Proceedings of the 2005 IEEE
International Conference on Robotics and Automation
Barcelona, Spain, April 2005

Task Assignment with Dynamic Perception and Constrained Tasks in a Multi-Robot System

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

***Abstract*— In this paper we present an asynchronous distributed mechanism for allocating tasks in a team of robots. Tasks to be allocated are dynamically perceived from the environment and can be tied by execution constraints. Conflicts among team mates arise when an uncontrolled number of robots execute the same task, resulting in waste of effort and spatial conflicts. The critical aspect of task allocation in Multi Robot Systems is related to conflicts generated by limited and noisy perception capabilities of real robots. This requires significant extensions to the task allocation techniques developed for software agents. The proposed approach is able to successfully allocate roles to robots avoiding conflicts among team mates and maintaining low communication overhead. We implemented our method on AIBO robots and performed quantitative analysis in a simulated environment.**

***Index Terms*— Multi-Robot Systems, Coordination, Task Assignment.**

I. Introduction

The problem of assigning tasks to a group of entities acting in a dynamic environment is a fundamental issue for both Multi Agent Systems (MAS) and Multi Robot Systems (MRS) and is relevant to several real world applications.

The growing complexity of applications for MAS and MRS requires novel solutions for task assignment, which are able to address specific features posed by the domain, such as dynamic task evolution and strict requirements on communication and constraints among tasks to be executed. In most real world applications involving MRS, tasks to be assigned cannot be inserted into the system in a centralized fashion, but are perceived by each entity during mission execution. This issue has a big impact on the task allocation process and at the same time is strictly dependent on perception capabilities of entities involved.

Perception capabilities play a fundamental role in every application involving robots and significantly constraint the design of the task assignment technique. In particular, as a difference with software agents, robots have noisy and limited perception of the surrounding environment. For example, robots may need to correctly identify objects with similar shape and color. This requires one to consider, in the data association process, properties that can change with time (such as object position in the working environment). Consequently erroneous are frequent, which can easily lead to conflicts in the task allocation process. Conflicts on tasks to be accomplished can cause significant inefficiencies of the overall system due to the complexity of acting in a real world environment.

Task assignment approaches developed for MAS (such as [4], [8], [9]) usually do not take into account the perception capabilities of entities involved in the task allocation process, thus such techniques are not well suited for our reference scenarios. As for MRS, previous approaches proposed for task assignment include: i) Sequential task assignment [6] where tasks are allocated to robot sequentially as they enter the system. ii) Iterative task assignment [7], [13] where all tasks present in the system are allocated from scratch at each time step. iii) Reactive task assignment [10], where each member of the team decides whether to employ itself in accomplishing a task, without (re)-organizing the other members activity. While, in all these works the perception capabilities of robots influence the allocation method proposed, the authors do not explicitly investigate the problem of conflicts in the allocation process due to dynamic task generation.

In this paper we take a significant step towards the integration of dynamic task perception and distributed, conflict free, task assignment for a robotic system. We present an approach to distributed task assignment that allows to assign dynamically perceived tasks in a team of robotic agents, while avoiding conflicts among team members. Our reference 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.

The basic idea of our approach is derived from previous works on token passing for task assignment which have been proved to be well suited for task allocation in similar scenarios [11]. Tokens are used to represent tasks that must be executed by the agents, and 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. Such method assigns

0-7803-8914-X/05/$20.00 ©2005 IEEE.

tasks using only a broad knowledge of team mates, sharing a minimal set of information among team members. Such approach ensures that task allocation is highly reactive and requires a low communication overhead.

To apply such method on a system composed of physical robots this paper introduces the new concept of dynamic token generation. In fact, tokens are not statically predefined, but generated on-line during mission execution as result of robots 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 tested our approach with a team of robots involved in a foraging task (see [3] for a description of MRS testbeds). The robots have to collect several objects scattered in the environment. The collection 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. Object number and position in the environment is not known, thus enforcing task discovery through perception. Moreover, objects are identical so 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.

It is important to highlight that this is a quite complex scenario for MRS coordination, that takes into account specific characteristics of a real-time environment (e.g. indistinguishable objects), that are not usually considered in Multi-Agent Systems. Moreover, these features have required the definition of a new task assignment problem and associated solutions, that are not taken into account in previous works (e.g. [1], [11]).

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

II. Problem Definition

The problem of assigning a set of tasks to a set of entities can be easily framed as a Generalized Assignment Problem (GAP) [12]. However, while the GAP is well defined for a static environment, where agents and tasks are fixed and capabilities and resources do not depend on time, in several real world applications it is useful or even necessary to solve a similar problem where the defined parameters changes with time.

Indeed several methods for dynamic task assignment implicitly take into consideration such an aspect, providing solutions that consider the dynamics of the world and derive a task allocation that approximate solutions of the GAP problem at each time steps [5], [10], [14].

The problem we will address in this paper differs from the GAP formulation in two main respects: 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 entities. While in general also entities involved in the task assignment process can vary over time, in this contribution we focus on a predefined static set of entities. Γ represents the set of tasks and is dependent on time with $\Gamma_t = \{\tau_1 \dots \tau_{m(t)}\}$, where $m(t)$ is the number of tasks at time t. Each task is a set of roles or operations $\tau_i\{r_i^1 \dots r_i^k\}$, with k varying from task to task. For example, the Task *lift object O* comprises two roles: *collect O* and *support collection of O*. Notice, that each operation can comprise a set of sub-operations and so on; for the sake of simplicity we will consider only two levels of the possible hierarchy. Each entity has different capabilities for each task and different resources available. We express the capabilities and the resources depending on time with $Cap(e_i, r_j^k, t)$, $Res(e_i, r_j^k, t)$, and $e_i.res(t)$. Where $Cap(e_i, r_j^k)$ represents the reward for the team when agent e_i performs role r_j^k at time t, $Res(e_i, r_j^k)$ represents the resources needed by e_i to perform r_j^k at time t, while $e_i.res(t)$ represents the available resources for e_i at time t.

A dynamic allocation matrix is used to establish task assignment, denoted by A_t; in A_t, $a_{e_i,r_j^k,t} = 1$ if and only if the agent e_i is assigned to task r_j^k at time t. Furthermore, to represent constraints, we define $\bowtie$ as the set of relationships which hold among roles. Consequently, the problem is to find a dynamic allocation matrix that maximizes the following function

$$f(A_t) = \sum_t \sum_i \sum_j \sum_k Val(e_i, r_j^k, \bowtie, t) \quad (1)$$

with:

$$Val(e_i, r_j^k, \bowtie, t) = \begin{cases} Cap(e_i, r_j^k, t) & if\ \ Cond \\ 0 & otherwise \end{cases} \quad (2)$$

subject to:

$$\forall t \forall i \sum_{j=1}^{m(t)} Res(e_i, r_j^k, t) \times a_{e_i,r_j^k,t} \leq e_i.res(t) \quad (3)$$

$$\forall t \forall j \in \{0, \dots, m(t)\} \sum_i a_{e_i,r_j^k,t} \leq 1 \quad (4)$$

where $Cond$ is true when constraints relative to r_j^k are satisfied.

Constraints can possibly be of several types (OR, XOR, AND), in this paper we focus only on AND constraints. When an AND constraint holds among a group of roles, each agent cannot perform any role if all the other roles are not being performed simultaneously by some other team mates. We write an AND constraint as $\text{AND}^i = \{r_v\}$ therefore $\bowtie = \{\text{AND}^1 \dots \text{AND}^s\}$. If a role $r_v \in \text{AND}^z$ then the equation describing $Cond$ is:

$$Cond \equiv \sum_{i} \sum_{r_j^k \in \text{AND}^z} a_{e_i,r_j^k,t} = |\text{AND}^z| \quad (5)$$

Notice that if the role is unconstrained, $|\text{AND}^z| = 1$, then $Val(e_i, r_j^k, \bowtie, t) = Cap(e_i, r_j^k, t) \times a_{e_i,r_j^k,t}$, as above. Agents have only a local view of the environment, thus we define two sets: $LOT_{i,t}$, which is the task set locally known to agent i at time t (Locally Observed Tasks), and $GOT_t = \bigcup_i LOT_{i,t}$ which is the union of tasks locally known to each agent i at time t (Globally Observed Tasks). The $LOT_{i,t}$ is built by each agent based on its local perception thus we can write $LOT_{i,t} = Mem(LOT_{i,t-1}, O(i,t))$ where $O(i,t) : A \times Time \rightarrow \mathcal{P}(\Gamma \times \{0,1\})$. Given an agent a and a time step t, $O(i,t)$ returns a set of pairs $< \tau, 1 >$ if task τ is active and visible for agent a at time step t, and $< \tau, 0 >$ if task τ is visible for the agent but is not active. An active task is a new perceived task that needs to be accomplished. Notice that assuming that the observation function is able to distinguish between active and non active tasks is quite a strong assumption; for example, in our reference scenario, objects are similar in shape and color and the distinction is made using object position, knowing if an object is being moved by another team mate is not a trivial problem. Thus, in this contribution we consider cases where the observation function can fail in deciding whether a detected object is to be considered active, and explicitly address this problem in the coordination method.

The Mem function integrates observation for an agent during the mission execution. We assume that the Mem function add newly discovered tasks and is able to remove non active tasks from the Locally Observed Tasks set. We define $LRS_{i,t} = \{r_j^k | a_{e_i,r_j^k,t} = 1\}$ (Local Roles Set) and $GRS_t = \bigcup_i LRS_{i,t}$ which are the currently assigned roles (Global Roles Set). An *allocation* for a D-GAP problem is the set $Alloc = \{LRS_{i,t}\}$. The global constraints 4 can be then expressed as

$$\bigcap_i LRS_{i,t} = \emptyset \quad (6)$$

We define a non conflicting allocation as an allocation for which the following holds:

$$\forall t \bigcap_i LRS_{i,t} = \emptyset \quad (7)$$

III. Token Passing Approach for Role Allocation

The main idea of the token passing approach is to regulate access to task execution, through the use of tokens, i.e. only the agent currently holding the token can execute the task. Following this approach the communication needed to guarantee that each task is performed by one agent at time is dramatically reduced.

If a task can benefit from the simultaneous execution of several agents, one possible strategy is to create several tokens referring to the same task. However, when tokens are perceived and generated by agents during mission execution conflicts on tasks may arise due to the fact that several agents may perceive the same task; and thus create an uncontrolled number of tokens leading too many agents to execute the same role. Moreover, an explicit procedure is needed to enforce AND constraints among roles. In the following, we present and discuss our approach to ensure that exactly n agents perform the same role simultaneously, thus solving both the issues previously highlighted.

A *Task* is characterized by the physical objects (or events) that the agent perceives, therefore given a perceived object o we define the related task $\tau(o)$. We associate a token to each role comprising a task to be accomplished, thus for the object o we will have $\tau(o) = \{\tau(o)^1 \dots \tau(o)^k\}$. In particular, in our reference scenario, when a new object obj is found we need two robots cooperating to grab the object, therefore we have a task $Move(obj)$ and two roles (and thus two tokens) for this task: $\{collect(obj), support(obj)\}$.

To prevent possible conflicts, that may arise during mission execution, we have to guarantee that no more than n tokens are created for the same role. The main idea of the proposed algorithm is that when an agent perceives an object, it records this information in a local structure and announces the presence of the object to all its team mates. Moreover, whenever an agent accomplishes a task, it announces to the entire team task termination, and each of the team members removes the tokens referring to the accomplished task from its local structures.

Using 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 $Tk(t,1)...Tk(t,s)$ exactly at the same time, such that both agents will have different tokens referring to the same task. ii) **Messages asynchrony**: messages are not guaranteed to arrive in a predefined order; Suppose an entity observes a new object, creates a new token for a specified role and decides to pass the token on. If the token reaches a team member before the new perception announcement, the team member can decide to perform the role possibly conflicting with other team mates. iii) **Errors in active object detection**: if the observation function fails in the recognition of an active object, a team member can decide to allocate itself to a role that is already being carried on by someone else.

In the following, we describe our approach to avoid conflicts during mission execution and further details of the token passing process.

A. Distributed conflict detection

Simultaneous task perception and message asynchrony are addressed using a distributed approach for conflict detection and a predefined global policy that orders all the team mates. Each agent maintains in a local structure called Invalid Token Set (ITS) all the detected conflicting tokens. Each new announce is registered in the Known Task Set (KTS) maintaining the announcer agent information. Tokens are added in the ITS structure when an announce message for an already known object is received and the

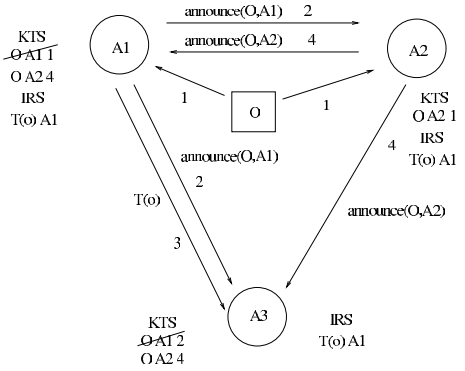

Fig. 1. Distributed conflict detection

announcer agent has a lower priority as compared with the agent that already announced the object detection. If a higher priority agent announces the same object, then the KTS is updated. An example of possible execution of this procedure is reported in Figure 1. Numbers on arrows correspond to time step at which messages are received by agents. The square represents the object and arrows starting from the square represent object perception. KTS is the Known Task Set and IRS is the invalid role set. Notice that KTS contains also information on the creator agent. As shown in Figure 1, all agents share the same IRS without need of any further communication.

Setting a static fixed priority among agents can obviously result in non optimal behavior of the team; for example, assuming that $Cap(e_1, r_j^k, t) > Cap(e_2, r_j^k, t)$ following a static priority based on id, we yield to the less capable agent the access to the task r_j^k. While in principle the difference among capabilities can be unbounded, generally, 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 small.

B. Avoiding failures in active task recognition

Suppose that when a collector robot moves an object, another robot observes the same object, This robot will consider the object as a new task to accomplish starting a new allocation process for the task and possibly conflicting with the collector robot.

To address this problem we partition the working environment using a regular grid. While an object is perceived inside a cell of the grid it is considered as the same task. Suppose a robot observes the moved object and starts the allocation of a new task; when the collector robot is reached by an announce message for the object creation, it can detect that the new task is actually the one it is performing and declare the new created task as invalid. The collector robot will then announce the task invalidation to all its team mates sending an InvalidTask message in broadcast. Tasks are considered invalid only for a given amount of time. This is needed to avoid that collector robots passing near other (active) objects invalidate them for the whole mission execution.

C. Token passing process

While we do not report here the complete algorithm we give some further details to clarify how the token passing process is used to assign roles to team members.

Once a token has been created the token-based access to values requires that each agent decides whether to execute the tasks represented by tokens it currently has or to pass the tokens on. Each agent follows a greedy policy in this decision process, i.e. it tries to maximize its utility given the tokens it currently can access, its resource constraints and a broad knowledge on team composition (see [11] for further details).

When roles are tied by AND constraints, following this procedure will lead to potential deadlocks or inefficiencies. For example, consider two tasks, r_j and r_k, that need to be simultaneously performed. When a team member a accepts task r_j, it may reject other tasks that it could potentially perform. If there is no team member currently available to perform task r_k, a must simply wait. Thus, an explicit procedure to enforce the AND constraints among roles is needed. The general idea is to use potential tokens to represent tokens 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 have impact on the current resource load of the agent). However, when a lock message arrives the agent is forced to execute the role.

IV. Experiments and Results

We implemented and tested the described method on the Sony AIBO robots. Our reference scenario is formed by a set of robots that need to perform a synchronized operation on a set of objects scattered in the environment. For each object two robots are needed to perform synchronized operations, while only one robot is needed to accomplish the task. In particular, robots have to collect a set of identically colored wheeled-bars. Each bar, to be correctly transported, requires one robot to grab it, blocking the center of the bar with its head. However, since robot perception is very noisy and unreliable, it is very hard for the grabbing robot to precisely estimate the position of the bar. In particular, if the robot hits the bar trying to reach the grabbing position, the bar will roll away from the robot. To avoid this, a second robot (the supporting robot) helps in the grabbing phase blocking the bar and preventing it to roll away from the grabbing robot. Once the bar has been grabbed, the supporting robot moves away from the bar and is ready to be allocated to a different task, while the grabbing robot can transport the grabbed bar in the desired position

In order to successfully operate in this scenario, three main components are required: 1) self-localization, 2) action synchronization, and 3) object recognition. To put emphasis on coordination issues, we have used an engineered environment, borrowed from the RoboCup Legged

League[1], that is a rectangular field with six landmarks in known positions where every landmark is clearly distinguishable by its color. In this way it is possible to implement a simple and effective landmark-based localization method.

Action synchronization has been realized through communication-based actions that exchange synchronization information among robots.

Finally, the identification of objects to collect (colored bars) 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 computed, in order to distinguish these objects. Note that these information are subject to errors due to noisy perception and imprecise self-localization, that can lead to false positives in task detection. To avoid this problem, we filter the absolute position of objects using a very conservative policy; moreover, we use active perception to ensure the correctness of the observations. In fact, due to the poor reactiveness in perception, if an object is incorrectly perceived while moving to the expected position of the current task, situations in which erroneous tasks are generated may occur. Therefore, once the robot is close enough to the object, it interleaves object tracking with landmark pointing action for self-localization in order to identify the object and understand if it is the one associated with his task. Although this makes the system more robust to false positives we cannot assume their absence. Thus, if during the execution of a role a robot realizes that an announced object is a false positive, it has to invalidate the corresponding task with an Invalid Role message.

Because of the small amount of messages needed by our coordination method, we conveniently adopt the TCP protocol, so that we do not have to manage message loss.

Since tasks to be accomplished are complex and we can not assume instantaneous actions nor deterministic effects, a reactive module is in charge of controlling the task execution, identifying action failures during their execution, and specifying ad-hoc recovery procedures.

The implementation on AIBO robots has shown the feasibility of the approach on robots with limited computational resources. In particular, we experienced no network problems, thanks to the low bandwidth required by the coordination mechanism presented here.

Moreover, in order to have a consistent data set to analyze, we used a general simulation framework, that has been developed for reproducing the behavior of a robot in a dynamic environment [2]. More specifically, each robot is simulated in a separate process, that integrates different components (perception, localization, navigation, action execution, planning, and coordination). Processes emulating robots communicate among them and with a global module that provides sensorial information. Finally, a graphical application is used to display the status of the robots and of the environment. The simulator accurately reproduces the problems arising in the real experiments (i.e. noisy perception, non-deterministic action execution, etc), thus allowing for a realistic analysis of the performance of the robots in a dynamic environment.

[1]See web site http://www.openr.org/robocup

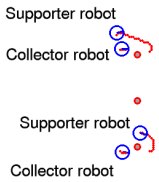

Fig. 2. Four robots correctly allocated to two tasks

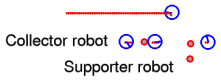

Fig. 3. Four robots almost completing their tasks

In Figures 2 and 3 we report an example of task execution: four robots are cooperating in the foraging task with three objects involved. In Figure 2 the four robots (represented as large circles) correctly performed the initial task allocation and are moving toward two objects (represented as smaller circle), satisfying the AND constraint; small dots represent the paths robots have planned. In Figure 3 two objects have already been collected, and the last object is being grabbed by two robots; the current supporter robot was a former collector coming back from the collect area, and the current collector was a former supporter. The other two robots are correctly unassigned and will begin their search procedure to check that no active objects are present in the working area.

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 noise that leads to conflicts and enforces that the grabbing phase must be performed by exactly two robots that synchronize their actions, thus the task is accomplished only if conflicts are correctly solved and robots synchronize their actions for every object in the environment. All performed experiments have terminated, thus showing that the algorithm successfully manage conflicts and that AND constrained roles do not cause deadlocks, always converging to a valid allocation.

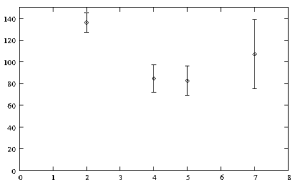

Fig. 4. Average time for completion measured in second, x-axis is number of agents; results averaged over 20 simulations

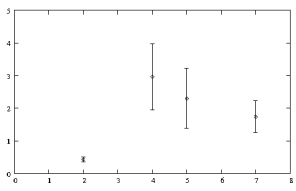

Fig. 5. Messages exchanged per second per agent, x-axis is number of agents; results averaged over 20 simulations

In Figures 4 and 5 we report the results obtained for two, four, five and seven robots and three objects. In the experiments all robots can perceive all objects, thus generating the highest number of possible conflicts. Figure 4 shows an interesting behavior with respect to time for completion: the time needed by agents to accomplish all tasks decreases when the number of agents increases, but when too many robots are present in the working environment performance 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 we leave the number of tasks and the size of the working environment fixed.

Figure 5 shows that the number of messages exchanged per agent per second is very small for all the experiments. Consider that coordination methods based on iterative task assignment (such as [7], [13]) for a similar application with n agents and m tasks would require nm messages for each agent at each time step; while in token based approaches it remains bounded with respect to the number of agents (with a fixed number of tasks) [11].

The variable number of messages with respect to team size is due to two main factors: i) the number of messages may increase with the team size because of a higher number of conflicts; ii) the number of messages may increase due to the number of cycles of not allocated tokens. In fact, when tokens are refused by all the agents, they wait a given amount of time and then they are reinserted into the system.

The spike present in the configuration with four agents is due to the second factor; in fact, in the configurations with four and five agents the number of tokens in the system is the same, but in the first case we have more cycles, because there are less agents. However, cycles due to not allocated tokens can be reduced tuning the waiting time. Moreover, the messages exchanged to resolve conflicts do not impact on the system communication overhead in normal situations. Based on the general behavior of token based approaches and on the above considerations we expect the method to be scalable with respect to team size.

V. Conclusions and Future works

In this paper we have presented a distributed asynchronous algorithm for task assignment in dynamic environment. The presented approach is based on token passing for role allocation and successfully achieves the integration of the task assignment approach with object perception from the environment. The approach can detect and solve conflicts in role execution and provide correct allocation for constrained roles.

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

As future work an interesting extension would be to dynamically change the number of robots involved in the task allocation process and to optimize the allocation process by deciding whether to accept tokens and whom to pass tokens based on probabilistic models of team members.

VI. Acknowledgment

This effort was partially founded by project "Simulation and Robotic Systems for intervention in emergency scenarios" within program COFIN03 of the Italian MIUR, grant number 2003097252 and partially sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-03-1-3A46. 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 Air Force Office of Scientific Research or the U.S. Government.

References

[1] F. Cottefoglie, 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.

[2] A. Farinelli, G. Grisetti, and L. Iocchi. Spqr-rdk: a modular framework for programming mobile robots. In Nardi et. al, editor, *Proc. of Int. RoboCup Symposium 2004*, pages 653–660. Springer Verlag, 2005.

[3] A. Farinelli, L. Iocchi, and D. Nardi. Multi robot systems: A classification based on coordination. *IEEE Transactions on System Man and Cybernetics, part B*, 34(5):pp. 2015–2028, October 2004.

[4] S. Fitzpatrick and L. Meetrens. *Distributed Sensor Networks A multiagent perspective*, chapter Distributed Coordination through Anarchic Optimization, pages 257–293. Kluwer Academic, 2003.

[5] 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.

[6] B. Gerkey and M. J. Matarić. Principled communication for dynamic multi-robot task allocation. In *Proceedings of the Int. Symposium on Experimental Robotics*, Waikiki, Hawaii, December 2000.

[7] L. Iocchi, D. Nardi, M. Piaggio, and A. Sgorbissa. Distributed coordination in heterogeneous multi-robot systems. *Autonomous Robots*, 15(2):155–168, 2003.

[8] Roger Mailler, Victor Lesser, and Bryan Horling. Cooperative negotiation for soft real-time distributed resource allocation. In *Proceedings of AAMAS'03*, 2003.

[9] P. J. Modi, P. Scerri, Shen W. M., and M. Tambe. *Distributed Sensor Networks A multiagent perspective*, chapter Distributed Resource Allocation, pages 219–256. Kluwer Academic, 2003.

[10] L. E. Parker. ALLIANCE: An architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, April 1998.

[11] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Token approach for role allocation in extreme teams: analysis and experimental evaluation. In *13th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE-2004)*., Modena, Italy, 2004.

[12] D. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62:461–474, 1993.

[13] B. B. Werger and M. J. Mataric. Broadcast of local eligibility for multi-target observation. In *DARS00*, pages 347–356, 2000.

[14] 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)*, pages 3016–3023, Washington DC, May 2002.