Dynamic Role Assignment for Cooperative Robots

Luiz Chaimowicz[1], Mario F. M. Campos[1] and Vijay Kumar[2]

[1]DCC – Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil, 31270-010

[2]GRASP Laboratory – University of Pennsylvania, Philadelphia, PA, USA, 19104

{chaimo,mario}@dcc.ufmg.br, kumar@grasp.cis.upenn.edu

Abstract

This paper proposes a new methodology for coordinating multi-robot teams in the execution of cooperative tasks. It is based on a dynamic role assignment mechanism in which the robots assume and exchange roles during cooperation. We model the role assignment under a hybrid systems framework, using a hybrid automaton to represent roles, transitions and controllers. Using a multi-robot simulator, the methodology is demonstrated in a cooperative transportation task, in which a group of robots must find and cooperatively transport several objects scattered in the environment.

1 Introduction

The coordination of multiple robots in the execution of cooperative tasks is one of the fundamental aspects of cooperative architectures. Basically, the actions performed by each team member during each phase of the cooperation must be specified considering several aspects such as robot properties, task requirements, and characteristics of the environment. In addition to organizing the robots in a purposeful manner, the coordination mechanism should provide flexibility and adaptability, allowing the robots to complete the cooperative task more efficiently and robustly. Strict coordination is even more important in the execution of tightly coupled tasks, where each robot depends on the action of its teammates and the task cannot be completed by a single robot working independently.

This paper presents a methodology for coordinating multiple robots in the execution of cooperative tasks. Each robot in the team performs a role that determines its actions during the cooperation. Dynamically assuming and exchanging roles, the robots are able to perform the task more efficiently, adapting to unexpected events in the environment and improving their individual performance in benefit of the team. This paper extends the work presented in [4] where dynamic role assignment was used for the coordination of real robots in the execution of a tightly coupled task. In that work, the robots could exchange leadership in a cooperative manipulation task, adapting their coordination patterns in the presence of unexpected events.

Generically, the role assignment presented in this paper can be viewed as a task allocation problem. Several researchers have studied this problem, both for multi-agent systems (MAS) [10] and distributed robots. In the cooperative robotics field, an interesting approach is Alliance [9], a behavior-based software architecture for heterogeneous multi-robot cooperation. It has a fault tolerance mechanism that allows the robots to detect failures in their teammates and adapt their behaviors to complete the task. Another behavior based approach is presented in [13], in which robots broadcast messages with their eligibility in order to coordinate their actions in a multi-target observation task. Although some approaches propose coordination methods without the use of explicit communication (for example [7]), the development of cheaper and more reliable communication mechanisms has motivated the use of explicit communication in multi-robot task allocation, mainly for tightly coupled tasks. In [5], for example, there is a description of a dynamic task allocation method based on publish/subscribe messaging.

The term dynamic role assignment is used in [11] and [3], where role assignment and formation switching are used in a multi-robot soccer domain. The definitions of roles and dynamic role assignment presented here are somewhat related but a more formal approach is used to describe them. We model roles and the role assignment mechanism respectively as discrete modes and mode switching in a hybrid automata. The parallel composition of these automata defines the cooperative task execution. Using this hybrid systems framework, it is possible to better describe the behavior of each robot, specifying the continuous controllers and information flow during the task execution.

Our role assignment mechanism is demonstrated in simulations of a cooperative transportation task, in which a group of robots must find and cooperatively transport several objects scattered in the environment. It is a combination of a loosely coupled

task, where the robots search the area independently looking for objects, and a tightly coupled task, in which the robots must manipulate objects in cooperation. This task is similar to the cooperative search and rescue proposed in [6] with the basic difference that more than one object must be transported to complete this task. Another similar task is the object sorting described in [8], where groups of robots must transport several objects between different locations in a bounded area.

This work is organized as follows. The next section presents the dynamic role assignment mechanism using a hybrid systems framework. Section 3 describes the cooperative transportation task used to demonstrate the role assignment mechanism. In Section 4, the experimental results are shown and Section 5 gives a summary and directions for future work.

2 Dynamic Role Assignment

2.1 Role

Before describing the role assignment mechanism, it is necessary to define what is a role in a cooperative task. Webster's Dictionary[1] defines **Role** as: (a) *a function or part performed especially in a particular operation or process* and (b) *a socially expected behavior pattern usually determined by an individual's status in a particular society.* We consider that a role is a function that one or more robots performs during the execution of a cooperative task. Each robot will be performing a role while certain internal and external conditions are satisfied, and will assume another role otherwise. Thus, a role depends on the internal robot state and on information about the environment and other robots, and defines the set of controllers that will be controlling the robot in that moment.

In [11], a role is defined as the specification of an agent's internal and external behaviors. A formation is a set of roles, decomposing the task space. Each agent knows the current formation and keeps mappings from teammates to roles in the current formation. Our definition of role is similar, the main difference being that we do not have the concept of formation and we use a more formal model to describe roles and role assignments, as it will be further explained in the next sections. As mentioned before, each role defines a robot controller and the role assignment allows the robots to change their behaviors dynamically during the task execution.

2.2 Hybrid Systems

More formally, a role can be described as a control mode in a *hybrid automaton*. A hybrid automaton is a finite automaton augmented with a finite number of real-valued variables that change continuously, as specified by differential equations and inequalities [1]. It is used to describe hybrid systems, i.e., systems that are composed by discrete and continuous states. A hybrid automaton H can be generally described by a tuple:

$$H = \{Q, X, E, f, Inv, G, Init, R\},$$

where Q is the set of discrete states, also called *control modes* and X represents the continuous states (variables). Discrete transitions between control modes are specified by the control switches E, while the continuous dynamics of the variables are determined by the flows f, generally described as differential equations inside each control mode. Invariants (Inv) and guards (G) are predicates related to the control modes and control switches respectively. The system can stay in a certain control mode while its invariant is satisfied, and can take a control switch when its guard (jump condition) is satisfied. The initial states of the system are given by $Init$, and each control switch can also have a reset statement R associated, to change the value of some variable during a discrete transition.

Using a hybrid automaton and representing roles as control modes, we are able to better describe the robots during a cooperative task. The internal states and sensory information can be specified by continuous variables, and updated according to the dynamic equations within each mode. The role assignment is represented by the discrete transitions. The invariants and guards define when each robot will assume a new role. Cooperative execution can be represented by a parallel composition of several automata, one for each robot. Using communication the robots are able to synchronize their automata and execute the cooperative task.

2.3 Role Assignment

The role assignment mechanism allows multi-robot coordination in the execution of cooperative tasks. As mentioned before, dynamically assigning and exchanging roles, the robots are able to perform the task more efficiently, adapting to unexpected events in the environment and improving their individual performance in benefit of the team.

Basically, there are three types of role assignment (Figure 1):

- Allocation: the robot assumes a new role after finishing its execution in another role;
- Reallocation: the robot interrupts the performance of one role and starts or continues the performance of another role;

[1] http:\\www.webster.com

- Exchange: two robots synchronize themselves and exchange the roles, each one assuming the role of the other;

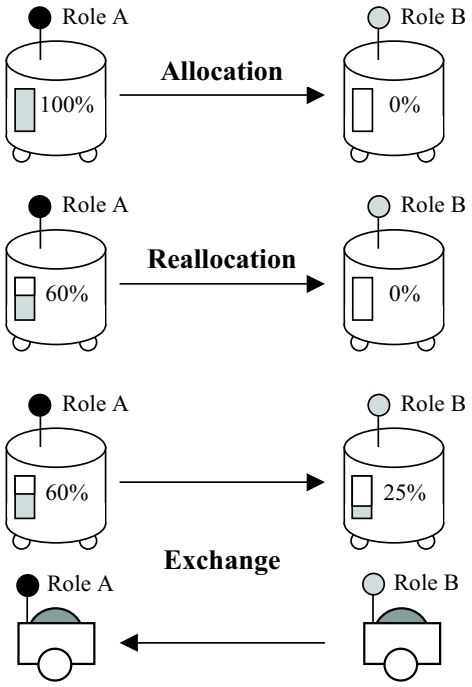

Figure 1: Types of role assignment.

An important point is to define when a robot should change its role. The allocation is simple: the robot detects that it has finished its role and assumes another available role according to its hybrid automaton. In the reallocation process, the robots should know when to give up the current role and assume other. A possible way to do that is to use a function that measures the utility of performing a given role. A robot performing a role r has a utility given by μ_r. When a new role r' is available, the robot computes the utility of executing the new role $\mu_{r'}$. If the difference between the utilities is greater than a threshold τ ($\mu_{r'} - \mu_r > \tau$) the robot changes its role. Function μ can be computed based on local and global information and may be different for distinct roles. Also, the value τ must be chosen such that the possible overhead of changing roles will be compensated by a substantial gain on the utility and consequently a better overall performance. The other type of role assignment is the role exchange, in which the robots agree in changing their roles and must synchronize the process. An example is the leadership exchange mechanism presented in [4].

3 Cooperative Transportation

In this paper, we demonstrate the use of the dynamic role assignment mechanism in a cooperative transportation task. The cooperative transportation can be stated as follows: a group of n robots must find m objects that are scattered in an area and transport them to a goal location. Each object i requires k robots ($k > 1$) to be transported and has a importance value v. So, each object can be described by a pair $\{k, v\}$. Differently from a common foraging task, in which the robots can act completely independent from each other and communication is not necessary, this task requires the robots to coordinate themselves in order to transport the objects in cooperation. Consequently, the cooperative transportation combines a completely loosely coupled task, in which the robots must find the objects, with a tightly coupled task that is the cooperative manipulation. It is important to mention that this work focuses on the coordination strategy. We do not focus on specific aspects of these tasks such as the impact of communication in the forage task [2] or the mechanics of cooperative manipulation [12].

In the cooperative transportation, all robots start in the **Explore** mode, in which they randomly move in the environment searching for items to be transported. When a robot detects an object, it finishes its explore role and starts the **Attach Lead** role. The attach leader is responsible for broadcasting messages informing the other robots about the new role available, and the number of volunteers that are necessary. All robots that receive this message compare the new role utility $\mu_{r'}$ with their current utility μ_r and send a message back to the attach leader if they want to volunteer for the new role. This works as a bidding process, where volunteers with the higher utility values are recruited by the attach leader. These robots reallocate to the **Approach** role and start moving towards the object. When the object is within the robot's sensor range, it assumes the **Attach** role. When the number of robots necessary to carry the object is sufficient, they assume the **Transport** role and move the object to the goal.

When a robot assumes the approach or attach roles, it makes a commitment to the attach leader. The attach leader keeps broadcasting messages in a fixed rate offering the role until the number of committed robots is sufficient to transport the object. If a committed robot reallocates to another role, it must send a message to the leader resigning its current role.

In each one of these roles, robots may be controlled by different continuous equations. For example, in the explore mode they move randomly while in the approach and attach modes they use a potential field

like controller in order to approach the objects. Also, other continuous and discrete variables may be stored within each mode and updated during the execution of the task. The use of hybrid systems allows the formalization of these discrete and continuous iterations, being a suitable tool for modeling the cooperative robots. Figure 2 shows the hybrid automaton for the robots executing the cooperative transportation. For clarity, only the control modes (roles) and discrete transitions (role assignments) are presented. The solid arrows represent the role allocation and the dashed arrows represent the reallocation, in which the robots interrupt the performance of one role to assume another. There are four role reallocations in this diagram: the first one is when an explorer volunteers and is recruited to approach an object, as explained before. The same thing can happen when the robot is already in the attach mode and an approach role with better utility is offered. The other two reallocations happen from/to the attach lead mode: an attach leader can reallocate itself to an approach role with higher utility if its object has no other attachers. In this case, the robot stores its position in local memory in order to possibly return to this object after finishing the new role. Also, a robot that is approaching can become an attach leader if it finds a new object and the utility of the new role is higher than its current utility. Another kind of reallocation is when a robot approaching an object i reallocates to approach a different object j. In this case, the robot will be performing the same role but with different parameters.

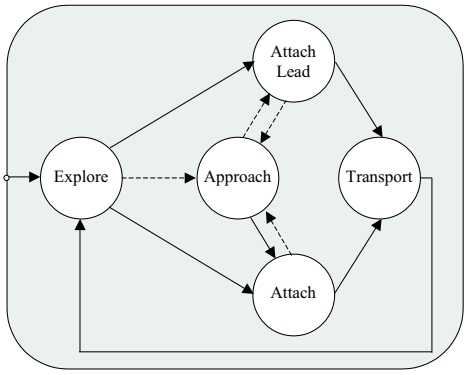

Figure 2: Control modes (roles) and discrete transitions (role assignments) for the cooperative task.

The choice of a suitable function to measure the role utilities is an important aspect of the task. The execution of the role assignment mechanism and consequently the performance of the task will vary according to the function chosen to measure the role utilities. Depending on the objective of the cooperative transportation, for example minimize execution time or maximize the value in a shorter time, different utility functions can be implemented. In the experiments presented in this paper we use a simple function in order to test the execution of the role assignment mechanism. We do not intend to compare different functions, analyze performance in details or search for optimal results. Instead, we just want to provide a simple test bed for our role assignment mechanism. The selection of optimal utility functions for the role assignment (and for task allocation in general) is a difficult problem in itself and is out of the scope of this paper.

4 Experiments

The dynamic role assignment in a cooperative transportation task was implemented and tested using a simulator that we have developed for cooperative robotics. MuRoS[2] is a multi-robot simulator that can be used for simulating various types of tasks, ranging from loosely coupled to tightly coupled cooperative tasks. Implemented using object orientation in the MS Windows environment, MuRoS has a very friendly user interface and can be easily extended with the development of new inherited classes defining new robots, controllers and sensors. Used alone or in conjunction with implementations in real platforms, the simulator has allowed the study of different aspects of cooperative robotics in several application domains.

Figure 3 shows a snapshot of the simulator during the execution of the cooperative transportation task. In this figure, the goal is represented by a square area marked with an x and the objects are represented by the five circles with numbers (a pair $\{k, v\}$) inside. Two of them (inside the goal region) have already been transported. The small circles are the robots and the dashed circles represent the boundary of their sensing area. The robot color represents its current role: two white robots, one at the bottom right and the other at the top left of the screen, are in the explore mode. Three black robots at the center of the screen are transporting the object marked with the numbers (3,1) to the goal. At the bottom left, there is the attach leader (light gray) and two gray robots attaching the object (5,1). The other two robots (dark gray) are approaching the same object.

4.1 Results

The cooperative transportation was executed with 20 holonomic robots and 30 objects randomly distributed in the environment. The value (v) and the number of robots (k) necessary to transport each object were also generated randomly, with $v = \{1, \ldots, 5\}$

[2]http:\\www.verlab.dcc.ufmg.br\muros

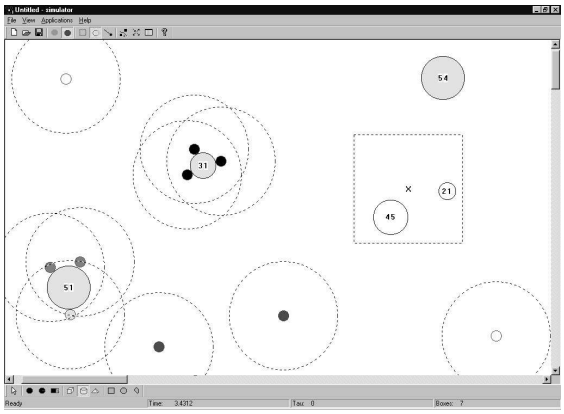

Figure 3: Snapshot of the simulator during the execution of the cooperative transportation task.

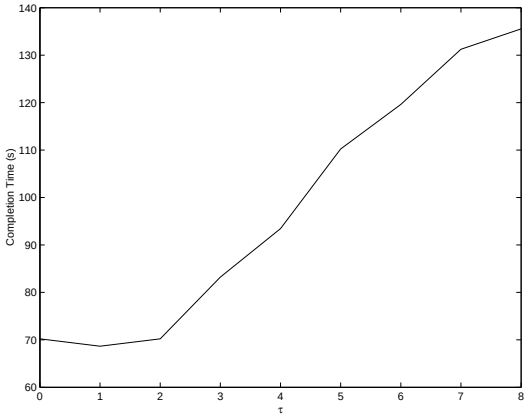

Figure 4: Completion time varying the threshold τ.

and $k = \{2, \ldots, 5\}$. We consider that the robots know their position on the environment and the explicit communication is error-free.

The utility function μ used in the experiments presented here is defined as follows: robots performing the explore role have a very low utility (0) while robots transporting an object have the higher utility (∞). For the other roles, we have defined an utility function that balances the value of the object (v) with the number of robots being waited to start the transportation (k_w) and a function of the distance to the object ($f(d)$). Thus, the utility of performing a role r is given by:

$$\mu = \begin{cases} 0, & r = \text{Explore}, \\ \infty, & r = \text{Transport}, \\ \frac{v^2}{k_w+1} + \frac{1}{f(d)}, & \textit{otherwise}. \end{cases}$$

Using this heuristic function, each robot tries to maximize the value recovered in a short time but also gives priority to objects that need few robots to be transported and are near the robot's current position. Note that robots in the Transport mode will never be reallocated while robots performing the Explore role have a great probability of being reallocated, depending on the threshold. For example, for a threshold $\tau = 0$ the robots in the Explore mode will always be reallocated.

The experiments were performed using this function and varying the threshold τ. As explained before, a robot performing a role r reallocates to another role r' when $\mu_{r'} - \mu_r > \tau$. Firstly, the average time to complete the task was measured. For each value of τ, 100 runs were performed and the average time was computed. The results are shown in Figure 4.

The graph shows that the completion time starts to increase for values of τ greater than 2. This result was expected because the number of role reallocations decreases as τ increases. With few reallocations, the robots act more independently as they do not accept new role offers. In this situation, the work force is divided and the time to gather the robots to transport objects increases. The extreme case of this division causes deadlocks. A deadlock occurs when each robot is performing the attach role onto a specific object, but the number of robots attached is not sufficient to transport the objects. In this case, the robots keep waiting indefinitely and do not complete the task. Figure 5 shows the number of deadlocks for each value of τ. In these experiments, more than 50% of the runs with large values of τ results in deadlocks.

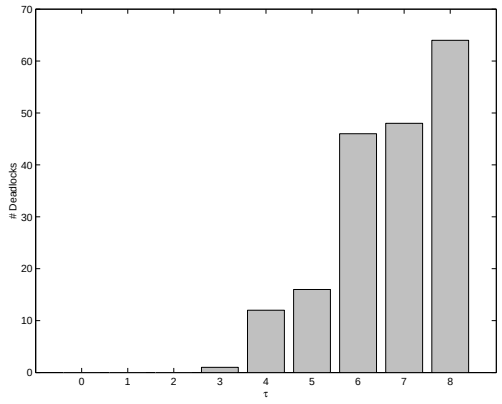

Figure 5: Number of deadlocks (in 100 runs) for different values of τ.

Another observed result is that the use of the dynamic role assignment with the utility function explained above helps maximizing the total value transported in the beginning of the execution. The graph of Figure 6 shows the percentage of the total value recovered as a function of the execution time for dif-

ferent values of τ. For small values of τ, objects with larger values are transported first, according to the utility function.

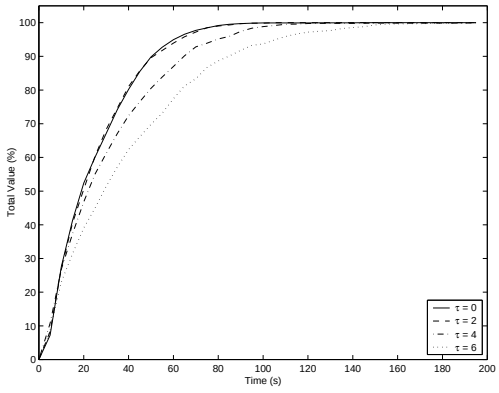

Figure 6: Percentage of the total value transported as a function of the execution time.

Observing the results, it can be seen that the dynamic role assignment allows the successful execution of the cooperative transportation task. Choosing a suitable utility function and adequate threshold values, it is possible to have a good performance in terms of time and other metrics while avoiding deadlocks that would prevent the task completion.

5 Conclusion

In this paper we presented a new methodology for coordinating multiple robots in the execution of cooperative tasks. Each robot performs a set of roles that define its actions during the cooperation. Dynamically assuming and changing roles, the multi-robot team is able to complete cooperative tasks successfully. A hybrid systems framework was used to model the dynamic role assignment, trying to provide a better and more formal way to represent the cooperative system. The methodology was tested in a cooperative transportation task and simulation results showed that the dynamic role assignment helps preventing deadlocks and allows the robots to perform the task successfully and efficiently.

Our future work is directed towards experimenting this methodology with other cooperative tasks both in simulated and real environments. We also intend to refine the description of the dynamic role assignment under a hybrid systems framework in order to provide a more formal approach to describe the execution of cooperative tasks by multi-robot teams.

Acknowledgments

Luiz Chaimowicz is partially supported by CAPES Foundation - Brazil and CNPq, and Mario F. M. Campos by CNPq.

References

[1] R. Alur, C. Coucoubetis, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[2] T. Balch and R. C. Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1):27–52, 1995.

[3] C. Castelpietra, L. Iocchi, D. Nardi, and R. Rosati. Coordination in multi-agent autonomous cognitive robotic systems. In *Proceedings of 2nd International Cognitive Robotics Workshop*, 2000.

[4] L. Chaimowicz, T. Sugar, V. Kumar, and M. Campos. An architecture for tightly coupled multi-robot cooperation. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, pages 2292–2297, 2001.

[5] B. Gerkey and M. Mataric. Principled communication for dynamic multi-robot task allocation. In D. Rus and S. Singh, editors, *Experimental Robotics VII, LNCIS 271*, pages 353–362. Springer Verlag, 2001.

[6] J. S. Jennings, G. Whelan, and W. F. Evans. Cooperative search and rescue with a team of mobile robots. In *Proceedings of the IEEE International Conference on Advanced Robotics*, pages 193–200, 1997.

[7] R. C. Kube and H. Zhang. Task modelling in collective robotics. *Autonomous Robots*, 4:53–72, 1997.

[8] F.-C. Lin and J. Y.-J. Hsu. Cooperation and deadlock-handling for an object-sorting task in a multi-agent robotic system. In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, pages 2580–2585, 1995.

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

[10] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence Journal*, 101(1-2):165–200, 1998.

[11] P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 1999.

[12] T. Sugar and V. Kumar. Metrics for analysis and optimization of grasps and fixtures. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, 2000.

[13] B. B. Werger and M. J. Mataric. Broadcast of local eligibility for multi-target observation. In Lynne E. Parker, George Bekey, and Jacob Barhen, editors, *Distributed Autonomous Robotic Systems 4*. Springer Verlag, 2000.