A Paradigm for Dynamic Coordination of Multiple Robots

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

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

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

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

Abstract

In this paper, we present a paradigm for coordinating multiple robots in the execution of cooperative tasks. The basic idea in the paper is to assign to each robot in the team, a role that determines its actions during the cooperation. The robots dynamically assume and exchange roles in a synchronized manner in order to perform the task successfully, adapting to unexpected events in the environment. We model this mechanism using a hybrid systems framework and apply it in different cooperative tasks: cooperative manipulation and cooperative search and transportation. Simulations and real experiments demonstrating the effectiveness of the proposed paradigm are presented.

1 Introduction

The coordination of multi-robot teams in dynamic environments is one of the fundamental problems in cooperative robotics: given a group of robots and a task to be performed in a dynamic environment, how to coordinate the robots in order to successfully complete the task? Coordination normally implies synchronizing robot actions and exchanging information among the robots. The amount of synchronization and communication depends heavily on task requirements, characteristics of the robots and the environment. Thus, different levels of coordination are required in different situations.

Let's consider a tightly coupled task such as cooperative manipulation, in which a team of robots must cooperate to transport or reposition a large object, as depicted in Figure 1. Common approaches to this task are based on force and form closure or object closure [17]. Techniques based on force and form closure normally require more stringent requirements concerning robot positioning and synchronization when compared to object closure approaches, as demonstrated in [11] and [15] for example. But, in both cases, robots must act in a synchronized fashion and this normally requires some knowledge about the states and actions of the teammates, either through implicit (sensory perception) or explicit communication. On the other hand, loosely coupled tasks such as foraging do not require much coordination and robots may act independently from each other.

A multi-robot coordination mechanism must cope with these different types of requirements. It should provide flexibility and adaptability, allowing multi-robot teams to execute tasks efficiently and robustly. To accomplish this, we propose a dynamic role assignment mechanism in which robots can dynamically change their behavior being able to successfully execute different types of cooperative tasks. We define a role to be a function that one or more robots perform during the execution of a cooperative task. Each role

Figure 1: Example of a tightly coupled task: cooperative manipulation.

can also be viewed as a behavior or a reactive controller. However, more generally, roles may define more elaborate functions of the robot state and on information about the environment and other robots including the history of these variables, and may encapsulate several behaviors or controllers. It not only dictates what controllers are used and how the state of a robot changes, but also how information flows between robots. By dynamically assuming and exchanging roles in a synchronized manner, the robots are able to perform cooperative tasks, adapting to unexpected events in the environment and improving their individual performance in benefit of the team.

In this paper, we use a hybrid systems framework to model cooperative tasks and dynamic role assignment between multiple robots. Hybrid systems are systems characterized by dynamics that are continuous at some levels and discrete at others. Hybrid systems explicitly capture the discrete and continuous dynamics in a unified framework, allowing us to model the interaction of these two types of dynamics. In this paper we use *hybrid automata* [9] to represent roles, role assignments, continuous controllers, and discrete variables related to each robot. The composition of these automata allows us to model the execution of cooperative tasks.

This paper is organized as follows. In the next section we discuss related work. Section 3 presents in detail our role assignment mechanism and our modeling using hybrid systems. Section 4 shows our experimental platform. In Section 5, we show some results of the role assignment mechanism in a cooperative manipulation task, in which robots coordinate themselves to transport a large object in an environment containing obstacles. Section 6 presents more results of the role assignment, this time in a cooperative search and transportation task, in which several simulated robots must find and transport together a large number of objects scattered in a certain area. Finally, Section 7 presents the conclusion and some directions for future work.

2 Related Work

There are many different approaches to multi-robot coordination, ranging from completely behavior based approaches [13] to three-layer architectures [18]. Our interest in this paper is on methodologies for role assignment (and other coordination mechanisms) for tackling the dynamic task allocation problem, *i.e.*, how to divide the work among the robots during the task execution, dynamically allocating subtasks and synchronizing their execution.

Several researchers have studied the task allocation problem, both for multi-agent software systems and distributed robots. The *Alliance* architecture [16], a behavior-based software architecture for heterogeneous multi-robot cooperation, has a fault tolerance mechanism that allows the robots to detect failures in their teammates and adapt their behaviors to complete the task. *Murdoch* [7] is a task allocation mechanism based on a publish/subscribe architecture in which robots broadcast messages with their resources and needs and dynamically allocate tasks based on this information exchange. A related market based approach, in which robots try to maximize their individual "profits", is presented in [6]. Another method is proposed in [10]: the *Method of Dynamic Teams* is a programming model which addresses the mapping of tasks into dynamic teams of agents. These teams can grow, shrink, and change dynamically, being programmed according to the task being executed. A comparative analysis of some of these and other approaches, including our dynamic role assignment mechanism, can be found in [8].

The term dynamic role assignment has also been used in [19] and others restricted to the multi-robot soccer domain. The roles in this domain are attacker, defender, etc., and, by switching roles and formations, the robots are able to modify their behavior in the field. As explained in this paper, we are interested in a more general and formal paradigm for role assignment that can be used in other types of tasks.

There are many papers on modeling cooperative tasks and coordination mechanisms that are relevant to this paper. A formal approach that uses a high-level modeling language for concurrent, multi-agent hybrid systems is presented in [1]. Another work that tries to formalize the execution of cooperative robotics is described in [12], in which finite state machines are used to encapsulate behaviors and discrete transitions are controlled by binary sensing predicates. Finite state machines do not model continuous aspects of the system but can help in modeling the discrete part of the cooperation. Petri Nets, that are commonly used for modeling multi-process systems, have also been used to model robotics tasks [14].

There are two main contributions in this paper. First, we propose the use of hybrid systems to provide a more formal approach to deal with multi-robot coordination. It allows the composition of different behaviors and the construction of hierarchies that facilitate the coordination. Another advantage is that hybrid systems lends itself to formal, control-theoretic descriptions of behaviors, potentially giving us the ability of performing a more elaborate analysis of coordination mechanisms via stability, convergence and reachability analysis. Finally, hybrid systems encompasses modern concepts of software engineering such as abstraction and modularity which simplify the development of programs for the coordination of multi-robot teams. The second contribution is related to the scope of the cooperative tasks. We are primarily interested in tasks

where physical interaction (e.g. manipulation) or maintaining continuous constraints (e.g. formation) is critical for cooperation. These tasks require a more formal approach to implement the robot controllers and the switching among them. Most of the approaches for task allocation do not specifically consider these aspects and, in this paper, we show that our framework provides a good way to address these issues.

3 Dynamic Role Assignment

3.1 Overview

In general, to execute cooperative tasks a team of robots must be coordinated: they have to synchronize their actions and exchange information. In our approach, each robot performs a role that determines its actions during the cooperative task. According to its internal state and information about the other robots and the task received through communication, a robot can dynamically change its role, adapting itself to changes and unexpected events in the environment. The mechanism for coordination is completely decentralized. Each robot has its own controllers and takes its own decisions based on local and global information. In general, each team member has to explicitly communicate with other robots to gather information but they normally do not need to construct a complete global state of the system for the cooperative execution. We consider that each team member has a specification of the possible actions that should be performed during each phase of the cooperation in order to complete the task. These actions must be specified and synchronized considering several aspects, such as robot properties, task requirements, and characteristics of the environment. The dynamic role assignment will be responsible for allocating the correct actions to each robot and synchronizing the cooperative execution.

Before describing in details the role assignment mechanism, it is necessary to define what is a role in a cooperative task. Webster's Dictionary 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 define a role to be a function that one or more robots perform 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. The role will define the behavior of the robot in that moment, including the set of controllers used by the robot, the information it sends and receives, and how it will react in the presence of dynamical and unexpected events.

The role assignment mechanism allows the robots to change their roles dynamically during the execution of the task, adapting their actions according to the information they have about the system and the task. Basically, there are three ways of changing roles during the execution of a cooperative task: the simplest way is the **Allocation**, in which a robot assumes a new role after finishing the execution of another role. In the **Reallocation** process, a robot interrupts the performance of one role and starts or continues the performance of another role. Finally, robots can **Exchange** their roles. In this case, two or more robots synchronize themselves and exchange their roles, each one assuming the role of one of the others. Sections

5 and 6 will show the use of all these types of role assignment in different cooperative tasks.

The role assignment mechanism depends directly on the information the robots have about the task, the environment and about their teammates. Part of this information, mainly the information concerning the task, is obtained a priori, before the start of the execution. The control software for each robot includes programs for each role it can assume. However, the definition of the task includes an *a priori* specification of the roles it can assume during the execution of the task and the conditions under which the role is reassigned or exchanged. The rest of the information used by the robots is obtained dynamically during the task execution and is composed by local and global parts. The local information consists of the robot's internal state and its perception about the environment. Global information contains data about the other robots and their view of the system and is normally received through explicit communication (message passing). A key issue is to determine the amount of global and local information necessary for the role assignment. This depends on the type of the task being performed. Tightly coupled tasks require a higher level of coordination and consequently a greater amount of information exchange. On the other hand, robots executing loosely coupled tasks normally do not need much global information because they can act more independently from each other.

Our approach allows for two types of explicit communication: synchronous and asynchronous. In synchronous communication, the messages are sent and received continuously in a constant rate, while in asynchronous communication an interruption is generated when a message is received. Synchronous messages are important in situations where the robots must receive constant updates about the state of the others. On the other hand, asynchronous communication is used for coordination when, for example, one robot needs to inform the others about unexpected events or discrete state changes such as the presence of obstacles, robot failures, etc.

An important point is to define when a robot should change its role. In the role allocation process, the robot detects that it has finished its role and assumes another available role. The possible role transitions are defined a priori and are modeled using a hybrid automaton as will be explained in the next section. In the reallocation process, the robots should know when to relinquish 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. The function μ can be computed based on local and global information and may be different for distinct robots, tasks and 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. An example of a utility function for a multi-robot task is presented in Section 6.1. It is also possible for two robots to exchange their roles. In this case, one robot assumes the role of the other. For this, the robots must agree to exchange roles and should synchronize the process, which is done using communication.

3.2 Modelling

The dynamic role assignment can be described and modelled in a more formal framework. In general, a cooperative multi-robot system can be described by its state ($\boldsymbol{X}$), which is a concatenation of the states of the individual robots:

$$\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}]^T. \tag{1}$$

Considering a simple control system, the state of each robot varies as a function of its continuous state ($\boldsymbol{x_i}$) and the input vector ($\boldsymbol{u_i}$). Also, each robot may receive information about the rest of the system ($\hat{\boldsymbol{z_i}}$) that can be used in the controller. This information consists of estimates of the state of the other robots that are received mainly through communication. We use the hat ($\hat{}$) notation to emphasize that this information is an estimate because the communication can suffer delays, failures, etc. Using the role assignment mechanism, in each moment each robot will be controlled by a different continuous equation according to its current role in the task. Therefore, we use the subscript q, $q = 1, \ldots, S$, to indicate the current role of the robot. Following this description, the state equation of each robot i, $i = 1, \ldots, n$, during the execution of the task can be defined as:

$$\dot{\boldsymbol{x_i}} = f_{i,q}(\boldsymbol{x_i}, \boldsymbol{u_i}, \hat{\boldsymbol{z_i}}). \tag{2}$$

Since each robot is associated with a control policy,

$$\boldsymbol{u_i} = g_{i,q}(\boldsymbol{x_i}, \hat{\boldsymbol{z_i}}), \tag{3}$$

and since $\hat{\boldsymbol{z_i}}$ is a function[1] of the state $\boldsymbol{X}$, we can rewrite the state equation:

$$\dot{\boldsymbol{x_i}} = f_{i,q}(\boldsymbol{X}), \tag{4}$$

or, for the whole team,

$$\dot{\boldsymbol{X}} = F_\Sigma(\boldsymbol{X}), \text{ where } F_\Sigma = [f_{1,q_1}, \ldots, f_{n,q_n}]^T, \; q_i \in \{1, \ldots, S\}. \tag{5}$$

The equations shown above model the continuous behavior of each robot and consequently the continuous behavior of the team during the execution of a cooperative task. These equations, together with the roles, role assignments, variables, communication and synchronization can be better understood and formally modeled using 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, or discretely, according to specific assignments. It is used to describe hybrid systems, i.e., systems that are composed by discrete and continuous

[1] Technically, we may allow estimators that depend on the history of evolution of the system state and model them as Markov processes. In this paper, we limit the scope of the treatment to reactive behaviors and estimators that are memoryless.

states. A hybrid automaton H can be defined as: $H = \{Q, V, E, f, Inv, G, Init, R\}$. $Q = \{1, 2, \ldots, S\}$ is the set of discrete states, also called *control modes*. The set V represents the variables of the system and can be composed by discrete (V_d) and continuous (V_c) variables: $V = V_d \cup V_c$. Each variable $v \in V$ has a value that is given by a function $\nu(v)$. This is called *valuation* of the variables. Thus, the state of the system is given by a pair (q, ν), composed by the discrete state $q \in Q$ and the valuation of the variables. The dynamics of the continuous variables are determined by the flows f, generally described as differential equations inside each control mode (f_q). Discrete transitions between pairs of control modes (p, q) are specified by the control switches E (also called *edges*). 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.

In our model, each role is a control mode of the hybrid automaton.[2] In fact, we can have hierarchical/sequential compositions in which a role is composed by other roles (control modes) as shown in Figure 2. Internal states and sensory information within each mode can be specified by continuous and discrete variables of the automaton. The variables are updated according to the equations inside each control mode (flows) and reset statements of each discrete transition. The role assignment is represented by discrete transitions and the invariants and guards define when each robot will assume a new role. Finally, we can model the cooperative task execution using a parallel composition of several automata as described in [3]. Table 1 summarizes the mapping from cooperative robotics to hybrid systems.

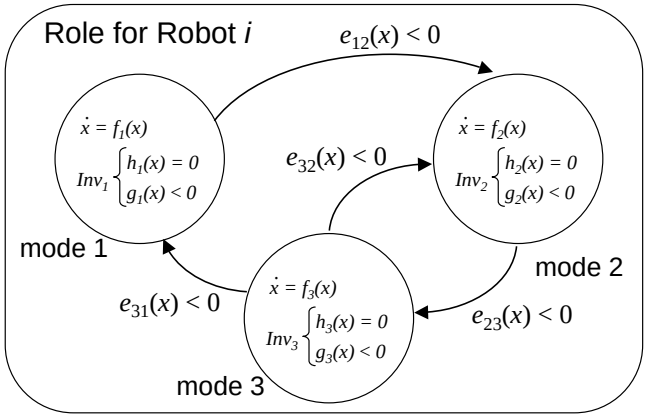


Figure 2: Schematic of a role composed by three roles (modes). Each mode is characterized by differential equations and algebraic constraints. The conditions under which transitions between modes occur and the conditions under which the robot stays in a particular mode are specified by the guards and invariants respectively.

[2]Since we are using control modes to represent roles, in the remainder of this paper we use the terms roles, modes and control modes with the same meaning.

Cooperative robotics	**Hybrid systems**
Roles	Control modes
Discrete and continuous information	Variables
Controllers	Flows
Role assignment	Discrete transitions, guards, and invariants
Communication	Send and receive actions, self transitions
Cooperative execution	Parallel composition of hybrid automata

Table 1: Modeling cooperative robotics using hybrid systems

Communication among robots can also be modeled in this framework. We use a message passing mechanism to exchange information among the agents. To model this message passing in a hybrid automaton, we consider that there are communication channels between agents and use the basic operations send and receive to manipulate messages. In the hybrid automaton, messages are sent and received in discrete transitions. These actions are modeled in the same way as assignments of values to variables (reset statements). It is very common to use a self transition, *i.e.*, a transition that does not change the discrete state, to receive and send messages.

We have chosen hybrid systems in order to represent cooperative robotics for two main reasons. The first one is that hybrid systems provides a formal framework for modeling the cooperative execution of tasks by multiple robots. To model cooperative robotics, it is necessary to represent both continuous and discrete dynamics together with synchronization and communication which are some of the main features of hybrid systems. Hybrid systems also encompasses modern concepts of software engineering such as abstraction and modularity which simplify the development of programs for the coordination of multi-robot teams. The second main reason is that the emerging theory for hybrid systems may allow the development of formal proofs about some aspects of the cooperative execution. For example, through an analysis of the hybrid systems model of a cooperative task it may be possible to detect deadlock states, test reachability of undesirable or goal states, and study stability of the cooperative system. Alternatively, as shown in [2], it may be possible to formally quantify and compare the performance with different algorithms. We do not explore these capabilities in this paper, but methodologies for obtaining some of these formal results are already available for some restricted classes of automata. With advances in hybrid systems theory and increase in availability of computational resources, we can expect that in the future we will have more advanced and efficient mechanisms for obtaining formal results from models based on general classes of hybrid automata.

4 Experimental Platform

As will be described in the next sections, we applied our role assignment mechanism in different cooperative tasks, both in simulated and real environments. Figure 1, showed in the introduction, depicts the robots used in our experiments. The robot on the left is a TRC Labmate platform, equipped with an actively controlled

compliant arm [20]. The platform is non-holonomic, and the only on-board sensors are encoders located at the arm and at the two actuated wheels. All the programming is done using Simulink and Real Time Workshop. The other robot is a XR4000, developed by Nomadic Technologies. It has a holonomic driving system offering three degrees of freedom (x, y, θ) and is equipped with several types of sensors, including two rings of 24 ultrasound and infrared sensors, a stereo vision system and several encoders. It is also equipped with a fork-lift arm that has one prismatic joint along a vertical axis. Both robots are equipped with wireless Ethernet boards and exchange messages using a connectionless protocol. This hardware motivates the paradigms and tasks used in this paper.

We have also developed MuRoS[3], a **Mu**lti-**Ro**bot **S**imulator 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 graphical user interface that allows the instantiation of different types of robots, the creation of obstacles and the observation of the simulation in real time. Also, result data can be exported to other tools such as Matlab for future analysis. The simulator can be extended with the development of new inherited classes defining new robots, controllers and sensors. Both implicit and explicit multi-robot communication can be simulated, allowing the robots to exchange information during the task execution. Robots also have sensing capabilities and the ability of building maps and plans in real time. 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.

In the next two sections, we will address two different cooperative tasks and illustrate our paradigm and methods using the experimental testbed and the simulation tool.

5 Cooperative Manipulation

In the cooperative manipulation task, a team of robots cooperate to carry a large object in an environment containing static and dynamic obstacles. Cooperative manipulation is a classical example of a tightly coupled task because it cannot be performed by a single robot working alone and requires a tight coordination to grasp and transport objects without dropping them. It is also a task where physical interaction among robots is critical, thus the the coordination mechanism should support that. In this section we present several results of the dynamic role assignment mechanism in a cooperative manipulation task. We start the section giving a brief description of the cooperative task and the leader-follower approach used. Then, we present the experiments and results obtained with simulations and real platforms.

5.1 Description

For executing the cooperative manipulation, we used a leader-follower architecture [5]. One robot is identified as a leader, while the others are designated as followers. The assigned leader has a planner and broadcasts

[3] http://www.verlab.dcc.ufmg.br/muros

its estimated position and velocity to all the followers using asynchronous messages. Each follower has its own trajectory controller that acts in order to cooperate with the leader. The planner and the trajectory controllers send set points to the low level controllers that are responsible for the actuators. All robots have a coordination module that controls the cooperative execution of the task. This module receives information from the sensors and exchanges synchronous messages with the other robots. It is responsible for the role assignment and for other decisions that directly affect the planners and trajectory controllers.

In this cooperative manipulation task, the robots can be executing one of the following roles: Dock, where they must coordinate themselves to approach and pick up the box; and Transport, where they march in a coordinated fashion. The Transport role is obtained by composing the roles Lead and Follow using *sequential composition* (discussed in Figure 2). Thus, a robot transporting a box will be performing either a leader role or a follower role. The control modes of the robots' automata are shown in Figure 3. The role assignment is used here mainly to exchange the leadership responsibilities among the robots: at any moment during transportation, the robot performing the leading role can become a follower, and any follower can take over the leadership of the team. There are two principle reasons for possible reassignment or exchange. First, leader-follower configurations are prone to instabilities when reversal of velocities are involved in non-holonomic systems. Thus, when reversing, it is better to have the previous trailing robot assume the leadership. Second, when robots' sensors are occluded or when one robot is better positioned than the other, it is conceivable that one robot is better suited to be the leader. The role assignment is also used for synchronizing actions, in such a way the robots are able to go from the dock role to the transport role in a coordinated manner.

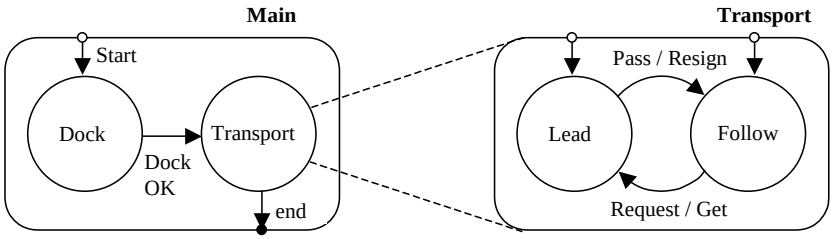


Figure 3: Possible roles in the execution of a cooperative manipulation task. Note that the Transport role is in fact composed by the Lead and Follow roles.

Different controllers and planners are used by each robot depending on its role in the task. They are the flows in each control mode of the hybrid automaton. To determine these flows and inputs it is necessary to consider the characteristics of each robot in each mode. In our experiment, when the robots are in the Dock mode, they use a proportional feedback controller based on the distance to the object to move in order to grasp the object. In the Transport mode, the robots have different behaviors when leading or following. In the Lead mode, they are controlled by planners that send set points to the actuators. In the following mode, the controllers are designed to enable the robots to follow a trajectory that is compatible with the leader's

in order to follow and cooperate with the leader.

Since the Labmate is non-holonomic, the inputs for its controllers are the linear and angular velocities ($u_1 = v$, $u_2 = \omega$). Thus, the state equations become (the subscript i is deleted for convenience):

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \rightarrow \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{6}$$

In the Follow mode, the robot uses information sent by the leader together with feedback information from the compliant arm to compute its own motion. As pointed out in [5], the compliant arm is used to compensate for errors in positioning caused by odometry errors and modeling errors. The controller in the Follow mode allows the follower to emulate the rear wheel of a bicycle in which the leader is the front wheel. More precisely, if the subscript l refers to information sent by the leader, D is the distance between the robots and $\boldsymbol{x}_a$ is the state of the compliant arm, the control laws are of the form:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \hat{v}_l \, cos(\hat{\theta}_l - \theta) + f(\boldsymbol{x}_a) \\ (\hat{v}_l / D) \, sin(\hat{\theta}_l - \theta) + g(\boldsymbol{x}_a) \end{bmatrix}. \tag{7}$$

The XR4000 is holonomic having three degrees of freedom and consequently three inputs ($\dot{x} = u_1$, $\dot{y} = u_2$, $\dot{\theta} = u_3$). When it is performing the Follow role, it tries to maintain a certain distance and orientation relative to the leader. The leader's pose ($\hat{x}_l, \hat{y}_l, \hat{\theta}_l$) and information about the leader's compliant arm ($\hat{\boldsymbol{x}}_a$) are received through explicit communication. For each degree of freedom, we use a proportional controller that tries to keep the robot at the desired position and orientation together with a function that is used to compensate odometry errors based on the position of the leader's compliant arm.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} k_1((\hat{x}_l + D\cos\theta) - x) + f(\hat{\boldsymbol{x}}_a) \\ k_2((\hat{y}_l + D\sin\theta) - y) + g(\hat{\boldsymbol{x}}_a) \\ k_3(\hat{\theta}_l - \theta) + h(\hat{\boldsymbol{x}}_a) \end{bmatrix}. \tag{8}$$

It is important to mention that the equations described above are specific for the case where we have one Labmate and one XR4000 performing the manipulation. In other cases the equations are different, but the basic idea is similar. We refer the reader to our previous work [5, 20] which discusses these controllers in greater detail.

As mentioned, the main purpose of the leadership exchange mechanism used here is to allow the robots to react and adapt easily to unexpected events such as obstacle detection and sensor failures. It is also important to assign the leadership to the appropriate robot in such a way that, in each phase of the cooperation, the robot that is best suited in terms of sensory power and manipulation capabilities will be leading the group. We implemented two methods for executing the leadership exchange under the role assignment paradigm: request and resignation. The main difference between them is the robot that starts the role exchange

process. In the leadership request, one of the followers sends a message requesting the leadership. This normally happens when one of the robots is not able to follow the leader's plan and/or knows a better way to lead the group in that moment. For example, if one of the followers detects an obstacle, it can request the leadership, avoid the obstacle, and then return the leadership to the previous leader. In the resignation process, the leader relinquishes the leadership to another robot. This can happen when the robot senses that it is unable to continue leading or when it finishes its leading turn in a task that has multiple leaders. The leadership can be offered to a specific robot or to all robots simultaneously.

Sometimes, the leadership exchange protocol may lead to conflicts that need to be resolved. Two examples in the small team with one leader are: (a) two or more followers request the leadership at the same time; and (b) a robot resigns its leadership, but there are no takers. A related problem is the possibility of a chattering phenomenon where changes in leadership occur too frequently. To avoid these types of conflicts it is necessary to use a priority based approach. The utility function μ can incorporate these priorities allowing the robots to decide by themselves how to resolve such conflicts. Another solution is to detect deadlocks (using timeout mechanisms for example) and, in such situations, relinquish the command to a human operator whose authority supersedes other robots. We use such approach in the experiments presented in this section.

5.2 Simulations

In the first simulation, four holonomic (XR4000) robots cooperate in order to carry an object from an initial position to the goal. In this experiment, we show the leadership request mechanism: one of the robots requests the leadership when it senses that it will not be able to follow the path determined by the leader. Figure 4 shows snapshots of the simulator during the task execution.

Snapshot (a) in Figure 4 shows the robots performing the Dock role. The robots are represented by circles and the object to be carried is the square in the middle of them. Each robot has a sensing area represented by a dashed circle around it. The other rectangles on the environment are obstacles and the goal is marked with a small x on the right of the figure. When they finish docking, they are allocated to the Transport role (snapshot (b)). To choose the initial leader, the robots exchange information and the robot that is closer to the goal (robot on the right, shown in black) is selected. As explained in the previous section, the leader has a trajectory to the goal and continuously sends its state information to the followers, that move in order to cooperate with it. But the current leader is unaware of the first obstacle, that is outside its sensing region. This obstacle is on the path of one of the followers (the robot on the bottom). When this robot senses the obstacle, it broadcasts a message requesting a role exchange. The leader receives this message and sends another message passing the leadership to the requester. The new leader is then able to avoid the obstacle and continue moving towards the goal, as shown in snapshot (c) in Figure 4. The same thing happens with the next two obstacles: the robot on the top assumes the leadership to avoid the second obstacle (snapshot (d)) and the robot on the bottom requests it again to avoid the last obstacle (snapshot (e)) before reaching the goal (snapshot (f)).

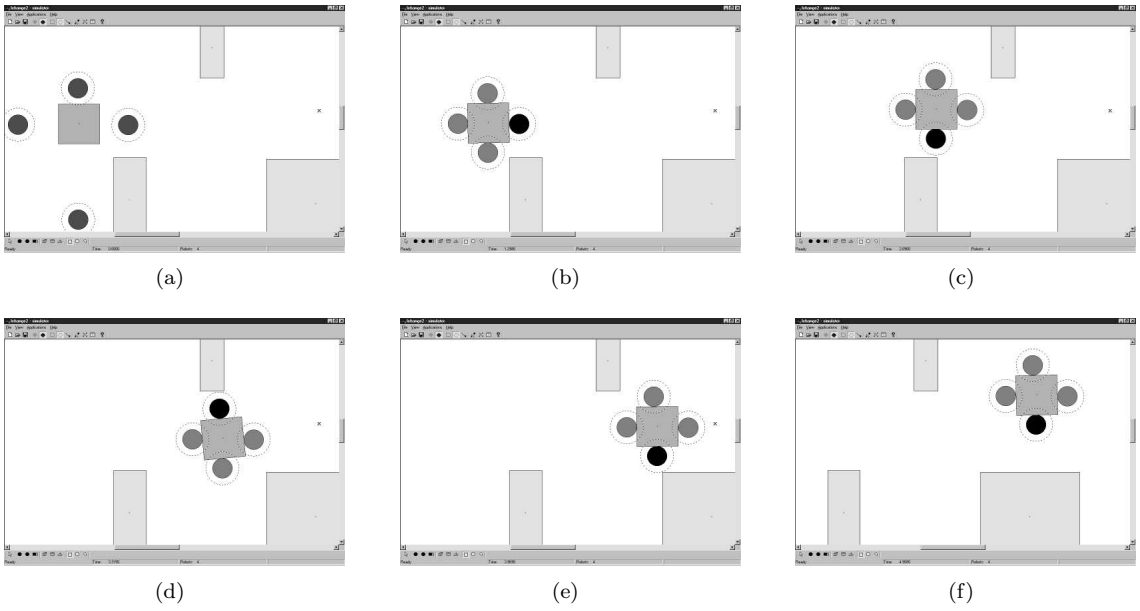


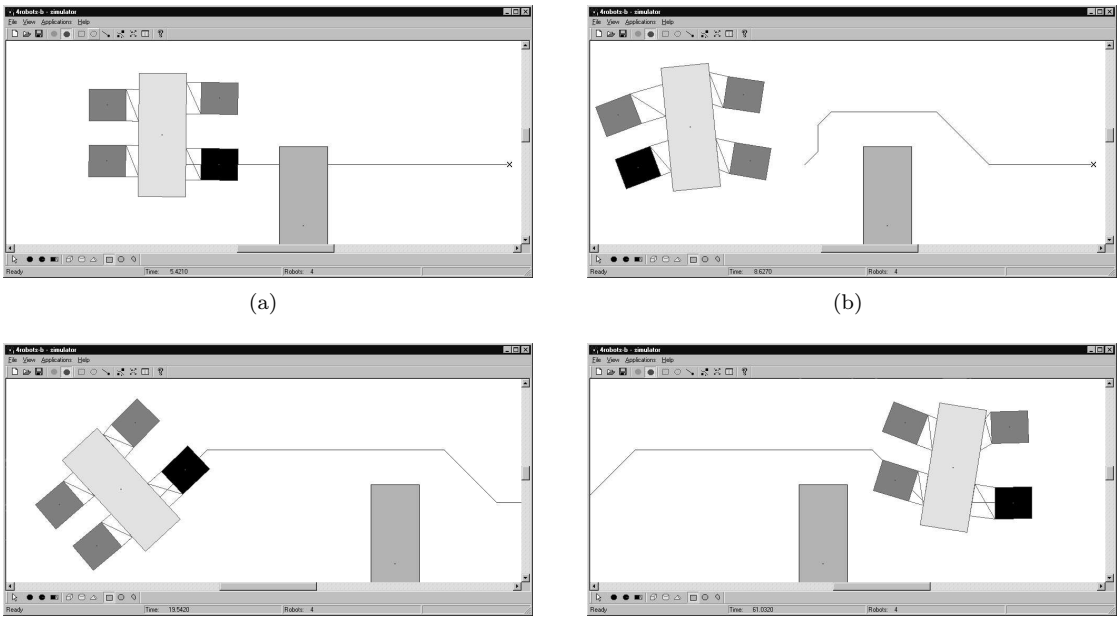
Figure 4: Leadership request with four holonomic robots transporting an object.

The second simulation shows the leadership resignation mechanism: the leader senses that it is not able to continue leading and resigns the leadership, that is accepted by one of the other robots. Figure 5 shows four nonholonomic (Labmate) robots manipulating an object using compliant arms. Snapshot (a) shows the robots already executing the Transport role. The robot on the bottom right is the initial leader and the line between the leader and the goal is the planned path. Note that the robot has not detected the obstacle yet, so it plans a straight line to the goal. When the leader detects the obstacle, it replans its path, and senses that it will not be able to continue leading due to its controller constraints (limitations on turning radius). So, it sends a message resigning the leadership. In this message, it includes other information, such as its planned path, the position of the obstacle, etc. This information is used by the other robots to choose the new leader. The robot on the bottom left is chosen and perform a backup maneuver in order to help the previous leader (snapshot (b)). After this maneuver, the robot returns the leadership to the previous leader (snapshot (c)) that replans its path and resumes the transportation avoiding the obstacle (snapshot (d)).

5.3 Experiments with Real Platforms

The robots shown in Figure 1 were used to implement the cooperative manipulation. Basically, they start the task performing the Dock role. The XR4000 uses its infrared sensor to approach the box and deploy its arm at the correct position. The Labmate uses feedback information from the compliant arm to move and grasp the box. In this experiment, docking is in one dimension (the robots and the box are aligned), and the Labmate waits until the XR4000 finishes before starting its own docking.

After finishing the docking phase, the robots are allocated to the Transport role. The first experiment



(a) (b)

(c) (d)

Figure 5: Leadership resignation with four non-holonomic robots transporting an object.

demonstrates the leadership resignation process. The robots' trajectories for this experiment are shown in Figure 6(a). The XR4000 begins leading (0X), followed by the Labmate (0L), until it detects an obstacle using its infrared sensor (1X). Then it sends a control message resigning the leadership to the Labmate. The new leader moves backwards in a curvilinear trajectory (from 1L to 2L), returning the leadership to the XR4000 (2X) when it finishes its plan. This experiment shows that, instead of trying to avoid the obstacle locally, which is difficult to accomplish while carrying a box in cooperation, the robots can exchange roles: the XR4000 offers the leadership to the Labmate, which takes it and modifies the trajectory. In this case, the modification is a simple open loop reversal with a turn. Note that during the execution the modes and controllers are changed dynamically due to the role exchange.

The second experiment (Figure 6(b)), shows the leadership request process. The Labmate begins leading by going backwards in a curvilinear trajectory (from 0L to 1L). The XR4000 begins following (0X) using its controller and requests the leadership when its infrared sensor detects an obstacle in its way (1X). After moving to avoid the obstacle, the XR4000 (2X) returns the leadership to the Labmate (2L) that leads until the end of the task. The leadership exchange here is very important because the leader is not aware of the obstacle in the path of the follower. The XR4000 therefore requests the leadership, avoids the obstacle, and returns the leadership to the Labmate.

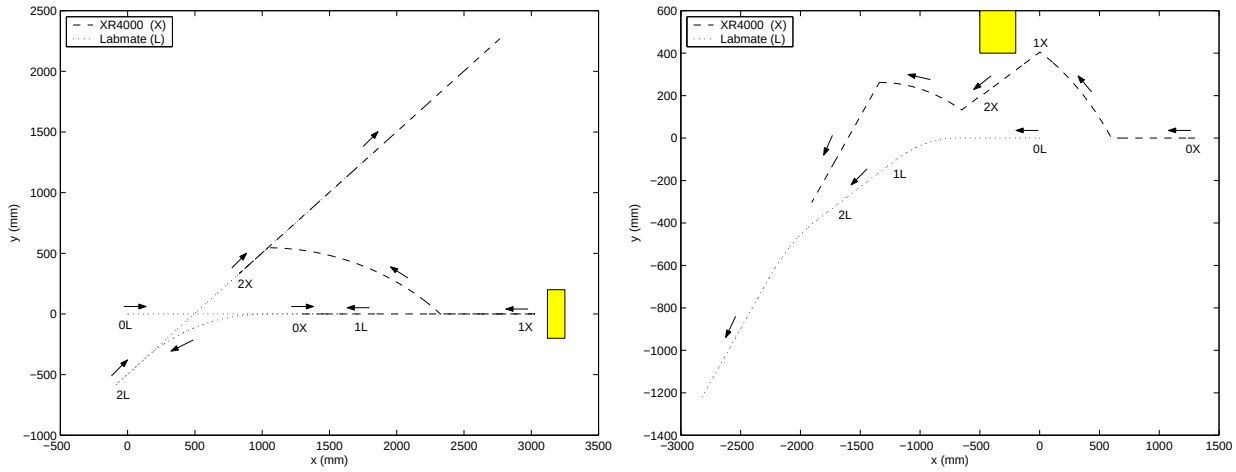


Figure 6: Experiments: (a) the XR4000 resigns the leadership (1X) and receives it back (2X); (b) the XR4000 requests the leadership (1X) and returns it (2X) to the Labmate.

6 Cooperative Search and Transportation

In this section, our role assignment mechanism is demonstrated in a Cooperative Search and Transportation task, in which a group of robots must find and cooperatively transport several objects scattered in the environment [4]. It is a combination of a loosely coupled task, where the robots search the area looking independently for objects, and a tightly coupled task, in which the robots must manipulate objects in cooperation. We use the cooperative search and transportation mainly to demonstrate the role reallocation mechanism, showing that its use can improve the task performance and avoid deadlocks.

6.1 Description

The cooperative search and transportation task 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 j requires at least k robots ($1 < k \leq n$) to be transported and has a importance value v. Thus, each object can be described by a pair $\{k, v\}$, representing respectively the amount of work (in terms of number of robots) and the reward for that object. Differently from a common foraging task, in which the robots can act independently from each other and communication is not strictly necessary, this task requires the robots to coordinate themselves in order to transport the objects in cooperation.

The coordination of the robots is done using the role assignment mechanism. Basically, there are five different roles in the cooperative search and transportation. As shown in Figure 7, in each of these roles, the robots compute a utility μ that is used for reallocating roles. Initially, all robots start in the Explore mode, in which they randomly move in the environment searching for objects to be transported. When a robot

detects an object inside its sensing area, it finishes its exploration role and starts the Attach role, in which it approaches the object preparing to transport it. But, if a robot is the first one to attach to an specific object, it assumes the Attach Lead role instead. 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 to transport that object to the goal. 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 like a bidding process[4], in which the 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 an approaching robot detects an object inside its sensor range, it assumes the Attach role and when the number of robots necessary to carry the object is sufficient, they assume the Transport role and cooperatively move the object to the goal. It is important to mention that, 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. We use c_j to represent the number of committed robots for a certain object j. If a committed robot reallocates to another role, it must send a message to the leader resigning the previous role. We are not considering the possibility of robot and communication failures in these experiments.

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. Figure 7 shows the hybrid automaton for the cooperative search and transportation. For clarity, only the roles (top level modes) and transitions between roles (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.

As mentioned, a robot performing a role r reallocates to another role r' when the difference $\mu_{r'} - \mu_r$ is greater than a threshold τ. There are four role reallocations in this diagram: the first one is when an explorer volunteers and is recruited to approach a certain 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 last position in a local memory in order to return to this object after finishing the new role. Also, a robot that is approaching can become a 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 that is possible is when a robot approaching an object j_1 reallocates to approach a different object j_2. In this case, the robot will be performing the same role but with a different parameter.

The choice of a suitable function to measure the role utilities is a fundamental aspect of this task. The

[4]The bidding process is relatively simple. There is a single auctioneer for each object (the Attach Leader) and all robots are considered, but only robots for which the gain in utility is greater that the threshold τ bid. The process is repeated until the number of robots is sufficient to transport the object.

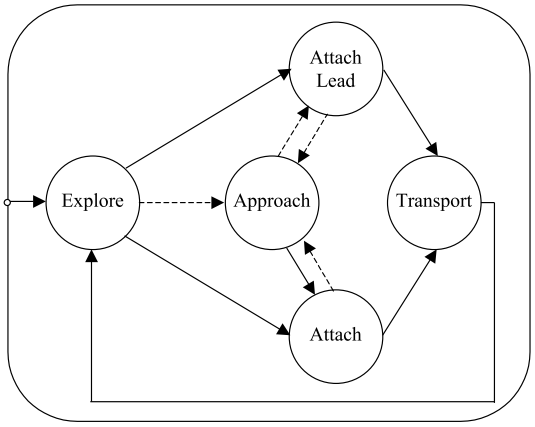


Figure 7: Roles and role assignments for the cooperative search and transportation task.

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 search and transportation, for example minimize execution time or maximize the value in a shorter time, different utility functions can be implemented. In the experiments presented here we use a simple heuristic function in order to test the execution of the role assignment mechanism. We do not intend to compare different functions or search for optimal results. Instead, we just want to provide a simple testbed 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 beyond the scope of this paper.

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 any object have the highest utility (1). The other roles (Attach, Attach Lead and Approach) depend on the target object j. We have defined an utility function that balances the value of the object (v_j), the number of robots that are still necessary to start its transportation ($k_j - c_j$) and a function of the distance from the robot to the object ($f(d_{ij})$). This value is multiplied by a constant α so the maximum value of this function is 1. Thus, the utility for a robot i to perform a role r with a target object j is given by:

$$\mu = \begin{cases} 0, & r = \text{Explore}, \\ 1, & r = \text{Transport}, \\ \alpha(\frac{v_j^2}{k_j - c_j + 1} + \frac{1}{f(d_{ij})}), & otherwise. \end{cases} \tag{9}$$

Using this heuristic function, each robot tries to maximize the value recovered in a short time but also gives priority to objects that need fewer 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.

6.2 Simulations

The dynamic role assignment in the cooperative search and transportation task was tested using MuRoS. The experiments were performed using 20 holonomic robots and 30 objects randomly distributed in the environment. The dimensions of the search area and the goal region are 30 by 30 meters and 4 by 4 meters respectively. Comparatively, the diameter of each robot is 30 centimeters. The value of each object (v) and the number of robots necessary to transport it (k) were generated randomly, with $v = \{1, \ldots, 5\}$ and $k = \{2, \ldots, 5\}$. The utility function described above was used varying the threshold τ from 0 to 0.8. For each value of τ, 100 runs were performed and the mean values were computed.

Firstly, the time to complete the task was measured. The results are presented in the left graph of Figure 8. The graph shows that the completion time starts to increase for values of τ greater than 0.2. 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 level of cooperation decreases since each robot only attaches to the objects detected by itself, not accepting offers from its teammates. Consequently, the time to gather the k robots necessary to transport each object will increase, also increasing the overall time to complete the task.

The extreme case of this work division causes deadlocks. A deadlock occurs when all robots are performing the Attach role to specific objects, but the number of robots attached to each object is not sufficient to transport it. In this case, the robots keep waiting indefinitely and do not complete the task. The graph on the right of Figure 8 shows the number of deadlocks for each value of τ in 100 experimental runs. More than 50% of the runs with large values of τ results in deadlocks.

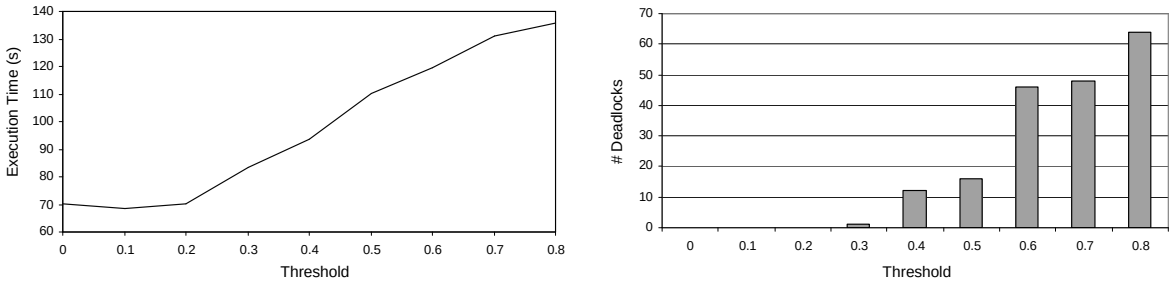


Figure 8: Completion time and number of deadlocks varying the threshold τ.

In this task, deadlocks are detected by a timeout period, *i.e.*, if the robots do not complete the task within a certain amount of time then a deadlock has occurred. The timeout mechanism works fine in this situation, where the only cause for not completing the task is the presence of a deadlock. But more elaborated mechanisms to detect deadlocks can be used in general situations. An example is the use of a supervisory

process that monitors the global state of the system and detects if it is changing as time flows. In the cooperative search and transportation, the global state can be a set composed by the current role plus the position of each robot. If there are no role assignments and the robots do not move within a fixed period of time, then the global state is not modified and the system is deadlocked.

Observing the results presented in this section, it can be seen that the dynamic role assignment allows the successful execution of the cooperative search and transportation task. In this specific case, the use of the role assignment with a suitable utility function and adequate threshold values brought good performance in terms of time and other metrics while avoiding deadlocks that would prevent the task completion.

7 Conclusion

This paper presented a role assignment paradigm 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, adapting to unexpected situations and improving their performance. The formulation is based on a hybrid systems framework, with control modes used to model the different roles and transitions between modes allowing for dynamic role assignment. Different types of cooperative tasks were performed both in real and simulated environments to show the applicability of the proposed mechanism with empirical results demonstrating the effectiveness of the paradigm.

As mentioned, we are primarily interested in tasks where physical interaction or maintaining continuous constraints is critical for cooperation. Such tasks necessitate a more formal approach for modeling, developing and implementing robot controllers and behaviors. We believe that the dynamic role assignment mechanism presented here specially address these issues. The hybrid systems model gives us a more formal approach to deal with these requirements allowing the implementation of a more stringent coordination mechanism.

The work also raises many interesting questions. A key issue is the choice of utility functions for the role reallocation. While we selected meaningful functions in our experiments, the choice of the functions was largely driven by heuristics. While one could apply formal optimization techniques to derive optimal roles for simple tasks, it is difficult to see how such an approach might work for cooperative search, manipulation and transportation. Empirical approaches to deriving good utility functions may well be the best approach.

Another avenue for further investigation is to see if the hybrid systems modeling paradigm might lend itself to validation and verification. Our preliminary work [2] suggests that with appropriate simplifications on the environment (cell decomposition), verification techniques can be used to compare different strategies with some simplifications on the dynamics. It is difficult to predict if this work can be extended to complex tasks such as the ones considered in this paper, primarily because nonlinearities such as the ones introduced by angular dependency are not handled well by current verification tools. However, the use of some probabilistic approaches may lead to satisfactory results.

References

[1] R. Alur, A. Das, J. Esposito, R. Fierro, G. Grudic, Y. Hur, V. Kumar, I. Lee, J. Ostrowski, G. Pappas, B. Southall, J. Spletzer, and C. Taylor. A framework and architecture for multirobot coordination. In D. Rus and S. Singh, editors, *Experimental Robotics VII, LNCIS 271*. Springer Verlag, 2001.

[2] R. Alur, J. Esposito, M.-J. Kim, V. Kumar, and I. Lee. Formal modeling and analysis of hybrid systems: A case study in multirobot coordination. In *Proc. of the World Congress on Formal Methods*, 1999.

[3] L. Chaimowicz. *Dynamic Coordination of Cooperative Robots: A Hybrid Systems Approach*. PhD thesis, Universidade Federal de Minas Gerais - Brazil, June 2002. Available at: http://www.cis.upenn.edu/~chaimo/thesis.pdf.

[4] L. Chaimowicz, M. Campos, and V. Kumar. Dynamic role assignment for cooperative robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 292–298, 2002.

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

[6] M. B. Dias and A. Stentz. A market approach to multirobot coordination. Technical Report CMU-RI -TR-01-26, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2001.

[7] B. Gerkey and M. Mataric. Sold! auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, October 2002.

[8] B. Gerkey and M. Mataric. A framework for studying multi-robot task allocation. In *Proc. of the Int. Workshop on Multi-Robot Systems*, pages 15–26, 2003.

[9] T. Henzinger. The theory of hybrid automata. In *Proc. of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292, 1996.

[10] J. Jennings and C. Kirkwood-Watts. Distributed mobile robots by the method of dynamic teams. In *Distributed Autonomous Robotic Systems 3*. Springer Verlag, 1998.

[11] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, A. Casal, and A. Baader. Force strategies for cooperative tasks in multiple mobile manipulation systems. In *Proc. of the 7th Int. Symp. on Robotics Research*, pages 333–342, 1995.

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

[13] M. Mataric. *Interaction and Intelligent Behavior*. PhD thesis, MIT, 1994.

[14] D. Milutinovic and P. Lima. Petri net models of robotic tasks. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 4059–4064, 2002.

[15] A. Ollennu, H. Nayar, H. Aghazarian, A. Ganino, P. Pirjanian, B. Kennedy, T. Huntsberger, and P. Schenker. Mars rover pair cooperatively transporting a long payload. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 3136–3141, 2002.

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

[17] G. A. S. Pereira, V. Kumar, and M. F. M. Campos. Decentralized algorithms for multi-robot manipulation via caging. *International Journal of Robotics Research (to appear)*, 2003.

[18] R. Simmons, T. Smith, M. B. Dias, D. Goldberg, D. Hershberger, and R. Zlot A. Stentz. A layered architecture for coordination of mobile robots. In A. Schultz and L. Parker, editors, *Multi-Robot Systems: From Swarms to Intelligent Automata*. Kluwer, 2002.

[19] P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, 1999.

[20] T. Sugar and V. Kumar. Control of cooperating mobile manipulators. *IEEE Transactions on Robotics and Automation*, 18(1):94–103, 2002.