth recalling that cooperation based on Task Assignment can be typically considered as a *strongly coordinated/distributed* approach [1]. Such form of coordination is based on explicit communication and the system is distributed, without any kind of static or dynamic leader deciding the assignment of tasks. However, there are no assumptions on the size and composition of the team: namely robots can be heterogeneous and the proposed technique is expected to scale up with the number of robots. In addition, Task Assignment, as proposed in the present article, can be viewed as a *socially deliberative* technique for cooperation in MRS since the agents explicitly interact (exchange information) in order to identify the most effective team organization. It is therefore significantly different from the *Reactive Task Assignment* (e.g., [11]), where each member of the team decides whether to engage itself in a task, without (re)-organizing the other member activities.

Reactive approaches drastically reduce the requirements on communication; they seem, however, inherently limited in terms of the level of cooperation that they can support.

Among the approaches that are much more directly related to socially deliberative Task Assignment in MRS, as proposed in the present work, we consider *Iterative Task Assignment* (e.g., [16], [17]). Iterative approaches allocate all tasks present in the system at each time step. In this way, the system can adapt to environmental conditions ensuring a robust allocation. However, such approaches generally require to know in advance the tasks which have to be allocated, while a main features of our approach is to dynamically generate tasks based on robot perceptions. Moreover, such approaches make use of broadcast messages for each task allocation, thus requiring a large communication bandwidth.

Sequential Task Assignment methods (e.g., [14], [15], [28]), are more closely related to our approach. Such methods allocate tasks to robots sequentially as they enter the system, thus tasks to be allocated are not required to be known before the allocation process begins. In this category we include market-based approaches, which have been recently successfully applied to MRS task assignment (e.g., [10], [27]). Such techniques suffer in general from a large requirement in terms of bandwidth due to the back and forth of messages exchanged in order to assign tasks. One possibility to overcome this problem is through the hierarchical organization of teams. Nonetheless, a Token Passing approach, which substantially limits the need for broadcast communication, seems to be more scalable and flexible in terms of communication requirements. As already noticed, in general, the above cited approaches to Task Assignment do not specifically address the problem of dynamic generation of tasks as well as the possibility of conflicts arising from dynamic task perception.

The work by Zlot et al. [10] presents a team of robots involved in an exploration task. Coordination is achieved by using a market-based approach. Robots exchange points of interest to be visited in order to build the map of the environment more effectively. Since the exchange of tasks is carried out through an auction mechanism based on broadcast communications, robots are able to detect whether a given interest point is allocated to more than one robot. In this respect the approach is similar to our conflict detection mechanism, but in our work we also consider conflicts arising from properties that change over time during task execution (e.g. identical moving objects) and provide a low-bandwidth approach to conflict resolution. Hence, we made a quantitative analysis of conflicts occurring both in the simulated environment and on real robots, with respect to the number of messages needed to address such situations. This approach is robust to robot loss and temporary communication break down, while these issues are not considered in this article. However, the problem of considering at the same time a complete conflict-free allocation and robustness to robot loss can only be achieved by using redundant messages, thus affecting network performance.

Hybrid solutions which merge characteristics of different types of task allocation have been investigated. For example, an emotion-based approach has been proposed for multi robot recruitment [29]. Such an approach can be considered intermediate between sequential and reactive task assignment. With respect to our approach such method seems to require a larger bandwidth due to extensive use of broadcasts message. Moreover, as previous approaches, this work does not explicitly take into account conflicts due to dynamic task perception.

Conflicts arising in Task Assignment are specifically addressed in other works [30], [31]. However, conflicts described in those works are only related to the use of shared resources (i.e space), while our approach can address a more general class of conflicts, such as the ones that arise when task properties change over time.

As for MAS an interesting approach to conflict detection is presented by Scerri et al. [32]. This work is specifically targeted toward large scale MAS. Agents are dynamically organized in sub-teams and share information only through point to point messages; task allocation is performed by using a token-based approach. Conflicts in this setting can be revealed and solved if overlaps among sub-teams exist. Authors show that for such large scale teams (i.e. 200 agents) chances of having overlaps among sub-teams is high, and thus conflicts can be solved most of the time. As previously mentioned, in our setting we are interested in providing guarantees of conflict free allocation, therefore we use broadcast communication to detect and solve conflicts.

Summarizing, the Token Passing approach chosen as the basis of the present work is suitable to address several issues highlighted in the above discussion. However, a possible weakness of Token Passing can be identified in the

non-optimality of the proposed solutions. There appears to be a trade-off between optimality and communication requirements. In this respect, Token Passing can be regarded as a good compromise between the need for communication and the optimality of the solution. In fact, methods based on Iterative Task Assignment guarantee an allocation based on constantly updated information, but require a consistent communication overhead to maintain synchronized agent knowledge. In particular, Token Passing has been tested and evaluated in a large scale team of heterogenous agents, with strict constraints on communication and where time needed to provide a solution to the allocation problem plays a central role [19].

Market-based approaches represent a more continuous spectrum of solutions to address the trade-off between optimality and communication load, but require, in general, a higher amount of messages to synchronize agent activities. The Token Passing approach, on the other hand, does not need any synchronization among robots in the task allocation process, thus allowing for a fully distributed system design.

In addition, one should consider that the dynamic re-assignment of tasks must take into account not only the optimality of the assignment, but also other aspects such as the need to avoid oscillations between nearly optimal solutions, which would substantially change the behavior of the individual robots; such a situation would lead the system to waste much time in selecting an assignment, while little progress is made toward the completion of the task. In other words, switching to a potentially superior assignment may not be effective if one takes into account the cost of switching. Moreover, the inaccuracies of perception may lead to repeated changes from one assignment to another, thus preventing the system from completing a task. This issue can be explicitly addressed through the use of hysteresis to regulate the task switching procedure [16]. Hysteresis, however, has to be finely tuned to reach good performance of the system, while approaches based on Sequential Task Assignment and on Token Passing can overcome the problem with an accurate design of the capability functions.

VIII. CONCLUSIONS AND FUTURE WORK

In this article we have presented a distributed algorithm for Task Assignment in dynamic environments. The presented approach is based on Token Passing to role allocation and successfully achieves the integration of Task Assignment with dynamic object perception from the environment. The approach can detect and solve conflicts in role execution and provide correct allocations for constrained roles.

It is worth emphasizing that our experimental scenario is quite complex for MRS coordination. In fact, it takes into account specific characteristics of real-world environments (e.g., allocation conflicts on indistinguishable moving objects), which are not usually considered in other coordination scenarios. Such features have required the definition of a new Task Assignment problem and associated solutions that are not taken into account in previous work (e.g., [19]).

The experiments performed show that our approach is able to effectively assign tasks to robots, while detecting and avoiding conflicts among team members. Moreover, the solutions adopted to implement the described coordination method on the Sony AIBO robots are well suited for our reference scenario. Finally, the quantitative evaluation performed in the simulated environment shows that the method successfully allocates tasks to robots in different operative conditions, scaling with the size of the team while maintaining a very low communication overhead.

Token Passing has been applied to Task Assignment in presence of conflicts in large scale Multi-Agent Systems [32]. The method presented there does not use broadcast messages, but it does not guarantee the absence of conflicts. In the present work the use of broadcast messages has been carefully introduced in order to guarantee a conflict free assignment for robots acting in real environments with dynamic perception.

The use of broadcast messages introduces a significant constraint on the robot communication network and thus limits the scalability of the system. However, the coordination algorithm proposed in this article does not pose severe limitations on communication since it requires a very low bandwidth: it scales up to the size that the communication system allows for reliably sending broadcast messages.

Moreover, it is important to observe that dealing with dynamic perception requires a certain amount of accuracy in the perception capabilities of the robot. A strict relationship exists between the perception capabilities of the robots

and their ability to distinguish tasks. In fact, when perception errors lead to incorrect task detection, the coordination mechanism may either lose performance or fail. Perception accuracy also limits task execution: for example, in our collecting scenario there is a limit of the minimum distance between two objects. The interplay between perception capabilities and Task Assignment is therefore a critical design issue in order to achieve successful coordination in MRS.

As future work several interesting extensions could be considered in order to realize a more robust system: relevant information on the object state (e.g. position) could be exchanged among agents, through the tokens. In this way, we would reduce the possibilities of false positives in the task generation process and further optimize the allocation process.

Cooperative perception techniques could be used for building a consistent global state of the world. Decisions based on such a global world state would improve the allocation process.

Another interesting direction of research is to dynamically change the number of robots involved in the task allocation. Robots should be added or removed (e.g. for a robot failure) from the system, while the team is performing its task. As a special case, this feature will allow for dealing with (temporary) robot failure or lost connection. In order to deal with a variable number of robots in the team, it is necessary to use redundant messages and higher communication bandwidth. The trade-off between robustness to robot loss or communication failures and communication bandwidth must be carefully considered depending on the application that is being realized.

Finally, to further scale the system size, we could divide the robotic agents into different communication channels, and broadcast information only within each channel. Robots should be able to dynamically leave and enter communication channels depending on the current environment configuration. The conflict detection algorithm should be properly extended to guarantee conflict-free allocation also in such a setting.

IX. ACKNOWLEDGMENT

This effort was partially founded by project “Simulation and Robotic Systems for intervention in emergency scenarios” within the program COFIN03 of the Italian MIUR, grant number 2003097252 and partially supported by the European Office of Aerospace Research and Development, grant number 053015. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the European Office of Aerospace Research and Development.

REFERENCES

- [1] A. Farinelli, L. Iocchi, and D. Nardi, “Multi robot systems: A classification based on coordination,” *IEEE Transactions on System Man and Cybernetics, part B*, vol. 34, no. 5, pp. pp. 2015–2028, October 2004.
- [2] G. Dudek, M. Jenkin, and E. Miliotis, *A Taxonomy of Multirobot Systems*. AK Peters, 2002, ch. in Robot Teams: From Diversity to Polymorphism, T. Balch and L. E. Parker, eds.
- [3] Y. U. Cao, A. Fukunaga, and A. Kahng, “Cooperative mobile robotics: Antecedents and directions,” *Autonomous Robots*, vol. 4, pp. 1–23, 1997.
- [4] L. E. Parker, “Current state of the art in multi-robot teams,” in *Distributed Autonomous Robotic Systems*. Springer, 2000, pp. 3–12.
- [5] R. Mailler, V. Lesser, and B. Horling, “Cooperative negotiation for soft real-time distributed resource allocation,” in *Proceedings of AAMAS’03*, 2003.
- [6] L. Hunsberger and B. Grosz, “A combinatorial auction for collaborative planning,” in *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-2000)*, 2000, pp. 151–158.
- [7] P. J. Modi, H. Jung, M. Tambe, W. M. Shen, and S. Kulkarni, “A dynamic distributed constraint satisfaction approach to resource allocation,” *Lecture Notes in Computer Science*, vol. 2239, pp. 685–700, 2001.
- [8] P. Scerri, D. V. Pynadath, L. Johnson, R. P., N. Schurr, M. Si, and M. Tambe, “A prototype infrastructure for distributed robot-agent-person teams,” in *In Proceedings of AAMAS*, 2003.
- [9] B. Gerkey and J. M. Matarić, “Multi-robot task allocation: Analyzing the complexity and optimality of key architectures,” in *Proc. of the Int. Conf. on Robotics and Automation (ICRA’03)*, Taipei, Taiwan, Sep 14 - 19 2003.
- [10] R. Zlot, A. Stenz, M. B. Dias, and S. Thayer, “Multi robot exploration controlled by a market economy,” in *Proc. of the Int. Conf. on Robotics and Automation (ICRA’02)*, Washington DC, May 2002, pp. 3016–3023.
- [11] L. E. Parker, “ALLIANCE: An architecture for fault tolerant multirobot cooperation,” *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, April 1998.

- [12] P. J. Modi, P. Scerri, S. W. M., and M. Tambe, *Distributed Sensor Networks A multiagent perspective*. Kluwer Academic, 2003, ch. Distributed Resource Allocation, pp. 219–256.
- [13] S. Fitzpatrick and L. Meetrens, *Distributed Sensor Networks A multiagent perspective*. Kluwer Academic, 2003, ch. Distributed Coordination through Anarchic Optimization, pp. 257–293.
- [14] B. Gerkey and M. J. Mataric, “Principled communication for dynamic multi-robot task allocation,” in *Proceedings of the Int. Symposium on Experimental Robotics*, Waikiki, Hawaii, Dec. 2000.
- [15] M. B. Dias and A. T. Stentz, “A market approach to multirobot coordination,” Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI -TR-01-26, August 2001.
- [16] L. Iocchi, D. Nardi, M. Piaggio, and A. Sgorbissa, “Distributed coordination in heterogeneous multi-robot systems,” *Autonomous Robots*, vol. 15, no. 2, pp. 155–168, 2003.
- [17] B. B. Weger and M. J. Mataric, “Broadcast of local eligibility for multi-target observation,” in *DARS00*, 2000, pp. 347–356.
- [18] A. Farinelli, L. Iocchi, D. Nardi, and V. A. Ziparo, “Task assignment with dynamic perception and constrained tasks in a multi-robot system,” in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005, pp. 1535–1540.
- [19] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe, “Token approach for role allocation in extreme teams,” in *In Proc. of AAMAS 05*, 2005, (to appear).
- [20] D. Shmoys and E. Tardos, “An approximation algorithm for the generalized assignment problem,” *Mathematical Programming*, vol. 62, pp. 461–474, 1993.
- [21] Dietl, J. S. Gutmann, and B. Nebel, “Cooperative sensing in dynamic environments,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’01)*, 2001.
- [22] A. Stroupe, M. Martin, and T. Balch, “Distributed sensor fusion for object position estimation by multi-robot systems,” *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on Volume: 2*, pp. 1092 – 1098, 2001.
- [23] S. Lenser and M. Veloso, “Sensor resetting localization for poorly modelled mobile robots,” in *Proceedings of International Conference on Robotics and Automation (ICRA’00)*, 2000.
- [24] T. Rfer and M. Ingel, “Fast and robust edge-based localization in the sony four-legged robot league,” in *Proc. 7th International Workshop on RoboCup 2003*, 2004.
- [25] A. Farinelli, G. Grisetti, and L. Iocchi, “Spqr-rdk: a modular framework for programming mobile robots,” in *Proc. of Int. RoboCup Symposium 2004*, N. et. al, Ed. Springer Verlag, 2005, pp. 653–660.
- [26] C. H. Goulden, *Methods of statistical Analysis*, 2nd ed. Wiley, 1956.
- [27] M. D. Dias and A. Stentz, “Opportunistic optimization for market-based multirobot control,” in *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS’02)*, Sept. 2002, pp. 2714–2720.
- [28] L. Chaimowicz, M. F. M. Campos, and V. Kumar, “Dynamic role assignment for cooperative robots,” in *Proc. of the 2002 IEEE Int. Conf. on Robotics and Automation (ICRA)*, Washington DC, May 2002, pp. 292 – 298.
- [29] A. Gage and R. R. Murphy, “Affective recruitment of distributed heterogeneous agents,” in *In proc. of Nineteenth National Conference on Artificial Intelligence*, 2004, pp. 14–19.
- [30] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert, “Multi robot cooperation in the martha project,” *IEEE Robotics and Automation Magazine*, vol. 5, no. 1, 1998.
- [31] D. Jung and A. Zelinsky, “An architecture for distributed cooperative planning in a behaviour-based multi-robot system,” *Journal of Robotics and Autonomous Systems* 26, pp149-174, 1999.
- [32] P. Scerri, Y. Xu, E. Liao, G. Lai, and K. Sycara, “Scaling teamwork to very large teams,” in *In Proceedings of AAMAS*, July 2004.
- [33] F. Cottefogle, A. Farinelli, L. Iocchi, and D. Nardi, “Dynamic token generation for constrained tasks in a multi-robot system,” in *International Conference on Systems, Man and Cybernetics*, The Hague, The Netherlands, 2004, pp. 911–917.