A Framework for Coordinated Control of Multi-Agent Systems

by

Howard Li

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2006

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.
I understand that my thesis may be made electronically available to the public.

Howard Li

Abstract

Multi-agent systems represent a group of agents that cooperate to solve common tasks in a dynamic environment. Multi-agent control systems have been widely studied in the past few years. The control of multi-agent systems relates to synthesizing control schemes for systems which are inherently distributed and composed of multiple interacting entities. Because of the wide applications of multi-agent theories in large and complex control systems, it is necessary to develop a framework to simplify the process of developing control schemes for multi-agent systems.

In this study, a framework is proposed for the distributed control and coordination of multi-agent systems. In the proposed framework, the control of multi-agent systems is regarded as achieving decentralized control and coordination of agents. Each agent is modeled as a Coordinated Hybrid Agent (CHA) which is composed of an intelligent coordination layer and a hybrid control layer. The intelligent coordination layer takes the coordination input, plant input and workspace input. After processing the coordination primitives, the intelligent coordination layer outputs the desired action to the hybrid layer. In the proposed framework, we describe the coordination mechanism in a domain-independent way, as simple abstract primitives in a coordination rule base for certain dependency relationships between the activities of different agents. The intelligent coordination layer deals with the planning, coordination, decision-making and computation of the agent. The hybrid control layer of the proposed framework takes the output of the intelligent coordination layer and generates discrete and continuous control signals to control the overall process. In order to verify the feasibility of the proposed framework, experiments for both heterogeneous and homogeneous Multi-Agent Systems (MASs) are implemented. In addition, the stability of systems modeled using the proposed framework is also analyzed. The conditions for asymptotic stability and exponential stability of a CHA system are given.

In order to optimize a Multi-Agent System (MAS), a hybrid approach is proposed to address the optimization problem for a MAS modeled using the CHA framework. Both the event-driven dynamics and time-driven dynamics are included for the formulation of the optimization problem. A generic formula is given for the optimization of the framework. A direct identification algorithm is also discussed to solve the optimization problem.

Acknowledgments

First of all, I would like to acknowledge the contributions of all the people that have provided invaluable help and support for the completion of this thesis.

My supervisors, Professor Fakhri Karray and Professor Otman Basir, deserve thanks for not only guiding me with a steady hand, but also leaving me the freedom to pursue my own research interests. They have generously given me invaluable suggestions on my research. They guided me to conduct scientific research as well as theoretical studies. Without their directing and support, I would not have finished my thesis research. I thank you for your patience and encouragement. You opened up a whole new world for me.

I Have to express my appreciation to Professor Mohamed A. Zohdy of Oakland University for serving as my external examiner. Many thanks to Professor Kumaraswamy Ponnambalam, Professor Hamid Tizhoosh, Professor Magdy Salama for serving in the exam committee. Your comments and suggestions have greatly improved my work.

The stimulating discussions in late night sessions with my lab mates have provided an energy boost when things seemed going nowhere. I greatly appreciate their friendship that has made my work more enjoyable. The amazing faculty and staff at the Department of Electrical and Computer Engineering and the Department of Systems Design Engineering at the University of Waterloo are appreciated for all the document submission reminders, seminar notices, and ready hands to help me. I also thank you for being great friends after work. It seems that there are get-together events all year around, and I never feel bored here.

I really enjoy the friendly and supporting environment here at the University of Waterloo. The University of Waterloo has provided a supportive and enjoyable environment in which students can study and work. I thank all the faculty, staff and students here sincerely.

My thanks must also go to the Natural Sciences and Engineering Research Council (NSERC) of Canada for awarding me the PGS-B scholarship and providing the funding for this research through research grants to my supervisors.

On a personal front, I am forever indebted to my family for my continued study. Special thanks to my parents, my brother and my sister-in-law who have provided both moral and financial support over the period of my education. Without all your love and support, I

wouldn't have been able to finish this thesis.

There have been many people who have actively contributed to this project, all their help and support are appreciated.

Contents

List of Tables

List of Figures

List of Acronyms

AI	Artificial Intelligence
AL	Artificial Life
COG	Center Of Gravity
CHA	Coordinated Hybrid Agent
CORBA	Common Object Request Broker Architecture
DAI	Distributed Artificial Intelligence
DES	Discrete Event System
DESs	Discrete Event Systems
HICA	Hybrid Intelligent Control Agent
MAS	Multi-Agent System
MASs	Multi-Agent Systems

Chapter 1

Introduction

Modern control systems must meet the requirements of significant degrees of dynamic environments to provide greater flexibility. Distributed Artificial Intelligence (DAI) is a sub-discipline of Artificial Intelligence (AI) that deals with problems requiring a distributed approach to effective practical solutions [5]. The implementation of complex AI systems can be approached by decomposing the global goal into simpler, well-specified tasks which are easier to be accomplished independently by a collection of interacting and autonomous components (i.e., agents). It is proved in [32] that agent-oriented approaches are well suited to the engineering of complex control systems. Multi-Agent Systems (MASs) represent a group of agents working cooperatively to solve common tasks in a dynamic environment. The control of MASs relates to synthesizing control schemes for systems which are inherently distributed and composed of multiple interacting entities.

1.1 Motivation

The control of large complex robotics and manufacturing systems require autonomous cooperating or coordinated multiple robots and other platforms to work together, where the term coordinated refers to tight coupling of the physical platform's kinematics and dynamic parameters. The control of multiple platforms is very different from that of a single platform. The environment is not static because all the other platforms are reacting in the environment at the same time. Many reported approaches usually are not generalized.

And a lot of them can not be applied to both homogeneous systems and heterogeneous systems. Agent activities need to be analyzed at both the strategic level and the tactical level that involves platforms' kinematics and dynamics.

1.1.1 Problem Statement

The tasks of cooperative robots, such as grasping, manipulation, lifting, dropping, and handling, require close and simultaneous coordination of all the robots.

Scenario 1.1.1 *Consider the control of a multi-crane system composed of two industrial overhead cranes operating in a shared workspace. The goal of this multi-crane system is to control the two cranes to move the payloads in the shared workspace without any collision. The overhead cranes are hybrid systems that have both discrete events and continuous dynamics. The two hybrid systems need to interact with each other in order to achieve a global goal.*

In the above scenario, the two overhead cranes have similar properties. A more interesting and more challenging case is the control of a heterogeneous Multi-Agent System (MAS). In this scenario, three systems with different properties need to cooperate with others.

Scenario 1.1.2 *Consider the control of a system consisting of a mobile robot, an overhead crane, and a robot manipulator. The mobile robot is a flexible, robust platform for either indoor or outdoor experiments and applications. The robot manipulator can provide six degrees of freedom. The goal for this control system is to develop cooperative tasks among the overhead crane, the mobile robot and the robot manipulator. The mobile robot picks up an object in the overhead crane's workspace and carries it to the manipulator's workspace. The robot manipulator is mounted on a track which is an extra axis of control for the robot manipulator. It picks up the object from the mobile robot and delivers it to the other end of its track.*

One thing that is common with these scenarios is to control multiple hybrid systems to cooperate and coordinate in order to achieve the global goal. Other examples can be the

distributed control of a reservoir system. The agent-based approach can be applied for distributed control of multiple hybrid systems. In a MAS, agents have various specializations for the subtasks. Individual agents can be implemented by non-adaptive techniques and they may also have learning capabilities. The basic functionality is mostly encapsulated in individual agents. Agents represent their own abilities locally, and the whole system becomes goal oriented.

1.1.2 The Control of Multi-Agent Systems

Multi-agent control systems relate to the decomposition of complex control problems so that multiple local controllers can solve the problem cooperatively. On the other hand, the control of MASs relate to synthesizing control schemes that are inherently distributed and consist of multiple entities. The control of MASs have been widely studied in the past few years.

In [8], an agent-based approach for distributed control systems is proposed which is adaptable and dynamically reconfigurable. The approach makes use of distributed artificial intelligence tools at both the planning and the control levels. In [3], the development and implementation of an agent-based distributed control system in a waste water treatment plant are introduced. In [16], the authors study a simplified version of the RoboFlag competition that they model as a hybrid system. In [13], a distributed algorithm for coordinating the flow of a mass of vehicles approaching a highway exit or a tollbooth is studied. An approach to detect and diagnose multiple faults in industrial processes with a hybrid multi-agent diagnostic system is presented in [23]. A method is proposed in [34] where the programs that identify the conditions of a specific type of data are defined and integrated by means of a multi-agent architecture. A dispatching control system for flexible manufacturing systems is presented in [31]. In [52], implicit communication is used to address the problem of coordination of multiple mobile robots. In [42], an extended Kalman filter-based algorithm for the localization of a team of robots is described. Coordinating the motions of multiple robots operating in a shared workspace without collisions is addressed in [1] for the coordination of multiple robots when their trajectories are specified. In [44], a negotiation protocol for verifying the feasibility of a cooperative task is proposed. In [66], the object closure method is defined and a set of decentralized algorithms are

developed to allow the robots to achieve the object closure. In [25], Petri nets are used to evaluate the efficiency of the MAS. In [43], it is described that the use of behaviors as the underlying control representation provides useful encoding that both lends robustness to control and allows abstraction for handling scaling in learning that focuses on multi-agent robot systems.

1.1.3 The Architecture for Multi-Agent Systems

Because of the wide applications of multi-agent theories in large and complex control systems, it is necessary to develop a framework to simplify the process of developing control schemes for MASs. The control architecture for MASs can be broadly characterized as deliberative control, reactive control, and a combination of both of them. Deliberative control is based on planning, while reactive control is based on coupling between sensing and actuation. Strategies which require that action be mediated by some symbolic representation of the environment are often called deliberative. In contrast, reactive strategies do not exhibit a steadfast reliance on internal models. Instead of responding to entities within a model, the control system can respond directly to perception of the real world.

Deliberative Control

An architecture for multi-robot systems is proposed in [55] that considers cooperation as an opportunity to increase the skills of robots which already possess some capabilities. In this architecture, several modules are defined for an agent. However, this framework only deals with the domain of a multi-robot system.

An architecture for MASs and the application of the architecture for the control of an autonomous mobile robot are introduced in [59]. A knowledge source and several layers are defined in this architecture. However, this framework only deals with the domain of mobile robots. Furthermore, more modules should be included in this architecture to make it generic.

In [53], the synthesis of a multi-agent supervisor for a multi-agent framework is studied. It is proved that there is an algorithmic procedure for the recursive construction of a multi-agent supervisor when an additional automaton is added to a system. This

work concentrates on the research of supervising a MAS rather than defining a generic framework.

In [27], an algebra for computing overall survivability from dimensions of success is studied. A control infrastructure is presented that leverages the degrees of freedom to make run-time adaptations at multiple hierarchical levels to maximize overall system survivability. But this framework concentrates on the system survivability rather than the control of the system.

Information technology has made today's manufacturing systems increasingly distributed. Such systems consist of a complex array of computer-based decision units, controllers, and databases. In [57], a multi-agent framework is presented for achieving system integration. This work specifically emphasizes the coordination mechanisms needed for ensuring the orderly operations and concerted decision making among the agents of the manufacturing systems. The application of the framework to a printed circuit board manufacturing system and the performance results are also described. While successful in many ways, this framework does not include the hybrid nature and the continuous dynamics of a system that always exist in complex manufacturing systems.

A multi-agent framework is proposed in [64] to develop product design and planning using the concurrent engineering approach. The model brings together constraints from all team members in the development cycle. The methodology uses conflict-resolution techniques and design-improvement suggestions to refine the initial product design and process plan generation. Simulation is used to verify the feasibility of the design. This framework focuses on providing a way of modeling design teams. It presents a way to model an individual team member's perspective as a segment of the task knowledge. It is not applicable to the control of MASs.

DAI lacks a clear and implementable model for cooperative problem solving, which specifies how agents should operate and interact in complex, dynamic and unpredictable environments. In order to address this problem with DAI, a new cooperation model has been developed in [32]. This model specifies pre-conditions which must be attained before collaboration can start and prescribes how individuals should behave when the joint activity is proceeding and also when the joint activity runs into difficulty. This model has been used to guide the implementation of a general-purpose cooperation framework and the

qualitative and quantitative benefits of this implementation have been assessed through a series of comparative experiments in the real-world domain of electricity transportation management. This framework deals with the cooperation issues of MASs, but it doesn't include the hybrid characteristics of a MAS either.

In [26], the concept of agent is applied to implement control algorithms and application architectures for flexible control in manufacturing systems. A framework is presented and demonstrated by running a simulated manufacturing system by a set of agents. The requirements for the implementation and overview of the building blocks of the agent architecture are given. The persistent data management facilities are included into the agent architecture which allows an agent to handle information and knowledge. This framework focuses on the overall management of the MAS, it lacks the ability to deal with the dynamics and uncertainty of the environment.

In [29], the traditional framework for discrete-event control is extended to include the case of control with active events, in which both the user and the environment have events that they can trigger. A synthesis algorithm is outlined for minimally restrictive controllers. Multiuser systems are also discussed. This work focuses on the control of discrete events. It provides a model for multi-users to generate events. It is not a generalized framework that can be applied to the control of MASs.

A stable control strategy is presented in [45] for groups of vehicles to move and reconfigure cooperatively in response to a sensed, distributed environment. Each vehicle in the MAS serves as a mobile sensor and the vehicle network as a mobile and reconfigurable sensor array. The underlying coordination framework uses virtual bodies and artificial potentials, which is based on gradient climbing missions in which the mobile sensor network looks for local maxima or minima in the environmental field. The network adapts its configuration in response to the sensed environment in order to optimize its gradient climb. This framework focuses on the mobile sensor network problem, which makes it problem dependent and not suitable for other domains.

A strategy is described in [58], in which agents attempt to make claims using tactical rules. As the debate among agents continues, a shared argument map is created, which is controlled by strategic rules. Inconsistency is assumed to stand unless it is attacked by another agent. An evaluation function calculates the strength of arguments in terms of

a number of structural constraints. This work provides a strategy for decision making, rather than a strategy for the control of a MAS.

Motivated by the distinction between controllable and uncontrollable events, in [6], the difference between two types of agents within a MAS are defined. The agents are distinguished as controllable agents and uncontrollable agents. Controllable agents represent agents that are directly controlled by the system's designer. Uncontrollable agents represent agents that are not under the designer's direct control. Such systems are referred to as partially controlled MASs. In this work, it is investigated how one might influence the behavior of the uncontrolled agents through appropriate design of the controllable agents. Different techniques are presented for controlling agents' behaviors in various domains. This work focuses on controlling agents to learn and affect other agents (for example, opponents in a game) in the environment. It does not provide a framework for the control of MASs either.

Nowadays, manufacturing systems are required to be capable of responding rapidly to dynamic changes, and fulfilling customer needs in order to rival with business competitors. There is a demand for the integration of process planning and production scheduling. A multi-agent based framework is introduced in [36] in which process planning and production scheduling are integrated. In addition, this framework enables the utilization of manufacturing resources to be dynamically optimized as well as provide a platform on which alternative configurations of manufacturing systems can be assessed. This work focuses on the scheduling of MASs which covers only one aspect of developing a generic framework. The hybrid nature of such an agent is not addressed.

Based on the analysis of the reinforcement learning and Markov games, a layered multi-agent coordination framework is proposed in [18]. Based on agents' interaction of competition and cooperation, this coordination framework adopts the zero-sum Markov game in higher layer to compete with opponent and adopts the team Markov game in lower layer to accomplish the team's cooperation. This coordination framework is applied to Robot Soccer. The learning issue for a MAS is studied in this framework, while a comprehensive and generic framework for the control of MASs still needs to be studied.

Reactive Control

Behavior-based control appears to be a popular approach to offer a practical level of flexibility, autonomy, and computational economy for preliminary design of MASs. It is based on the reactive control strategy [9].

An agent's architecture is proposed in [38] for multi-robot systems. This architecture is based on satisfaction and altruism allowing the agents to amend their low-level behavior in order to solve more complex problems. Unfortunately, rather than defining a complete model for MASs, this architecture focuses on the "altruistic" reactions.

In [47], a modeling concept is presented that relies on a multi-agent based modeling framework, which decomposes the control problem into highly autonomous entities. These entities communicate via abstract sensors and actuators. However, this work is focused on the communication of MASs. As a result, it is targeted on measuring systems that are networked with multi-processors.

In [62], a framework is introduced that provides distributed control of large collections of mobile physical agents in sensor networks. The agents sense and react to virtual forces. This framework also provides an effective basis for self-organization, fault-tolerance, and self-repair. Examples show how this framework can be applied to construct distributed sensing grids, and dynamic behaviors for perimeter defense and surveillance. It is also analyzed how to facilitate system understanding and predictability. This frame work focuses on the application of sensing and surveillance, which is not applicable to the problems that we focus on.

Distributed control offers robustness, scalability, and simplicity to the control and organization of module based systems. In [35], a general control framework is proposed and a distributed control system based on the framework is presented. The behavior of the complete robot is a collective behavior of all independent modules. All modules in the robot contain their own processing and actuation units, which allow them to evaluate and react to the environment independently. The modules can perform passive communication with their immediate neighbors and can exhibit aggressive or tolerant behaviors based on the environment changes to generate emergent group behaviors. Similarly, a multi-agent approach is proposed in [46] for grasping tasks. Control of the system is distributed among five different types of agents: link agents, joint agents, end-effector agents, task agents,

and object agents. As each agent attempts to achieve a desired individual behavior, the manipulator itself exhibits an emergent behavior that avoids obstacles while approaching the object to be grasped. Another similar architecture for the control of robot systems is discussed in [69]. By virtue of the support of adequate tools, this schema lends itself both to a general increase of robot programming capability and flexibility and to rapid prototyping of different architectural solutions. All the three frameworks focus on the implementation of agent-based approaches to solve the control problem of a single robot. It does not deal with the problem of controlling a complex system that is inherently distributed.

The Combination of Deliberative and Reactive Control

Exploration of highly risky terrains such as cliffs and risky construction sites by autonomous robotic systems requires a control architecture that is able to autonomously adapt to uncertainties in the environment. A software/hardware framework for cooperating multiple robots is proposed in [30] to facilitate the development of such coordinated tasks. This work builds upon earlier research into autonomous planetary rovers and robot arms. A distributed control architecture is presented in which integrated multi-robot mobility and control mechanisms are derived as group compositions and coordination of basic behaviors. This framework includes the necessary group behaviors and communication mechanisms for coordinated/cooperative control of heterogeneous robotic platforms. However, it doesn't provide a mechanism for analyzing the dynamics of the platforms. The stability of the MAS is not considered either.

Song proposes a framework for controlling and coordinating a group of robots for cooperative manipulation tasks in [61]. This framework enables a decentralized approach to planning and control that allows the robots approach the object, organize themselves into a formation that will trap the object and transport the object to the desired destination. But this framework focuses on the formation control of the MAS. It is not applicable for other applications.

A multi-agent approach for developing flexible real-time control systems for autonomous mobile robots is presented in [37]. It is studied how to integrate heterogeneous algorithms and functionalities on-board a robot, while still guaranteeing a reasonable response time. A framework is provided for developing complex intelligent machines. The proposed control

system is applied to robot navigation problems for which a robot runs through complex environments by combining visual tracking, obstacle avoidance, and command receiving behaviors within a single system. This architecture uses the agent approach to develop the control strategy for one single robot. It does not deal with control problems for multiple entities.

A new architecture for an agile shop floor control system is presented in [11]. The shop floor control system is a dynamic system with the capability of adapting and accepting unpredictable changes in its structures. The proposed architecture in [11] is based on the methodology of multi-agent systems and Distributed Artificial Intelligence. Characteristics of MASs are explored to implement a distributed, cooperative architecture for a shop floor control system. To implement the framework, a coordination model between agents and behavioral models of some representative agents are also established. This work focuses on using the agent-based approach to solve the shop floor control problem. It does not provide a strategy for the control of multi-agent problem.

1.1.4 Hybrid Systems and Multi-Agent Systems

Complex natural and engineering systems typically possess a hierarchical structure, characterized by continuous variable dynamics at the lowest level and logical decision making variables at the highest [7]. Virtually all control systems today issue continuous variable controls and perform logical checks that determine the modes and the control algorithms. The continuous variable system is operating at any given moment. Almost all control systems contain continuous-variable control commands and discrete logical commands. The interaction of the discrete dynamics and continuous dynamics lead to the challenging hybrid control problems. Hybrid systems involve both continuous variables and discrete variables. The evolution of continuous variables and discrete variables is given by dynamic equations that generally depend on both. These dynamic equations often contain a mixture of logic and discrete variables along with continuous variables. The continuous dynamics of hybrid systems may evolve according to continuous time or discrete time. Generally the continuous dynamics is given by differential equations. The discrete dynamics of hybrid systems is generally governed by a digital automaton or an input-output transition mapping with a limited number of states. The continuous and discrete dynamics interact

at certain events or certain firing times when the continuous state hits certain prescribed values in the continuous state space. Because of the hybrid nature of MASs, we have to consider simulating the continuous dynamics and the discrete dynamics of the agents. The evolution of continuous variables and discrete variables of an agent interacts with discrete events and continuous dynamics of another agent, and determines the evolution of the discrete variables and continuous variables of the whole MAS.

In [14], a hybrid systems framework for the real-time multi-agent coordination and control of multiple vehicles in the context of the multiple autonomous underwater vehicles are discussed. The authors address the use of hybrid systems techniques for analyzing and synthesizing the control architecture and describe how it can be implemented using an object-oriented framework for implementing real-time, event-driven, distributed multi-agent control systems. The work considers the hybrid nature of a MAS, it presents an architecture for the control of a MAS. However, it is domain dependent, and can not be regarded as a comprehensive framework.

Abstraction is a natural way to hierarchically decompose the analysis and design of hybrid systems. Given a hybrid control system and some desired properties, a system can be extracted while the properties of interest being preserved. In [63], a framework for abstraction that applies to discrete, continuous, and hybrid systems, is presented. A composition operator is introduced that allows the development of complex hybrid systems from simpler ones. The compatibility between abstractions and this compositional operator is also shown. This work presents an effective way for abstracting a hybrid agent, while it does not solve the control problem of MASs.

In [22], a generic framework for integrated modeling, control and coordination of multiple multi-mode dynamical systems is developed. This framework of distributed control of MASs is called Hybrid Intelligent Control Agent (HICA). In this framework, a certain form of knowledge-based deliberative planning is integrated with a set of verified hybrid control primitives along with coordination logic to provide coordinated control of systems of agents. This work gives the basis for analyzing MASs as hybrid control systems. Although in this framework, coordination factors have been defined as input coordination factors and output coordination factors, there is no generic cooperation and coordination mechanism defined for the HICA agents. Furthermore, because this framework is based

on the multiple unmanned ground vehicle/unmanned air vehicle pursuit-evasion problem, not all essential primitives are defined. It is not shown how to optimize the performance of a MAS either.

1.1.5 Objectives

While feasible in many applications, most of the above architectures lack some degree of generality and remain problem dependent. A generic framework for the control of MASs is desired. A generic framework should be applicable for both homogeneous and heterogeneous MASs. It should include hybrid primitives. Both time-driven and event-driven dynamics should be included in the framework. A generic framework can be applied to a wide range of problems, especially intelligent control of hybrid systems. In this study, the distributed intelligent control of multiple hybrid systems is discussed, and a framework for the cooperative and coordinated control of MASs is proposed. In order to make the framework generic, we need to include both deliberative control and reactive control in the agent. The results obtained will be valuable for the control of various MASs. The analysis and design approach proposed in this study for MASs will provide insights to researchers in intelligent control systems. The agents defined in this study are autonomous problem-solving entities that have the following properties:

- **Situated:** They can receive inputs related to the state of their environment through sensors.
- **Reasoning:** They are able to reason about the environment and have particular objectives to achieve. They have intelligence built in them.
- **Reactive:** They are able to respond to the changes of the world and make changes to the environment through actuators.
- **Interactive:** They are able to interact with other agents through explicit or implied communication channels in order to achieve the global goal.
- **Performance Measure:** They have a performance measure with which the agent's performance on the tasks can be evaluated.

1.2 Contribution

The major contribution of this work is that a generic Coordinated Hybrid Agent (CHA) framework is proposed to tackle the control problem for MASs. This generic framework is able to model the control of multiple hybrid agent systems. The proposed framework can be applied to the design and analysis of both homogeneous and heterogeneous MASs. An example of the homogeneous system is the multi-crane system mentioned above. An example of the heterogeneous MAS consists of an overhead crane, a mobile robot, and a robot manipulator, which has also been introduced. The CHA framework is proposed as a novel approach for the integrated modeling, cooperative and coordinated control of MASs that consist of multiple hybrid systems. In the proposed framework, the control of MASs is considered as achieving decentralized control and coordination of agents. Each agent is modeled as a CHA, which is composed of an intelligent coordination control layer and a hybrid control layer. The intelligent coordination control layer deals with the planning, coordination, decision-making and computation of the agent. The hybrid control layer of the proposed framework takes the output of the intelligent coordination layer and generates discrete and continuous control signals to control the overall process. The proposed framework includes both the hybrid dynamics, controls, and discrete logic for coordination and cooperation for the system. Different from other researchers' work, in the proposed framework, we use the theory of coordination states, and include a coordination rule base, an intelligent planner and a direct communication module in the intelligent coordination control layer which make the proposed framework generic for various problems. With the introduction of the intelligent coordination control layer and the hybrid control layer, we are able to analyze the local stability of the agent, and then to analyze the global stability of the MAS as logical DESs.

With the development of the framework for the control of MASs where agents cooperate, coordinate and interact with each other, the contribution of this work includes:

- A generic framework for the control of MASs is proposed. The issues that usually arise in real-world hybrid systems are identified. The mathematical model of hybrid systems as interacting collections of dynamical systems is introduced in this work. MASs discussed in this work evolve in continuous-variable state spaces and subject

to continuous controls and discrete transitions. The hybrid model proposed in the framework generalizes the concept of hybrid systems, and retains enough information on which to pose and solve meaningful control problems.

- The proposed framework models both strategic control and tactic control. The integrated framework consistently and transparently deals with both high level design requirements and low-level design requirements.

- The proposed framework also provides the integration of deliberative control with hybrid reactive control through the interaction between the intelligent coordination control layer and the hybrid control layer.

- The proposed framework has been applied to a few scenarios illustrating homogeneous and heterogeneous configurations. The feasibility of the proposed generic framework for the control of MASs is demonstrated using experiments and/or numerical simulations.

- The stability of MASs modeled by the CHA framework is defined. Because each CHA can be considered as a Discrete Event System (DES), we define the stability of CHA systems in the sense of Lyapunov as a logical DES. The necessary and sufficient conditions for the stability of CHA MASs are presented. In addition, the asymptotic stability and exponential stability of a MAS modeled using the proposed framework is also studied. It is also proved that both the heterogeneous and homogeneous MASs developed using the proposed framework are exponentially stable.

- The optimization of MASs modeled by the CHA framework has also been studied. The optimization of a CHA system can be described using time-driven dynamics and event-driven dynamics. In particular, we have demonstrated the existence of optimal controls. A direct identification algorithm is applied to optimize the performance and timing of the MAS. Simulation results are given to check on the efficiency of the optimization algorithm.

- In a MAS, it is desirable to set the order of tasks in multiple concurrent hybrid systems. We define this as scheduling for a CHA framework. A scheduling scheme

for a CHA system is proposed that provides two features. First, an algorithm for ordering the use of shared resources among agents. Second, a means of predicting the worst-case behavior of the MAS when the scheduling algorithm for ordering is applied.

- In order to coordinate the agents during planning, we introduce the concept of coordination rule base in the intelligent coordination control layer of a CHA. The coordination rules can be defined for the MAS in a rule base called coordination rule base. Planning of the agents have to follow the rules defined in the coordination rule base. The coordination rules can be considered as desirable choices and constraints for the actions of agents. The constraints specify which of the actions are in fact not allowed in a given state. The desirable choices in general are desirable actions that are available for a given state.

We have proposed a generic framework that includes the notion of hybrid systems. The proposed framework is applicable for the control of MASs that relates to differential equations and automata. A hybrid controller is included in the framework that issues continuous-variable commands and makes logical decisions. We have developed a theory for synthesizing hybrid controllers for MASs in order to optimize the control strategy. The major contribution of this work is illustrated throughout this dissertation by various examples, simulations and experiments. Since the proposed framework is to be applied to various applications, the overhead might be unnecessary for certain applications.

1.3 Related Research Areas

To accomplish the objectives of this study, areas of related research are identified as distributed systems, MASs, discrete event systems, artificial intelligence, computer vision, mobile robotics, robot manipulators, hybrid control systems, and so on.

1.4 Organization

Chapter 2 gives some background information about this research work, including MASs, centralized and decentralized control, input/output automata, continuous and discrete event systems, hybrid systems and hybrid intelligent control agent. Chapter 3 describes the proposed framework for the control of multi-agent systems. Chapter 4 gives the stability analysis of the proposed framework. The optimization of MASs with the proposed framework is analyzed in Chapter 5. The direct identification algorithm is also introduced for solving the optimal control problem for a CHA MAS. Chapter 6 describes the application of the proposed framework to some scenarios to illustrate the feasibility of the proposed framework. By using the proposed framework, the control schemes are developed for these MASs. It is demonstrated that the proposed framework is generic and is applicable to both homogeneous and heterogeneous MASs. Simulation and experimental results are also given. Chapter 7 gives some examples to analyze the stability and optimization of MASs using the methodology we have proposed. In Chapter 8, we summarize this work and give some directions for future research.

Chapter 2

Background

In this chapter, we introduce the background knowledge of various areas related to this research.

2.1 Multi-Agent Systems

The two most important fields of multi-agent systems are DAI and Artificial Life (AL) [19]. The purpose of DAI is to create systems that are capable of solving problems by reasoning based on dealing with symbols. The purpose of AL is to build systems that are capable of surviving and adapting to the environments. The research into agents was originated in 1977 [28] by Hewitt. He proposed the actor model of computation to organize programs in which the intelligence is modeled using a society of communicating knowledge-based problem-solving experts. Since then, the research in agents has continued and evolved. The research of sharing data among agents dates back to 1980 [17]. In this work, the model of the blackboard system was developed. Objects in the working area were inserted, modified and withdrawn in a common area called the blackboard.

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators. An agent within a multi-agent system can be thought of as a system that tries to fulfill a set of goals within the complex, dynamic environment. Agents have only a partial representation of the environment. In recent years, there has been a growing interest in the control of systems that are composed

of several interacting autonomous agents.

2.2 Centralized Control and Decentralized Control

The centralized control paradigm is characterized by a complex central processing unit that is designed to solve the whole problem. The central unit must gather the data from the whole system. The solution algorithms are necessarily complex and problem specific. Thus the processing unit is able to check that a solution is the globally optimal solution which is not easily achieved in a decentralized control paradigm. However, utilizing complex algorithms and analyzing all information in a centralized controller always cause slower responses than a decentralized control system.

Decentralized control paradigms are based on distributed control in which individual components react to local conditions simultaneously. These individual components interact with neighboring components to exhibit desired adaptive behaviors. The complex behaviors are a resultant property of the system of connections. The decentralized nature of information in many large-scale systems, requires the control systems to be decentralized. Decentralized control of discrete-event systems, in the absence of communication, has been well studied. Control of logical discrete-event systems with communication is investigated in [4]. Controllers observe events generated by the system and are allowed to pass messages in order to attempt to resolve ambiguities and to determine correct control actions.

A problem is called intractable if the time required to solve instances of the problem grows exponentially with the size of the instances. Exponential growth means that even moderately large instances cannot be solved in any reasonable time. One should try to divide the overall problem of generating intelligent behavior into tractable subproblems rather than intractable ones [54].

2.3 Input/Output Automata

The input/output automaton model is defined in [40] as a tool for modeling concurrent and distributed discrete event systems. I/O automata provides an appropriate way to model discrete event systems consisting of concurrently-operating components. A funda-

mental property of the model is that there is a clear distinction between the actions whose performance is under control of the autamaton and the actions whose performance is under the control of the environment. An automaton generates output and internal actions, and transmits output to the environment. The automaton's input is generated by the environment and transmitted to the automaton. The I/O automata model allows precise statements of the problems of concurrent systems. The problems are formulated as sets of finite and infinite sequences of external actions. The I/O automata model is general enough to serve as an operational model for many different languages describing sets of action sequences.

An I/O automaton A has five major components:

1. An action signature $sig(A)$;
2. A set $states(A)$ of states;
3. A nonempty set $start(A) \subseteq states(A)$ of start states;
4. A transition relation $steps(A) \subseteq states(A) \times acts(A) \times states(A)$;
5. An equivalence relation $part(A)$ partitioning the set $local(A)$ into at most a countable number of equivalence classes.

2.4 Continuous Systems and Discrete Event Systems

"A variable is called discrete if it takes values in a countable set and it is called continuous otherwise" [39].

Discrete Event Systems (DESs) are dynamical systems which evolve in time by occurrence of events at time intervals not necessarily regular. Some examples include flexible manufacturing systems, computer networks, logic circuits, and traffic systems [48]. "Logical" DESs are a class of discrete time DES with equations of motion that are most often nonlinear and discontinuous in the occurrence of the events. It has been long known that a stability theory can be developed in a very broad setting which is phrased in terms of motions of dynamical systems and which does not require the description of the system under investigation in terms of specific equations. In [49], Passino introduces a logical DES model and defines stability in the sense of Lyapunov and asymptotic stability for logical DESs. He shows that the metric space formulation can be used for the analysis of stability

for logical DESs by employing appropriate Lyapunov functions. Consider stability properties of discrete event systems that can be modeled accurately with $G = (X, \xi, f_e, g, E_a)$, where X is the set of states, ξ is the set of events, $f_e : X \to X$, g is the enable function, and E_a is the allowed event trajectories. The r-neighborhood of a set X is denoted by the set $S(X_m; r) = \{x \in X : 0 < \rho(x, X_m) < r\}$, where $r > 0$ and ρ denotes a metric on X. The necessary and sufficient conditions for Lyapunov stability of the DES are given as:

"For a closed invariant set $X_m \subset X$ of G to be stable in the sense of Lyapunov w.r.t. E_a, it is necessary and sufficient that in a sufficiently small neighborhood $S(X_m; r)$ of the set X_m there exists a specified functional V, V is called the Lyapunov function, with the following properties:

i) For all sufficiently small $c_1 > 0$, it is possible to find a $c_2 > 0$ such that $V(x) > c_2$ for $x \in S(X_m; r)$ and $\rho(x, X_m) > c_1$.

ii) For any $c_4 > 0$ as small as desired, it is possible to find a $c_3 > 0$ so small that when $\rho(x, X_m) < c_3$ for $x \in S(X_m; r)$ we have $V(x) \leq c_4$.

iii) $V(X(x_0, E_k, k))$ is a non-increasing function for k, for $x_0 \in S(X_m; r)$, for all integer k, as long as $X(x_0, E_k, k) \in S(X_m; r)$ for all E_k such that $E_k E \in E_a(x_0)$."

Modern systems involve both discrete and continuous states. Systems of interest in this study are typically governed by continuous dynamic equations at particular discrete states. Systems like these are considered as hybrid systems. In order to study the multi-agent systems consisting of hybrid systems, we need to include the hybrid system concept to model the controlled processes that have both discrete and continuous variables. Hence it is necessary to define the proposed framework to deal with both the discrete and continuous states.

2.5 Hybrid Systems

Hybrid systems are characterized by the combination of time-driven and event-driven dynamics. Time-driven dynamics are represented by differential equations, while the event-driven dynamics are described through various frameworks used for modeling of discrete event systems, such as timed automata, or Petri nets [51]. It is motivated by the structure of robotics and manufacturing systems. In these systems, discrete entities move through a

network of processing units which process the jobs so as to change their physical characteristics according to certain specifications.

Hybrid control systems are control systems that involve both continuous and discrete dynamics and continuous and discrete controls [7]. The continuous dynamics are usually modeled by ordinary differential equations and depend on some discrete phenomena, corresponding to discrete states, dynamics and controls. Examples of such systems include computer disk drives, transmissions and stepper motors, programmable logic controllers, constrained robotic systems, and automated highway systems. Such systems arise whenever logical decision-making is mixed with continuous control laws. In order to deal with hybrid control systems, there are two paradigms. The first one is aggregation, and the second one is continuation. The aggregation paradigm treats the entire system as a finite automaton or discrete-event dynamic system. The continuation paradigm treats the whole system as a differential equation.

2.6 Hybrid Intelligent Control Agent

Many control problems involve processes that are inherently distributed, complex or that operate in multiple modes. Agent-based control is an emerging paradigm within the subdiscipline of distributed intelligent control. In [22], Fregene proposes the Hybrid Intelligent Control Agent (HICA) as a conceptual basis for the synthesis of intelligent controllers in problem domains which are inherently distributed.

The main advantage of HICA is that the number of agents that have to be developed, and for which coordination protocols need to be provided, is significantly reduced. HICA combines deliberative planning/coordination with reactivity. The main elements of HICA are Y, Σ, U, C, S, S_q, Obj and $Status$, where Y represents the output from the controlled process; Σ is a discrete control output generated by the control module; U is the continuous control signal; C is a set of coordination factors; S represents supervisory commands; S_q is the sequence of control primitives to achieve the objective; Obj is the short-term objectives to achieve the overall goal of the agent; and $Status$ represents a direct feedback of hybrid states. In the hybrid control system, the controller is represented by

$$\begin{aligned} \eta_c &: Obj \times Sq \times U_c \times \Sigma_c \rightarrow \{0,1\} \\ h_c &: U_c \rightarrow 2^{Y_c} \end{aligned} \tag{2.1}$$

in which η_c is the discrete transition function for the control module; h_c is the controller output function.

The plant is represented by

$$\begin{aligned} f_p &: Q \times X \times U \times D \rightarrow X \\ \eta_p &: Q \times X \times \Sigma \rightarrow 2^{Q \times X} \\ h_p &: Q \times X \rightarrow 2^{Y} \end{aligned} \tag{2.2}$$

where f_p is a vector field that describes the evolution of the continuous state X within the mode Q; D is the disturbance; η_p is the discrete transition function for the plant and h_p is the plant output function.

The key idea of HICA is to combine concepts from hybrid control and multi-agent systems to build agents which are especially suitable for multi-mode control purposes. HICA conceptually wraps an intelligent agent around a core that is itself a hybrid control system. Fregene [22] illustrates how HICA might be used as a control agent to synthesize agent-based controllers for inherently distributed multi-mode problems.

2.7 Summary

This chapter gives background knowledge related to the thesis. It includes multi-agent systems and control of multi-agent systems, centralized and decentralized control, continuous and discrete event systems, hybrid intelligent control agent, input/output automata, and so on. The proposed framework will be introduced in the next chapter.

Chapter 3

The Proposed Framework for the Control of Multi-Agent Systems

The control of multi-agent systems is an emerging paradigm within the sub-discipline of distributed intelligent control. In this chapter, a framework is proposed for the distributed control and coordination of multi-agent systems. In the proposed framework, the control of multi-agent systems focuses on decentralized control and coordination of agents. Each agent is modeled as a CHA which is composed of an intelligent coordination control layer and a hybrid control layer as shown in Figure 3.1. The core of the proposed framework is on developing coordinated agents for the control of hybrid multi-agent systems. A generic control architecture is developed to control either a homogeneous multi-agent system or a heterogeneous multi-agent system. The proposed framework is able to model the cooperation, coordination and communication among the members of the multi-agent system. The control scheme is able to control a multi-agent system where agents cooperate, coordinate and interact with each other.

3.1 The Agent Workspace

Agents can either work within the same workspace or have their own workspace. In order to execute a common task, two or more agents might need to cooperate and coordinate within the same workspace. For other tasks, agents may need to work in their own workspace and

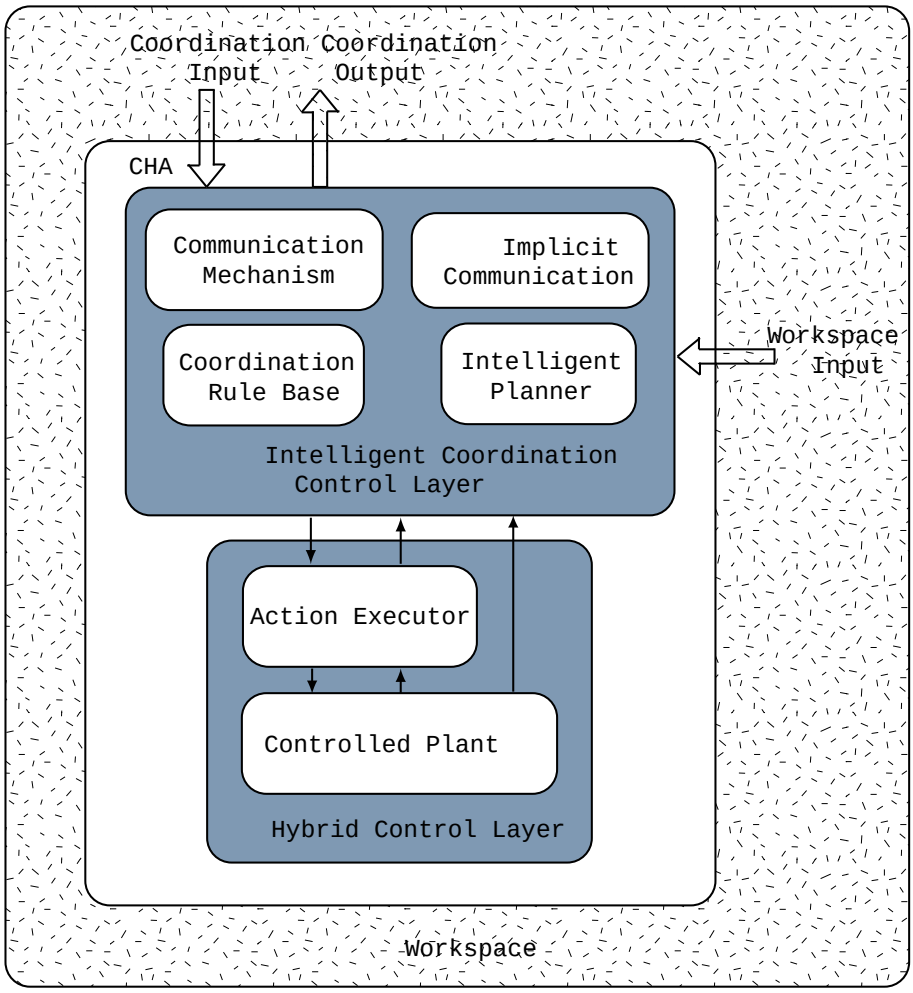

Figure 3.1: The Internal Structure of a CHA Agent.

communicate with each other to achieve a global goal.

In a workspace $s_i \in S$ for an agent or for a group of agents AG, we have the following variables:

- agent(s) working in the workspace, represented by AG;
- one goal or a group of goals GL;
- obstacles and constraints O;
- objects J;
- boundaries B for the workspace.

These variables are called entities of a workspace. Entities of a workspace can trigger events for the agents to react.

3.2 The Hybrid Control Layer

We introduce the hybrid control layer, which is composed of trajectories of the system, the controlled process, the action executor and the execution of hybrid actions.

3.2.1 Trajectories of the System

Let T denote the time axis. Since a hybrid system evolves in continuous time, we assume an interval V of $T \subseteq \mathcal{R}$ to be $V = [t_i, t_f] = \{t \in T | t_i \leq t \leq t_f, i \in \mathcal{Z}^+, f \in \mathcal{Z}^+\}$. The variables of the system evolve either continuously or in instantaneous jumps. The addition of T is also allowed. For an interval V and $t_0 \in T$, we have $t_0 + V = \{t_0 + t' | t' \in V\}$. Using the concepts from [39], we have the following definitions.

Definition 3.2.1 *If we denote the discrete evolution space of a hybrid system as Q and the continuous evolution space of a hybrid system as X, a* trajectory *of a hybrid system can be defined as a mapping $V \rightarrow Q \times X$.*

The evolution of the continuous state in each sub-interval of V is described as $f : Q \times X \times U \rightarrow Q \times TX$, where U represents the continuous control signal space, TX represents the tangent space of space X. Thus for every sub-interval of V, we have $\dot{x}(t) = f(q(t), x(t), u(t))$, in which f is the vector field. We assume the existence and uniqueness of solutions to the ordinary differential equation on f.

Definition 3.2.2 *The application of the continuous control signal $u \in U$ and the discrete control signal $m \in M$ is defined as a* hybrid action *which is denoted by $a \in A$. In each sub-interval, $q(t)$ is a constant.*

The discrete jumps of the state occur at t_{i+1}, t_{i+2}, ..., t_{f-1} while the value of q and x change simultaneously. The state of the system takes a discrete jump at time t from $(q(t), x(t)) \in Q \times X$ to $(q(t'), x(t')) \in Q \times X$ when a discrete control signal m of an action a is taken (controlled jumps), or when certain criteria of the system are met (autonomous jumps).

Definition 3.2.3 *We define $\diamond$ as the* restriction *of trajectory E to a subset of its domain $d(E)$ in which discrete state transitions occur only at the starting point and/or at the ending point. There is no discrete transition at the starting point or the ending point if the interval is left-open or right-open, respectively. $E \diamond [t_1, t_2]$ means the subset of trajectory E over $t_1 \leq t \leq t_2$. It can also be denoted as $E \diamond V$, which means the subset of trajectory E over $[t_i, t_f]$.*

Definition 3.2.4 *If E_1 is a trajectory with a right-closed domain $V_1 = [t_i, t_j]$, E_2 is a trajectory with domain $V_2 = [t_j, t_f]$, we define the* trajectory link *of E_1 and E_2 to be the trajectory over $[t_i, t_f]$ as*

$$E_1 \propto E_2(t) = \begin{cases} E_1(t) & \text{if } t \in V_1; \\ E_2(t) & \text{otherwise.} \end{cases}$$

For a countable sequence of trajectories, if E_i is a trajectory with domain V_i, while all V_i are right-closed, and if $1 \leq i \leq \infty$ and $i \in \mathcal{Z}$, the infinite trajectory link can be written as $E_1 \propto E_2 \propto E_3 \ldots$ over $V_1 \cup V_2 \cup V_3 \ldots$.

3.2.2 The Controlled Process in the Proposed Framework

The controlled process for each agent is essentially a hybrid system whose dynamics are controlled by the coordinated hybrid agent. The evolution of the controlled process is given by

$$I_p \subset Q_p \times X_p \tag{3.1}$$
$$Y_p \subset Q_a \times X_a \tag{3.2}$$
$$E_p = E_{p_1} \propto E_{p_2} \propto \ldots \propto E_{p_k} \tag{3.3}$$
$$\eta_p : Q_p \times X_p \times M \rightarrow \mathcal{P}(Q_p \times X_p) \tag{3.4}$$
$$\gamma_p : Q_p \times X_p \rightarrow \mathcal{P}(Q_p \times X_p) \tag{3.5}$$
$$f_p : Q_p \times X_p \times U \rightarrow TX_p \tag{3.6}$$
$$h_p : Q_p \times X_p \rightarrow Y_p \tag{3.7}$$

- I_p is the initial state of the controlled process that gives both the initial discrete state Q_p and the initial continuous state X_p.

- Y_p is the output space of the controlled process which is a subset of the space $Q_a \times X_a$, where Q_a is the discrete state of the hybrid system read by the sensors, X_a is the continuous state of the hybrid system read by the sensors.

- $E_p = E_{p_1} \propto E_{p_2} \propto \ldots \propto E_{p_k}$ is the trajectory of the controlled process. It has k discrete states in consequence and $E_{p_i} = E_p \diamond V_i$, in which $V_i = [t_{i1}, t_{i2}]$, $i \in \mathcal{Z}^+$, t_{i1} represent the starting point of the sub-interval and t_{i2} represent the ending point of the sub-interval, respectively. E_p is determined by the discrete state evolution and the continuous state evolution of the controlled process.

- η_p is a function that governs the controlled discrete transition of the controlled process. $\mathcal{P}(.)$ represents the power set. $\forall V = [t_i, t_f]$, the controlled discrete jumps of the controlled process is given by

$$q_p(t') = \eta_p(q_p(t), x_p(t), m) \tag{3.8}$$

  where $q_p \in Q_p$, $x_p \in X_p$ and $m \in M$ represents the discrete control signal.

- γ_p is the function that governs the autonomous discrete transition of the process. As mentioned before, there are both controlled jumps and autonomous jumps for the hybrid system. $\forall V = [t_i, t_f]$, the autonomous discrete jumps of the controlled process is given by

$$q_p(t') = \gamma_p(q_p(t), x_p(t)) \tag{3.9}$$

  where $q_p \in Q_p$ and $x_p \in X_p$.

- f_p is the vector field determined by the evolution of the continuous state ($x_p \in X_p$) of the controlled process at a certain discrete state ($q_p \in Q_p$) of the controlled process (i.e., within the sub-interval of V while the discrete state $q_p(t)$ is a constant or a set of constants).

  Thus, $\exists c$, if $q_p(t_i) = c$ over interval V_i, $1 \leq i \leq \infty$, $i \in \mathcal{Z}$, $V_i = [t_{i1}, t_{i2}]$, then the restriction $(E_p \diamond V_i)$ of the trajectory of the controlled process E_p is given by

$$\dot{x_p} = f_p(q_p(t_i), x_p(t_i), u(t_i)) \tag{3.10}$$

  where $q_p \in Q_p$, $x_p \in X_p$ and $u \in U$ represents the continuous control signal.

- The output $y_p(t) \in Y_p$ is the feedback of the controlled process. The output is read by the sensors and is given by

$$y_p(t) = h_p(q_p(t), x_p(t)). \tag{3.11}$$

3.2.3 The Action Executor

For each single agent, the evolution of the discrete and continuous state of the system is considered as the execution of a hybrid action.

The action executor has two functions f_e and η_e defined as:

$$f_e \;:\; A \times Y_p \times X_r \rightarrow U \tag{3.12}$$

$$\eta_e \;:\; A \times Y_p \rightarrow M \tag{3.13}$$

- f_e is the continuous action execution function that takes the desired hybrid action $a \in A$, the output $y_p \in Y_p$ of the process, and the reference value $x_r \in X_r$ as input, then generates the continuous control signal $u \in U$ for the process.

- η_e is the discrete action execution function that takes the desired hybrid action $a \in A$ and the output $y_p \in Y_p$ of the process as input, then generates the discrete control output to the process.

The selection of appropriate actions and sequence of the actions are handled by the intelligent coordination control layer which will be introduced later. Because the action executor deals with all the local control problems, in the view of the intelligent coordination control layer, the controlled process can be considered as a discrete event system.

3.2.4 The Execution of Hybrid Actions

Similar to [41], we describe the execution of the hybrid actions as a finite or infinite alternating sequence.

Definition 3.2.5 *An* execution sequence *is defined as* $\beta = E_{p_1} a_1 E_{p_2} a_2 E_{p_3} a_3 \ldots$, *where* E_{p_i} *is the restriction* $E_p \diamond V_i$ *and* a_i *is the hybrid action that occurs between* E_{p_i} *and* $E_{p_{i+1}}$.

Note that there will always be a hybrid action between E_{p_i} and $E_{p_{i+1}}$ no matter the discrete jump is a controlled jump or an autonomous jump. This is because if there is an autonomous jump between E_{p_i} and $E_{p_{i+1}}$, a null action can be used to represent that no action is taken.

A finite execution sequence ends with a restriction. If E_{p_i} is not the last restriction in β, then after the execution of a_i, we have a new trajectory link $E_{p_i} \propto E_{p_{i+1}}$.

The execution sequence β of the hybrid actions determines the trajectory E_p. E_p represents the evolution of the discrete states of the hybrid system, and the evolution of the continuous states in between the discrete transitions.

3.3 The Intelligent Coordination Control Layer

In general, problems involving multi-agent coordination can be modeled by assuming that states represent the joint state of n agents. Also, the joint action represents actions of all the agents, where, each agent may not have knowledge of other agents' actions. In most practical problems, the joint state and action sets are exponential in the number of

agents, and the aim is to find a distributed solution that does not require combinatorial enumeration over joint states and actions.

The key of the proposed approach is to build an intelligent coordination control layer above the hybrid control layer for the intelligent agent. Thus the hybrid dynamical system is hidden under the intelligent coordination control layer. In the CHA framework, local hybrid dynamics are considered as hybrid actions. The intelligent coordination control layer has full authority control and coordination of the agent in an abstract way. The intelligent coordination control layer plans the sequence of control primitives and selects appropriate hybrid actions without violating the coordination rules. The intelligent coordination control layer is built upon the action executor. It also communicates with the supervisor and neighboring agents whenever necessary to enhance the cooperation and coordination.

3.3.1 Coordination States

Definition 3.3.1 *At the intelligent coordination control layer, we define the states of the agent in an abstract way, which we call* coordination states *of the CHA. We denote the set of coordination states as R.*

Although the coordination states are also discrete states, they are different from the discrete states Q defined for the controlled process. The coordination states represent how much an agent has completed a series of hybrid actions in order to complete a desired task. The evolution of the coordination state $r \in R$ is determined by the intelligent coordination control layer. The evolution of the coordination state along a planned trajectory is accomplished by the action executor.

3.3.2 The Model of the Intelligent Coordination Control Layer

Agents repeatedly and simultaneously take actions, which lead them from their previous states to new states. As illustrated in Figure 3.1, in the proposed framework, an intelligent coordination control layer is built above the hybrid control layer. The hybrid dynamic system is hidden under the intelligent coordination control layer. The intelligent coordination control layer interacts with other agents through the *communication mechanism*. In addition, the intelligent coordination control layer takes Q_a and X_a as feedback from the

controlled plant, then it outputs the desired action $a \in A$ and reference value $x_r \in X_r$ to the action executor. The intelligent coordination control layer is modeled as

$$
\begin{aligned}
I &\subset R && (3.14)\\
\varphi &: Q_a \times X_a \to R && (3.15)\\
\beta &: Q^{\triangleleft} \times X^{\triangleleft} \times S^{\triangleleft} \times Q_s \times X_s \times R \to A && (3.16)\\
f_c &: R \to R && (3.17)\\
g &: R \to \mathcal{P}(A) - \{\emptyset\} && (3.18)\\
f_r &: R \times X^{\triangleleft} \times X_s \times X_a \to X_r && (3.19)\\
f_o &: R \times X_s \times X_a \to X^{\triangleright} \times S^{\triangleright} && (3.20)\\
\phi_o &: R \times Q_s \times Q_a \to Q^{\triangleright} \times S^{\triangleright} && (3.21)
\end{aligned}
$$

where

- I is the initial state of the agent that gives the initial coordination state R.
- φ is the logic function that maps the feedback from the controlled plant Q_a and X_a to the coordination state set R.
- β is the function that maps the discrete coordination input $Q^{\triangleleft}$, the continuous coordination input $X^{\triangleleft}$, the coordination input signature $S^{\triangleleft}$, the discrete workspace state Q_s, the continuous workspace state X_s and the coordination state to the desired action set A.
- f_c is the function that governs the transition from the current coordination state to the next coordination state. It is defined by the coordination rule base and the intelligent planner that will be introduced later.
- g is the enabling function for $a \in A$. We only need f_c to be defined when $a \in A$ occurs, g maps R to a non-empty state set (i.e., there are always some actions that lead to the next state).

- f_r is the function that maps the current coordination state R, the continuous coordination input $X^{\triangleleft}$, the continuous workspace state X_s and the continuous controlled plant state X_a to the reference value X_r for the action executor.

- f_o is the function that maps the current coordination state R, the continuous workspace state X_s and the continuous controlled plant state X_a to the continuous coordination output $X^{\triangleright}$. The destination agent of the output is given by the coordination output signature $S^{\triangleright}$.

- ϕ_o is the function that maps the current coordination state R, the discrete workspace state Q_s and the discrete controlled plant state Q_a to the discrete coordination output $Q^{\triangleright}$. The destination agent of the output is given by the coordination output signature $S^{\triangleright}$.

3.3.3 Coordination Rule Base

In order to coordinate the agents while planning, we introduce the concept of coordination rule base which is inspired by social laws defined in [56]. The coordination rules can be considered as desirable choices and constraints for the actions of agents. The constraints specify which of the actions are in fact not allowed in a given state. The desirable choices in general are desirable actions that are available for a given state.

Definition 3.3.2 *Given a set of coordination states R, a set of rules L, and a set of actions A, a* desirable choice *is a pair (a, l_o) where $a \in A$ and $l_o \in L$ is a rule that defines a desirable action that results in a transition with the maximum distance along the path of R in the metric space at the given coordination state $r \in R$.*

Definition 3.3.3 *Given a set of coordination states R, a set of rules L, and a set of actions A, a* constraint *is a pair (a, l_c) where $a \in A$ and $l_c \in L$ is a rule that defines a constraint at the given coordination state $r \in R$.*

Definition 3.3.4 *A* coordination rule set *is a set of desirable choices (a, l_{oi}) and constraints (a_i, l_{ci}). We denote the coordination rule set as C. The coordination rule set defines which action should be taken at a given coordination state $r \in R$.*

A set of rules L is used to describe what is true and false in different coordination states of the agent. Given a coordination state $r \in R$ and a rule $l \in L$, r might satisfy or not satisfy l. We denote the fact that r satisfies l by $r \models l$. The meaning of (a_i, l_i) will be that l_i is the most general condition about coordination states which chooses or prohibits the action a_i.

Definition 3.3.5 *A* coordination rule base *for the intelligent coordination control layer of a CHA is a tuple (R, L, A, C, T) in which C is a coordination rule set, and T is the transition function $T : R \times A \times L \to \mathcal{P}(R)$ such that: For every $r \in R$, $a \in A$, $c \in C$, if $r \models l_c$ holds and $(a, l_c) \in C$, then $T(r, a, l) = \emptyset$, the empty set, which means the desired transition is prohibited; For every $r \in R$, $a \in A$, $c \in C$, if $r \models l_o$ holds and $(a, l_o) \in C$, then $T(r, a, l) = \check{r}$, where $\check{r}$ is the coordination state after the desirable action is taken.*

The coordination rule base provides a skeleton for the agents to coordinate with others. Agents in a multi-agent system with a coordination rule base share the set of abstract states, the convention for describing states, the set of potential actions and the transition functions.

3.3.4 Intelligent Planner

Without violating the coordination rule base, the intelligent coordination control layer can have built-in intelligent planners to generate actions as the input to the action executor. Following a coordination state $r \in R$, the selected action is determined by $T : R \times A \times L \to \mathcal{P}(R)$. The AI approaches for planning tasks such as potential field methods, artificial neural networks, and knowledge based planning schemes can be implemented as possible intelligent planners.

The intelligent planner plans the desired coordination state trajectory that is checked against the coordination rule base to make sure that the trajectory is not violating the rules. After the desired trajectory has been planned, the action $a \in A$ for each step along the trajectory can be selected and output to the action executor.

For a given present state in R, denoted by r_p, the next state r_n is obtained by

$$r_n \Leftarrow \Big(x_{r_n} = \max\{x_i, i = 1, 2, \ldots, k\} \Big), \tag{3.22}$$

where x is the degree of fitness of the coordination state, i is the number of the neighboring coordination states including itself (i.e. all the possible next states).

3.3.5 Implicit Communication

In addition to the coordination rule base and the intelligent planner, in a CHA framework, agents interact to a module called *implicit communication* to coordinate their behaviors. This is necessary for applications in which agents need to cooperate and react at a high speed. Instead of using a network-based communication mechanism, agents interact with each other through sensors and actuators in order to cooperate and coordinate.

Implicit communication can be modeled as reactive agents working in the same workspace. However, since reactive agents don't have the ability to plan, the planning ability for agents is implemented through the intelligent coordination control layer.

3.4 The Capacity of a CHA System

In a CHA MAS, it is possible that multiple concurrent systems share the same resource. In order to analyze how many agents can share the limited resource, we need to solve the capacity problem of a CHA system. A scheduling scheme can be designed for a CHA system with multiple systems sharing the resources, which provides two features:

1. An algorithm for ordering the use of shared resources among agents. In particular, multiple agents depend on the output of one single agent.

2. A means of predicting the worst-case behavior of the system when the scheduling algorithm is applied.

A scheduling scheme can be static (if the predictions are undertaken before execution) or dynamic (if run-time decisions are used).

In this thesis, we will concentrate mainly on schemes. Agents are assigned priorities such that at all times the agent with the highest priority is occupying the shared resource (if it is not blocked by other agents). A scheduling scheme will therefore involve a priority assignment algorithm.

3.4.1 Simple Concurrent Model

It is necessary to impose some restrictions on the structure of concurrent CHA systems.

- The CHA system is assumed to consist of a fixed set of agents.
- All agents requiring the same resource are periodic, and the periods are known.
- The agents requiring the same resource are completely independent of each other.
- All agents have fixed worst-case execution times.

One consequence of the agents' independence is that it can be assumed that at some point in time all agents start executing. This represents the maximum capability of a CHA system.

3.4.2 Fixed-Priority Scheduling

Each agent has a fixed and static priority which is computed at pre-run-time. The shared resource is assigned in the order determined by agents' priority. The agent that takes the shortest time on the shared resource has the highest priority. The agent that takes the longest time on the shared resource has the lowest priority.

Assume C_i is the time the ith agent spends on the shared resource, and T_i is the time the ith agent can finish one cycle of its job, then we need

$$\sum_{i=1}^{N}(\frac{C_i}{T_i}) \leq 1. \tag{3.23}$$

This means that if the utilization of the agent set is less than the total capacity of the shared resource, all agents can have access to the shared resource. This is true because if the condition holds, the sum of the time all agents spend on the shared resource will be less than or equal to the period of the agent with the lowest priority, which is the worst case.

3.5 Summary

Multi-agent systems represent a group of agents operating cooperatively to solve common tasks in dynamic environments. In this chapter, a generic framework is proposed for the control of multi-agent systems. In the proposed framework, the control of multi-agent systems is considered as achieving decentralized control and coordination of distributed agents. Each agent is modeled as a CHA which is composed of an intelligent coordination control layer and a hybrid control layer. Different from other researchers' work, in the proposed framework, we use the theory of coordination states, and include a coordination rule base, an intelligent planner and an implicit communication module in the intelligent coordination control layer which makes the proposed framework generic for various problems. The proposed framework includes the primitives necessary for the modeling of MASs. With the introduction of the intelligent coordination control layer and the hybrid control layer, we are able to analyze the local stability of the agent, and then to analyze the global stability of the MAS as logical DESs.

Chapter 4

Stability Analysis of the Proposed Framework

In this chapter, we discuss the stability of MASs modeled using the CHA framework. We are interested in both the local stability and the global stability of MASs. The local stability is used to describe each single agent's ability to maintain the stability of one entity in a MAS. The global stability of a MAS describes the ability of a group of agents' ability to achieve a desired goal. In order to achieve the global stability, the local stability of all agents has to be guaranteed. First, we discuss the local stability of a single agent. For the local stability, we analyze the stability of a CHA which is modeled as a hybrid system with two layers (i.e., the hybrid control layer and the intelligent coordination control layer). Then, we discuss the global stability of a MAS. In order to analyze the global stability of a MAS, each CHA can be treated as a DES at the upper layer. The dynamics of the DES evolve in time with the occurrence of events at possibly irregular time intervals.

4.1 Local Stability of the Proposed Framework

The local stability of a CHA can be analyzed using the similar approach as discussed in [21]. In order to analyze the stability of an agent, we have to consider both discrete and continuous variables. An agent is said to be locally stable if the control of an agent has been designed such that the continuous state evolution at each abstract state is stable in

the sense of Lyapunov with respect to the equilibrium point; transition from one abstract state to another abstract state does not cause the transition from the other abstract state back to the abstract state immediately; and unsafe continuous states are avoided during transition. Based on these requirements, we can define the local stability of a CHA.

Definition 4.1.1 *An agent in a CHA MAS is said to be* locally stable *if*

1. *The action executor is designed such that the continuous state evolution is stable in the sense of Lyapunov with respect to the equilibrium point of that abstract state.*
2. *The abstract state transition is achievable through the intelligent coordination control layer, and the reverse state transition does not happen immediately.*
3. *All unsafe regions in the continuous space are avoided with the proper design of the action executor.*

With the definition of the local stability of an agent, we are able to analyze the global stability of a CHA MAS.

4.2 Global Stability of the Proposed Framework

Since agents modeled with the proposed framework interact with other agents through discrete events, in order to analyze the global stability of a MAS, each CHA can be treated as a DES at the intelligent coordination control layer. The dynamics of the DES evolve in time with the occurrence of events at possibly irregular time intervals. Manufacturing systems, computer networks, logic circuits, and robotic systems are good examples of DESs. Events in a manufacturing system could be the arrival of a part, the commencement of the processing of a job, or the finishing of the processing of a job. Events happen at random time. In a CHA MAS, we have multiple hybrid systems, which involve events happening asynchronously. We have discussed the local stability. Since we need to treat the agents as DESs to analyze the global stability, we will apply the stability analysis method proposed by Passino in [49] and [48].

According to the model of the intelligent coordination control layer introduced above, the stability properties of the CHA systems can be accurately modeled as:

$$G = (R,\ A,\ f_c,\ g,\ \mathcal{E}_v), \tag{4.1}$$

where R is the coordination states, A is the set of hybrid actions, $f_c : R \to R$ for $a \in A$ is the transition function. $g : R \to \mathcal{P}(A) - \{\emptyset\}$ is the enable function which is governed by the coordination rule base, and $\mathcal{E}_v$ is the set of valid event trajectories for the coordination states R. Note that the events we are discussing here are the hybrid actions that the agent might take. It is also possible that, at some states, no actions should be taken; this is represented by a null action.

Let $r_k \in R$ represent the kth coordination state of the CHA and $a_k \in A$ represent an enabled action for r_k (i.e. $a_k \in g(r_k)$). As described above, at state $r_k \in R$, action $a_k \in A$ is taken, the next coordination state r_{k+1} is given by the transition function f_c. Thus, $r_{k+1} = f_c(r_k)$. Each valid event trajectory $\mathcal{E}_v$ represents a physically possible event trajectory. If $r_k \in R$ and $r_k \in g(r_k)$, a_k can be taken if it lies on a valid event trajectory that leads the state to $r_{k+1} = f_c(r_k)$.

In the proposed framework, we model the intelligent coordination control layer as G. First, we model the system via R, A, f_c and g. Then, the possible trajectories $\mathcal{E}_v$ are given. The allowed event trajectories are denoted as $\mathcal{E}_a \subset \mathcal{E}_v$. In the proposed framework, $\mathcal{E}_a$ is governed by the coordination rule base. The allowed event trajectories that begin at state $r_0 \in R$ is denoted by $\mathcal{E}_a(r_0)$. If we use $\mathcal{E}_k = \mathcal{E}_0\mathcal{E}_1\mathcal{E}_2 \ldots \mathcal{E}_{k-1}$ to denote an event sequence of k events, the value of function $\mathcal{R}(r_0, \mathcal{E}_k, k)$ to denote the coordination state reached at time k from $r_0 \in R$ by the application of event sequence $\mathcal{E}_k$, then $\mathcal{R}(\mathcal{R}(r_0, \mathcal{E}_k, k), \mathcal{E}_{k'}, k') = \mathcal{E}(r_0, \mathcal{E}_k\mathcal{E}_{k'}, k + k')$. In order to guarantee the global stability of the CHA system, we need to define the coordination rule base properly to guarantee desired action sequences that will make the system stable.

Definition 4.2.1 *For a CHA MAS, a closed invariant set $R_m \subset R$ is called* stable in the sense of Lyapunov *w.r.t. $\mathcal{E}_a$ if for any $\epsilon > 0$, it is possible to find a quantity $\delta > 0$ such that when the metric $\rho(r_0, R_m) < \delta$ we have $\rho(\mathcal{R}(r_0, \mathcal{E}_k, k), R_m) < \epsilon$ for all $\mathcal{E}_k$ such that $\mathcal{E}_k\mathcal{E} \in \mathcal{E}_a(r_0)$ and $k \in \mathcal{Z}^+$ where $\mathcal{E}$ is an infinite event sequence, and $\mathcal{Z}^+$ is the set of*

positive integers. Moreover, if $\rho(\mathcal{R}(r_0, \mathcal{E}_k, k), R_m) \to 0$ *as* $k \to \infty$*, then the invariant set* R_m *is defined as being* asymptotically stable *w.r.t.* $\mathcal{E}_a$*.*

Here, the metric $\rho(.)$ is a generalized distance measure. Let $R_m \subset R$, then the distance from point r to the set R_m is denoted by $\rho(r,\ R_m) = \inf\{\rho(r,\ r') : r' \in R_m\}$. where inf(.) is the greatest lower bound of a set.

Definition 4.2.2 *For a CHA MAS, a closed invariant set* $R_m \subset R$ *is called* exponentially stable in the sense of Lyapunov *w.r.t.* $\mathcal{E}_a$ *if for* $r_0 \in$ *r-neighborhood,* $\rho(\mathcal{R}(r_0, \mathcal{E}_k, k), R_m) < \zeta e^{-\alpha k}\rho(r_0, R_m)$*, for some* $\alpha > 0$ *and some* $\zeta > 0$ *and for all* $\mathcal{E}_k$ *such that* $\mathcal{E}_k\mathcal{E} \in \mathcal{E}_a(r_0)$ *and* $k \in \mathcal{Z}^+$ *where* $\mathcal{E}$ *is an infinite event sequence, and* $\mathcal{Z}^+$ *is the set of positive integers.*

Given a coordination state $r \in R$ and a rule $l \in L$, r might satisfy or not satisfy l. Recall that we denote the fact that r satisfies l by $r \models l$. Based on the definitions and the CHA model we have described, we give the definition of the global stability of a CHA system.

Definition 4.2.3 *A CHA multi-agent system is said to be* globally stable *if*

1. *Each agent is locally stable based on Definition 4.1.1. The action executor of each agent can accomplish the hybrid actions so that the coordination states can transition according to* f_c*.*

2. *All the actions taken are on the allowed event trajectories* $\mathcal{E}_a$ *that lead the system to the goal set, and for* $r \in R$*,* $a \in A$*,* $c \in C$*, we have* $r \models l_o$ *holds and* $(a, l_o) \in C$*,* $r \models l_c$ *holds and* $(a, l_c) \in C$ *respectively.* $l_o \in L$ *defines an optimal action and* $l_c \in L$ *defines a constraint respectively.* L *is a set of coordination rules.*

3. *Our goal set of the abstract states* r*, the invariant set* $R_m \subset R$ *is stable in the sense of Lyapunov w.r.t.* $\mathcal{E}_a$*.*

Moreover, if the invariant set $R_m \subset R$ *is asymptotically stable in the sense of Lyapunov w.r.t.* $\mathcal{E}_a$*, the CHA multi-agent system is called* asymptotically stable*. If the invariant set* $R_m \subset R$ *is exponentially stable in the sense of Lyapunov w.r.t.* $\mathcal{E}_a$*, the CHA multi-agent system is called* exponentially stable*.*

In order to satisfy the third global stability requirement of a CHA system, we need to give the conditions for closed invariant set R_m to be stable. By applying the approach from [49] and [48], we have the following theorems for the abstract states of a CHA system.

First, the necessary and sufficient conditions for a closed invariant set to be stable are given.

Theorem 4.2.1 *(Passino et al. [49]) For a closed invariant set $R_m \subset R$ to be stable in the sense of Lyapunov w.r.t. $\mathcal{E}_a$, it is necessary and sufficient that in a sufficiently small r-neighborhood of the set R_m there exists a specified functional V with the following properties:*

1. For all sufficiently small $c_1 > 0$, it is possible to find a $c_2 > 0$ such that $V(r) > c_2$ for $r \in$ r-neighborhood of R_m and $\rho(r, R_m) > c_1$.

2. For any $c_4 > 0$ as small as desired, it is possible to find a $c_3 > 0$ so small that when $\rho(r, R_m) < c_3$ for $r \in$ r-neighborhood of R_m, we have $V(r) \leq c_4$.

3. $V(\mathcal{R}(r_0, \mathcal{E}_k, k))$ is a non increasing function for $k \in \mathcal{Z}^+$, as long as $\mathcal{R}(r_0, \mathcal{E}_k, k) \in$ r-neighborhood for all $\mathcal{E}_k$ such that $\mathcal{E}_k\mathcal{E} \in \mathcal{E}_a(r_0)$.

Here, the r-neighborhood of an arbitrary set $R_m \subset R$ is denoted by the set $S(R_m; r) = \{r : 0 < \rho(r, R_m) < r\}$.

Proof Proof for this theorem can be found in [49]. ∎

Then, the necessary and sufficient conditions for a closed invariant set to be asymptotically stable are given.

Theorem 4.2.2 *(Passino et al. [49]) For a closed invariant set $R_m \subset R$ to be asymptotically stable in the sense of Lyapunov w.r.t. $\mathcal{E}_a$, it is necessary and sufficient that in a sufficiently small r-neighborhood of the set R_m there exists a specified functional V having all the properties of Theorem 1 and, furthermore, $V(\mathcal{R}(r_0, \mathcal{E}_k, k)) \to 0$ as $k \to \infty$, for all $\mathcal{E}_k$ such that $\mathcal{E}_k\mathcal{E} \in \mathcal{E}_a(r_0)$ and for all $k \in \mathcal{Z}^+$ as long as $\mathcal{R}(r_0, \mathcal{E}_k, k) \in$ the r-neighborhood of the set R_m.*

Proof Proof for this theorem can be found in [49]. ∎

Finally, the sufficient conditions for a closed invariant set to be exponentially stable are given.

Theorem 4.2.3 *(Passino et al. [48]) In order for the invariant set R_m to be exponentially stable w.r.t $\mathcal{E}_a$, it is sufficient that in a sufficiently small r-neighborhood of the set R_m, there exists a specified functional V and three positive constants c_1, c_2, and c_3 such that $c_2 > c_3$ and*

1. $c_1\rho(r, R_m) \leq V(r) \leq c_2\rho(r, R_m)$, *and*
2. $V(\mathcal{R}(r_0, \mathcal{E}_{k+1}, k+1)) - V(\mathcal{R}(r_0, \mathcal{E}_k, k)) \leq -c_3(\rho(\mathcal{R}(r_0, \mathcal{E}_k, k), R_m))$,

for all $r_0 \in$ the r-neighborhood of the set R_m, for all $\mathcal{E}_k$ such that $\mathcal{E}_{k+1} = \mathcal{E}_k e$ ($e \in \mathcal{E}$), and $\mathcal{E}_{k+1}\mathcal{E} \in \mathcal{E}_a(r_0)$, $k \geq 0$.

Proof Proof for this theorem can also be found in [48]. ∎

4.3 Summary

In this chapter, the local stability and the global stability of a CHA MAS are defined. Conditions for a CHA MAS to be stable, asymptotically stable, and exponentially stable are also given. In the next chapter, we apply the stability analysis methodologies to analyze the stability of CHA MASs later.

Chapter 5

The Optimization of a CHA System

In the proposed framework, the control of the MASs is regarded as a decentralized control and coordination of agents. The CHA framework is able to implement coordination tasks for multi-agent systems. In this chapter, the optimization of MASs modeled by the CHA framework is studied.

There are numerous research done in the field of MASs and optimization techniques. Most of them concentrate on the decision making and optimization using MAS techniques. [24] gives an example of how expert systems techniques for distributed decision-making can be combined with contemporary numerical optimization techniques for the purposes of supply chain optimization and for software implementation. The system measures supply chain performance and the effect of different parameters in the replenishment control system. The system can be used to simulate the behavior of a system that uses optimization for part of its decision-making. [70] proposes a fuzzy multi-agent decision-making strategy to facilitate supplier management. A fuzzy model is used to evaluate the environmental performance of the suppliers and the life cycle environmental impact of the purchased product. Through analysis of manufacturer's business strategy, combined with other decision parameters, an optimal supplier is selected under fuzzy multi-criteria decision analysis. In [20], multi-agent constraint systems are considered with preferences, modeled as soft constraint systems in which variables and constraints are distributed among multiple autonomous agents. As a case study, it is considered as a distributed meeting scheduling problem where each agent has a pre-existing schedule and the agents must decide on a

common meeting that satisfies a given optimality condition. In [67], the problem of group decision making is considered where the selection process is based upon a group preference function, obtained by an aggregation of the participating agents individual preference functions. Individual agents strategically manipulate the information they provide, so as to further their own goal of getting their most preferred alternative selected by the group. In [2], a simple single-decider optimization model with a real application is described, and solution methodologies for optimal resource allocation fitting different scenarios (centralized, distributed, multi-agent) are discussed, identifying ranges of autonomy, quantifying rewarding and defining a negotiation protocol between the agents and the supervisor. [50] presents a methodology that, for the problem of scheduling of a single server on multiple products, finds a dynamic control policy via intelligent agents. The Reinforcement Learning approach was implemented via a multi-agent control architecture where a decision agent was assigned to each of the products.

In our study, we consider both time-driven dynamics and event-driven dynamics for the optimization of a CHA system. The optimization problem of the MASs is analyzed. An example is also given to illustrate how to define the optimization problem for a CHA. The direct identification algorithm is also introduced for solving the optimal control problem for a CHA MAS.

5.1 Problem Formulation

In this section, the optimization problem of a CHA MAS is formulated. In our CHA framework, each agent is modeled as a hybrid control layer and an intelligent coordination control layer. For a single agent, the controlled plant is at some initial physical state $x_{r_0}(t_0)$ at time t_0 and subsequently evolves according to the time-driven dynamics

$$\dot{x}_{r_0} = f_{p_{r_0}}(x_{r_0}, u_{r_0}, t), \tag{5.1}$$

where the subscript r_0 represents the initial abstract state. x is the continuous state, u is the continuous control signal, t represents time.

At time t_{r_0}, an event takes place. The abstract state becomes r_1 and the physical state becomes $x_{r_1}(t_{r_0})$. There might be a jump of the physical state at t_{r_0}. Therefore it

is possible that $x_{r_1}(t_{r_0}) \neq x_{r_0}(t_{r_0})$. Then the physical state subsequently evolves according to new time-driven dynamics with this initial condition. The time t_{r_0} at which this event happens, is called the temporal state of the agent. It depends on the event-driven dynamics of the form

$$t_{r_0} = w_{r_0}(t_0, x_{r_0}, u_{r_0}). \tag{5.2}$$

Let $r_k \in R$ represent the kth coordination state of a single agent. In general, after the abstract state switches from r_{k-1} to r_k at time $t_{r_{k-1}}$, the time-driven dynamics are given by

$$\dot{x}_{r_k} = f_{p_{r_k}}(x_{r_k}, u_{r_k}, t), \tag{5.3}$$

where the initial condition for x_{r_k} is $x_{r_k}(t_{r_{k-1}})$. The event-driven dynamics are given by

$$t_{r_k} = w_{r_k}(t_{r_{k-1}}, x_{r_k}, u_{r_k}). \tag{5.4}$$

Both the physical state x_{r_k} and the next temporal state t_{r_k} are affected by the choice of the control schemes at the abstract state r_k. Note that in order to solve the optimization problem, $t_{r_0}, t_{r_1}, t_{r_2}, \ldots, t_{r_k}$ are considered as temporal states intricately connected to the control of the system.

In a CHA MAS, events corresponding to the actions of one agent can be indexed as $k = 0, 1, \ldots, N_i - 1$, where subscript i represents the ith agent in the system. Each agent can be considered as a multi-stage process modeled as a single-server queuing system. The objective for the ith agent is to finish N_i actions. In the CHA framework, once the agent takes an action, it cannot be interrupted, and continues its task until it finishes it. Let $a_k \in A$ represent an enabled action for r_k. As an agent takes an action a_k, the physical state, denoted by x_{r_k}, evolves according to the time-driven dynamics of the form

$$\dot{x}_{r_k} = f_{p_{r_k}}(x_{r_k}, u_{r_k}, t), \tag{5.5}$$

where the initial condition for x_{r_k} is $x_{r_k}(t_{r_{k-1}})$. The continuous control variable u_{r_k} is used to attain a desired physical state.

If the time required to finish the kth action is s_{r_k} and $\Gamma_{r_k}(u_{r_k}) \subset \mathcal{R}^n$ is a given set that defines x_{r_k} satisfying the desired physical state, then the control signal u_{r_k} can be chosen

to satisfy the criteria

$$\begin{aligned} s_{r_k}(u_{r_k}) &= \min\Big(t \geq 0 : \big(x_{r_k}(t_{r_{k-1}} + t) \\ &= x_{r_k}(t_{r_{k-1}}) + \int_{t_{r_{k-1}}}^{t_{r_{k-1}}+t} f_{p_{r_k}}(x_{r_k}, u_{r_k}, \tau)d\tau\big) \\ &\in \Gamma_{r_k}(u_{r_k})\Big) \end{aligned} \tag{5.6}$$

where we can assume that under the best circumstance, i.e. without any disturbance, u_{r_k} is a fixed constant value at the abstract state r_k. The temporal state t_{r_k} of the kth action represents the time when the action finishes.

In a MAS, we have two or more agents interacting with each other in order to achieve a global goal. Therefore, when the ith agent finishes its kth action a_{k_i}, it might have to wait for the jth agent finishes its lth task a_{l_j} before the ith agent can starts its $(k+1)$th task $a_{(k+1)_i}$. Assume that agent i's tasks will only depend on agent j's tasks. Let $t_{a_{(k+1)_i}}$ represent the starting time of the $(k+1)$th action for the ith agent. In this case $t_{a_{(k+1)_i}} \neq t_{r_{k_i}}$. Instead, $t_{a_{(k+1)_i}} = t_{r_{l_j}}$ where the temporal state $t_{r_{l_j}}$ represents the time when the lth action of the jth agent finishes. Therefore, the event-driven dynamics of the temporal state $t_{r_{k_i}}$ of the ith agent can be represented by

$$t_{r_{k_i}} = \max(t_{r_{(k-1)_i}}, t_{r_{l_j}}) + s_{r_{k_i}}(u_{r_{k_i}}) \tag{5.7}$$

where $l = 0, 1, 2, \ldots, N_j - 1$, j is the index of the agent and $j \neq i$. Thus if action a_{k_i} does not depend on the completion of any actions of any other agent, Equation 5.7 can be simplified as

$$t_{r_{k_i}} = t_{r_{(k-1)_i}} + s_{r_{k_i}}(u_{r_{k_i}}). \tag{5.8}$$

One may notice that we need to set $t_{0_i} = 0$ to make sure that $t_{r_{0_i}} = t_{r_{l_j}} + s_{r_{0_i}}(u_{r_{0_i}})$ and $t_{r_{0_i}} = s_{r_{0_i}}(u_{r_{0_i}})$ in case action a_{0_i} does not depend on the completion of other actions.

In order to simplify the optimization problem for the ith agent, we assume that the temporal states $t_{r_{l_j}}$ of the lth action of the jth agent are known. Then we can see that when $t_{r_{l_j}} > t_{r_{k_i}}$, there is an idle period in the interval $[t_{r_{k_i}}, t_{r_{l_j}}]$ during which the physical state of the ith agent does not change.

Therefore, the optimization problem for the ith agent of the CHA framework becomes the optimization problem of the hybrid control layer, i.e., the optimization of the hybrid

system that combines the time-driven dynamics in Equation 5.4 and the event-driven dynamics in Equation 5.7.

The optimization problem to be solved for the ith agent takes the general form

$$\min_{u_{r_0},\ldots,u_{r_{N_i-1}}} \Big(\sum_{k=0}^{N_i-1} L_{r_k}(t_{r_{k_i}}, u_{r_{k_i}}) \Big) \tag{5.9}$$

where $L_{r_k}(t_{r_{k_i}}, u_{r_{k_i}})$ is the cost function defined for the kth action of the ith agent in the system. The cost function has been defined without including $x_{r_{k_i}}$ because $x_{r_{k_i}}$ is supposed to reach the desired value as defined in Equation 5.6 which gives the $s_{r_k}(u_{r_k})$ for the ith agent here.

Notice that for the optimization problem defined in Equation 5.9, the index $k = 0, 1, 2, \ldots, N_i - 1$ does not count time steps, but rather asynchronous actions. Rewriting Equation 5.9, we can represent the optimization problem as

$$\min_{u_{r_0},\ldots,u_{r_{N_i-1}}} \Big(\sum_{k=0}^{N_i-1} \big(\phi(t_{r_{k_i}}) + \theta(u_{r_{k_i}})\big) \Big). \tag{5.10}$$

Example 5.1.1 *To illustrate how to formulate the optimization problem using the model discussed above, we present the optimal control problem for the multi-agent system involving an overhead crane and a mobile robot as mentioned in Scenario 1.1.2. We consider the optimization problem for the mobile robot.*

The mobile robot needs to finish various tasks in order to coordinate and cooperate with the overhead crane to achieve the final goal of the multi-agent system. The "quality" of the work of the mobile robot has to be maintained otherwise the cooperation would not be possible. For example, if the mobile robot turns too early or too late, the mobile robot would not stay on track when it gets into the overhead crane's workspace and it will fail the task. As a result, the whole system would not complete the mission. In particular, the mobile robot's actions involve searching the landmark, aligning its body to the target, landmark following, turning left, turning right, and so on. The goal of the whole system is that the mobile robot needs to move into the overhead crane's workspace and wait there, until the overhead crane finishes its dropping action. Then, the mobile robot takes the object that the overhead crane has dropped and transports the object out of the overhead crane's workspace.

From this example, we can see that the actions of the agents are different from the jobs defined in [12]. The actions that we have defined are different tasks. In [12], jobs are done for different products by the single-stage process. However, in our framework, each agent has to take various actions on a single object in order to accomplish the overall task. Of course, we can also apply the model to systems that require the same task to be done on multiple products.

In this example, the coordination is defined in the coordination rule base. The mobile robot needs certain amount of time to finish the task. The position of the mobile robot is critical for the cooperation. The distances between the mobile robot and the landmarks are used to determine the quality level of the actions. In a sufficiently large empty space, a mobile robot can be driven to any position with any orientation, hence the robot's configuration space has three dimensions, two for translation and one for rotation. The physical state of the kth action of the mobile robot is denoted by $(x_{r_k},\ y_{r_k},\ \theta_{r_k})$ *and represents the translational and rotational position of the mobile robot. Thus the mobile robot can be illustrated as the model shown in Figure 5.1, the wheels are aligned with the vehicle. The kinematic model of the mobile robot can be represented as*

$$\begin{aligned} \dot{x}_{r_k} &= u_{1_{r_k}} \cos\theta_{r_k}, \\ \dot{y}_{r_k} &= u_{1_{r_k}} \sin\theta_{r_k}, \\ \dot{\theta}_{r_k} &= u_{2_{r_k}}, \end{aligned} \tag{5.11}$$

where $u_{1_{r_k}}$ *corresponds to the forward velocity of the vehicle and the angle of the vehicle body with respect to the horizontal line is* θ_{r_k}*, the angular velocity of the vehicle body is* $u_{2_{r_k}}$*,* $(x_{r_k},\ y_{r_k})$ *is the location of the center point of the robot. The forward velocity* $u_{1_{r_k}}$ *and the angular velocity* $u_{2_{r_k}}$ *are used to control the motion of the mobile robot.*

The path of the mobile robot can be obtained by integrating Equation 5.11. Since the mobile robot needs to take a series of actions to achieve the global goal for the multi-agent system, we use the subscript r_k *to represent the abstract state of the mobile robot which indicates which action the robot is taking.*

Next, the temporal state of the kth action of the mobile robot represents the time when the mobile robot starts the next action. Let $t_{r_{l_j}}$ *be the ending time of the lth action of the jth agent that the mobile robot depends on in order to finish its own action, the event-driven*

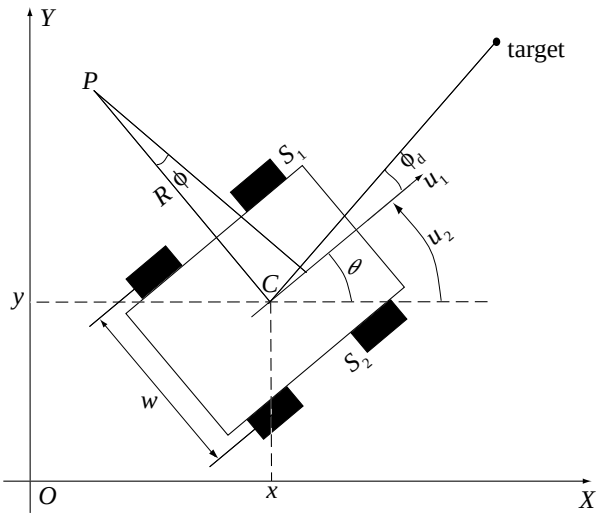

Figure 5.1: The Model of the 4-Wheeled Mobile Robot.

dynamics describing the evolution of the temporal states of the mobile robot are given by

$$t_{r_k} = \max(t_{r_{(k-1)}}, t_{r_{l_j}}) + s_{r_k}(u_{r_k}), \tag{5.12}$$

where $s_{r_k}(u_{r_k})$ *is the time for the mobile robot to finish the* k*th action. Notice that we have omitted subscript* i *for simplicity. In this system, we consider two control objectives: 1) Increasing the performance of the mobile robot, and 2) Reducing the time for the mobile robot to finish all the tasks. Thus, the optimal control problem of interest can be expressed as:*

$$\min_{u_{r_0},\ldots,u_{r_{N_i-1}}} \Big(\sum_{k=0}^{N_i-1} \big(\phi(t_{r_k}) + \theta(u_{r_k})\big)\Big). \tag{5.13}$$

The function $\phi(t_{r_k})$ *above is the cost related to the time an action is finished and the time the task it depends on is finished. Generally, if the robot moves slower, its performance is better. The function* $\theta(u_{r_k})$ *is the cost function to penalize lower speed since we want the robot to finish its tasks faster. As an example, we can choose* $\phi(t_{r_k}) = |t_{r_k} - t_{r_{l'_j}}|$ *and* $\theta(u_{r_k}) = \frac{1}{u_{r_k}}$.

5.2 Necessary Conditions

Recall the optimal control problem described in Equation 5.9, we define the following augmented cost for the ith agent by extending the analysis procedures for a single stage system proposed in [10] to a multi-agent system (for simplicity, the subscript i is omitted)

$$\begin{aligned}\bar{J}(t,\lambda,u) &= \sum_{k=0}^{N_i-1}\Big(L_{r_k}(t_{r_k},u_{r_k}) \\ &\quad +\lambda_{r_k}[\max(t_{r_{k-1}},t_{r_{l_j}})+s_{r_k}(u_{r_k})-t_{r_k}]\Big),\end{aligned} \tag{5.14}$$

where t and u are N_i-dimensional vectors for the temporal state and the control, and λ is an N_i-dimensional vector for the costate sequence used to adjoin the temporal dynamics to the cost function. Throughout the rest of the analysis, the following assumptions are made.

Assumption 5.2.1 *(Cassandras et al. [10]) The one-step cost $L_{r_k}(.,.)$ and the service functions $s_{r_k}(.)$ are continuously differentiable for all $k = 0, 1, \ldots, N_i - 1$.*

Assumption 5.2.2 *(Cassandras et al. [10]) The service functions $s_{r_k}(.)$ are monotonically increasing for all $k = 0, 1, \ldots, N_i - 1$.*

Assumption 5.2.2 can be service functions that are monotonically decreasing, depending on the nature of the control variables u_{r_k}, yielding dual results to those we will subsequently derive.

Given Assumption 5.2.1, the augmented cost $\bar{J}$, as the sum of Lipschitz functions, is itself a Lipschitz function. Such functions are continuous, but not everywhere differentiable. For Lipschitz functions, non smooth optimization gives the necessary conditions for optimality [10]. Suppose $f : \mathcal{R}^n \to \mathcal{R}$ is a locally Lipschitz continuous function of $u \in \mathcal{R}$. The necessary condition for the optimization of non smooth Lipschitz functions is given in terms of $\partial f(u)$. Our task now is to identify $\partial \bar{J}$.

If we ignore the non differentiability associated with the "max" operation in Equation 5.14, the standard first-order necessary conditions for optimality require that

$$\frac{\partial \bar{J}}{\partial u_{r_k}} = 0, \tag{5.15}$$

$$\frac{\partial \bar{J}}{\partial \lambda_{r_k}} = 0, \tag{5.16}$$

$$\frac{\partial \bar{J}}{\partial t_{r_k}} = 0, \tag{5.17}$$

for all $k = 0, 1, \ldots, N_i - 1$. Equation 5.15 gives

$$\frac{\partial L_{r_k}(t_{r_k}, u_{r_k})}{\partial u_{r_k}} + \lambda_{r_k} \frac{ds_{r_k}(u_{r_k})}{du_{r_k}} = 0. \tag{5.18}$$

Equation 5.16 gives the state equation

$$t_{r_k} = \max(t_{r_{k-1}}, t_{r_{l_j}}) + s_{r_k}(u_{r_k}) \tag{5.19}$$

with initial condition $t_0 = -\infty$. Finally, Equation 5.17 gives

$$\lambda_{r_k} = \frac{\partial L_{r_k}(t_{r_k}, u_{r_k})}{\partial t_{r_k}} + \lambda_{r_{k+1}} \frac{d \max(t_{r_k}, t_{r_{l'_j}})}{dt_{r_k}} \tag{5.20}$$

with boundary condition

$$\lambda_{r_{N_i-1}} = \frac{\partial L_{r_{N_i-1}}(t_{r_{N_i-1}}, u_{r_{N_i-1}})}{\partial t_{r_{N_i-1}}}. \tag{5.21}$$

Assumption 5.2.3 *In this study, we assume the task of one agent only depends on one other agent. When we optimize agent i, we assume that the actions of agent j that agent i depends on are constants.*

Notice that $r_{l'_j}$ of $t_{r_{l'_j}}$ represents a different state of agent j. This is necessary because for a different task of agent i, it does not depend on the same task of agent j. We can see that if the task of agent i at state r_k does not depend on any other tasks of agent j, the optimization problem becomes a similar problem as defined in [10] that can be analyzed

with the results shown in [10]. In this study, we provide analysis for tasks of an agent depend on another agent.

Equations 5.18-5.21 define a boundary-value problem, in which the solution provides a control sequence satisfying the necessary conditions for optimality. The problem becomes complicated because of the presence of the "max" function in Equation 5.20. This function is Lipschitz continuous, differentiable in t_{r_k} everywhere except at the single point where $t_{r_k} = t_{r_{l'_j}}$ with

$$\frac{d}{dt_{r_k}} \max(t_{r_k}, t_{r_{l'_j}}) = 0, \tag{5.22}$$

if $t_{r_k} < t_{r_{l'_j}}$;

$$\frac{d}{dt_{r_k}} \max(t_{r_k}, t_{r_{l'_j}}) = 1, \tag{5.23}$$

if $t_{r_k} > t_{r_{l'_j}}$.

At the point where $t_{r_k} = t_{r_{l'_j}}$, the left and right derivatives clearly exist, given by 0 and 1, respectively. As the system operates, the sequence of arrival and departure times define a sample path. On any sample path, the points where $t_{r_k} = t_{r_{l'_j}}$, acquire special significance, since they are responsible for the non differentiability of the "max" function in Equation 5.20.

5.3 Non Smooth Optimization of a CHA MAS

In order to identify $\partial \bar{J}$, we introduce the following terminology for a CHA MAS:

Definition 5.3.1 *In a CHA MAS, an action a_{r_k} is called a* normal action *of the ith agent if it does not depend on the action of another agent (i.e. $t_{r_{k-1}} + s_{r_k}(u_{r_k}) = t_{r_k}$).*

Definition 5.3.2 *A* critical action *with index r_k of agent i is an action that satisfies $t_{r_k} = t_{r_{l_j}^k}$. Superscript k of r is used to represent the state of agent j that the kth action of agent i depends on.*

Definition 5.3.3 *An* idle period *for the ith agent in a CHA MAS is a time interval $(t_{r_k}, t_{r_{l_j}}]$ such that $t_{r_k} < t_{r_{l_j}}$ for any $k = 0, 1, \ldots, N_i - 1$.*

Definition 5.3.4 *A* busy period *for the ith agent in a CHA MAS is a continuous set of actions,* $a_k, \ldots, a_n$ *for* $1 \leq k \leq n \leq N$ *such that the following conditions are satisfied:*

1. $t_{r_k} < t_{r_{l_j}}$;
2. $t_{r_{n+1}} < t_{r_{l_j}^{n+1}}$;
3. $t_{r_s} \geq t_{r_{l_j}^{s}}$, *for every* $s = k, \ldots, n$ *when action* a_s *depends on an action of agent* j.

Superscript s *of* r *is used to represent the state of agent* j *that the* s*th action of agent* i *depends on.*

From Definition 5.3.3, we can see that an idle period is a time interval of strictly positive value during which the agent has no other actions to take but a null action. From Definition 5.3.4, we can see that a busy period is a time interval during which the agent takes a series of actions without any interruption, and without waiting for the other agent. A busy period, initiated at time $t_{r_{l_j}}$, always follows an idle period and be followed by another idle period. There is no idle periods within a busy period.

Notice that a critical action corresponds to the situation where the "max" function is not differentiable in Equation 5.20. Notice that a critical action does not end a busy period while a busy period may contain one or more critical actions.

Assumption 5.3.1 *In agent* i*, if action* a_{r_k} *is a normal action, then the next action* $a_{r_{k+1}}$ *starts at the same time as* a_{r_k} *finishes.*

In order to identify the busy period structure and the locations of critical actions within a busy period, we associate with every action $a_{r_k}, k = 0, \ldots, N_i - 1$ the following indice

$$n_k = \min(n \geq k : t_{r_n} < t_{r_{l_j}^{n+1}}) \tag{5.24}$$

$$m_k = \min(m \geq k : t_{r_m} \leq t_{r_{l_j}^{m+1}}) \tag{5.25}$$

It can be seen that n_k is the index of the last action in the busy period containing action a_{r_k}. For m_k, if action a_{r_k} is critical or there are critical actions between action a_{r_k} and the end of its busy period, then m_k is the index of the first such critical action. Thus

$m_k < n_k$ and we have $t_{r_{m_k}} = t_{r_{l_j}^{m_k}}$. If, on the other hand, action a_{r_k} is not critical and there are no critical actions between action a_{r_k} and the end of its busy period, then m_k is the index of the action that ends the busy period ($m_k = n_k$). Then we have the following analysis.

- $m_k = n_k$

  This means that action a_{r_k} is not critical, there are no critical actions between action a_{r_k} and the end of its busy period, and we have $\max(t_{r_p}, t_{r_{l_j}^p}) = t_{r_p}$ for all $p = k, \ldots, n_k - 1$ and $\max(t_{r_{n_k}}, t_{r_{l_j}^{n_k}}) = t_{r_{l_j}^{n_k}}$. Therefore, all derivatives in Equation 5.14 exist under Assumption 5.3.1 and we get

$$\lambda_{r_k} = \sum_{p=k}^{n_k} \frac{\partial L_{r_p}}{\partial t_{r_p}}. \tag{5.26}$$

  Then, the optimality condition 5.15 becomes

$$\frac{\partial \bar{J}}{\partial u_{r_k}} = \frac{\partial L_{r_k}}{\partial u_{r_k}} + \frac{ds_{r_k}}{du_{r_k}} \sum_{p=k}^{n_k} \frac{\partial L_{r_p}}{\partial t_{r_p}} = 0. \tag{5.27}$$

  Recalling Assumption 5.2.1, we have the following lemma.

  Lemma 5.3.1 *With Assumption 5.2.1 and Assumption 5.3.1, if $m_k = n_k$, then $\bar{J}(.)$ is continuously differentiable in u_{r_k}, and the optimality condition is*

$$\frac{\partial \bar{J}}{\partial u_{r_k}} = \frac{\partial L_{r_k}}{\partial u_{r_k}} + \frac{ds_{r_k}}{du_{r_k}} \sum_{p=k}^{n_k} \frac{\partial L_{r_p}}{\partial t_{r_p}} = 0. \tag{5.28}$$

  Proof The proof follows the analysis above. ∎

  We can see that if $m_k = n_k$ for all $k = 0, \ldots, N_i - 1$, then the function $\bar{J}$ would be differentiable.

- $m_k < n_k$

  $\bar{J}$ exhibits non differentiability associated with critical actions, when $m(i) < n(i)$. For any critical action, we have $\max(t_{r_{m_k}}, t_{r_{l_j}^{m_k}}) = t_{r_{m_k}} = t_{r_{l_j}^{m_k}}$ and the corresponding derivative in Equation 5.20 does not exist. Since actions a_{r_k} and $a_{r_{m_k}}$ are in the same busy period and $m_k \geq k$, we have

$$t_{r_{m_k}} = \max(t_{r_{k-1}}, t_{r_{l_j}^{k-1}}) + s_{r_k}(u_{r_k}) + \sum_{p=k+1}^{m_k} s_{r_p}(u_{r_p}) \tag{5.29}$$

  where the "max" accounts for the fact that action a_{r_k} may be the first in the busy period. Through Equation 5.29 we see that the control for action a_{r_k} affects the departure time of action m_k. Suppose that we fix all other control variables and adjust u_{r_k}. Recalling Equation 5.22 and Equation 5.23, we have the following one-sided derivatives:

$$\lim_{t_{r_{m_k}} \uparrow t_{r_{l_j}^k}} \frac{d}{dt_{r_{m_k}}} \max(t_{r_{m_k}}, t_{r_{l_j}^k}) = 0, \tag{5.30}$$

$$\lim_{t_{r_{m_k}} \downarrow t_{r_{l_j}^k}} \frac{d}{dt_{r_{m_k}}} \max(t_{r_{m_k}}, t_{r_{l_j}^k}) = 1. \tag{5.31}$$

  The limit in Equation 5.30 represents adjusting u_{r_k} so that $t_{r_{m_k}}$ increases toward $t_{r_{l_j}^k}$. It is same for other critical actions between m_k and n_k. Combining Equation 5.14 and Equation 5.20, we have

$$\begin{aligned} \frac{\partial \bar{J}}{\partial u_{r_k}} &= \frac{\partial L_{r_k}}{\partial u_{r_k}} + \frac{ds_{r_k}}{du_{r_k}} \\ &\quad \Big(\sum_{p=k}^{m_k} \frac{\partial L_{r_p}}{\partial t_{r_p}} + \lambda_{m_k+1} \frac{d \max(t_{r_{m_k}}, t_{r_{l_j}^k})}{dt_{r_{m_k}}} \Big). \end{aligned} \tag{5.32}$$

  Recall Assumption 5.2.2 and Equation 5.29, $t_{r_{m_k}}$ is monotonically increasing with $u(r_k)$. With Equation 5.30 and Equation 5.31, the above equation leads to the one-sided derivative

$$(\frac{\partial \bar{J}}{\partial u_{r_k}})^- = \frac{\partial L_{r_k}}{\partial u_{r_k}} + \frac{ds_{r_k}}{du_{r_k}} \sum_{p=k}^{m_k} \frac{\partial L_{r_p}}{\partial t_{r_p}} \tag{5.33}$$

and

$$(\frac{\partial \bar{J}}{\partial u_{r_k}})^+ = \frac{\partial L_{r_k}}{\partial u_{r_k}} + \frac{ds_{r_k}}{du_{r_k}} \sum_{p=k}^{n_k} \frac{\partial L_{r_p}}{\partial t_{r_p}}. \tag{5.34}$$

regardless of whether one or more critical actions are present between k and n_k. Then we have the following Lemma.

Lemma 5.3.2 *Under Assumption 5.2.1 and Assumption 5.2.2, for every* $k = 0, \ldots, N_i - 1$,

$$(\frac{\partial \bar{J}}{\partial u_{r_k}})^+ = (\frac{\partial \bar{J}}{\partial u_{r_k}})^- + \frac{ds_{r_k}}{du_{r_k}} \sum_{p=m_k+1}^{n_k} \frac{\partial L_{r_p}}{\partial t_{r_p}}. \tag{5.35}$$

Proof From Equation 5.33 and Equation 5.34

$$\begin{aligned}
(\frac{\partial \bar{J}}{\partial u_{r_k}})^+ &= \frac{\partial L_{r_k}}{\partial u_{r_k}} + \frac{ds_{r_k}}{du_{r_k}} \sum_{p=k}^{n_k} \frac{\partial L_{r_p}}{\partial t_{r_p}} \\
&= \frac{\partial L_{r_k}}{\partial u_{r_k}} + \frac{ds_{r_k}}{du_{r_k}} \sum_{p=k}^{m_k} \frac{\partial L_{r_p}}{\partial t_{r_p}} + \frac{ds_{r_k}}{du_{r_k}} \sum_{p=m_k+1}^{n_k} \frac{\partial L_{r_p}}{\partial t_{r_p}} \\
&= (\frac{\partial \bar{J}}{\partial u_{r_k}})^- + \frac{ds_{r_k}}{du_{r_k}} \sum_{p=m_k+1}^{n_k} \frac{\partial L_{r_p}}{\partial t_{r_p}}.
\end{aligned} \tag{5.36}$$

Thus Equation 5.35 holds. ∎

As a conclusion, we have the following theorem.

Theorem 5.3.3 *Under Assumption 5.2.1 and Assumption 5.2.2, an optimal control* u_{r_k} *for the* i*th agent of a CHA MAS,* $k = 0, \ldots, N_i - 1$*, satisfies the following conditions:*

1.

$$0 \in \frac{\partial \bar{J}}{\partial u_{r_k}} = [\min\big((\frac{\partial \bar{J}}{\partial u_{r_k}})^-, (\frac{\partial \bar{J}}{\partial u_{r_k}})^+\big), \max\big((\frac{\partial \bar{J}}{\partial u_{r_k}})^-, (\frac{\partial \bar{J}}{\partial u_{r_k}})^+\big)] \subset \mathcal{R},$$

where

$$(\frac{\partial \bar{J}}{\partial u_{r_k}})^- = \frac{\partial L_{r_k}}{\partial u_{r_k}} + \frac{ds_{r_k}}{du_{r_k}} \sum_{p=k}^{m_k} \frac{\partial L_{r_p}}{\partial t_{r_p}},$$

$$(\frac{\partial \bar{J}}{\partial u_{r_k}})^+ = \frac{\partial L_{r_k}}{\partial u_{r_k}} + \frac{ds_{r_k}}{du_{r_k}} \sum_{p=k}^{n_k} \frac{\partial L_{r_p}}{\partial t_{r_p}},$$

$$n_k = \min(n \geq k : t_{r_n} < t_{r_{l_j}^{n+1}}),$$

and

$$m_k = \min(m \geq k : t_{r_m} \leq t_{r_{l_j}^{m+1}}).$$

2. $t_{r_k} = \max(t_{r_{k-1}}, t_{r_{l_j}^{k-1}}) + s_{r_k}(u_{r_k})$, $t_0 = -\infty$.

Proof The proof follows directly from the necessary condition of non smooth optimization $0 \in \frac{\partial \bar{J}}{\partial u_{r_k}}$, and from Lemma 5.3.1 and Lemma 5.3.2. ∎

Remark 5.3.1 *We can see that when $m_k < n_k$, we have*

$$\frac{\partial \bar{J}}{\partial u_{r_k}} = [\min\big((\frac{\partial \bar{J}}{\partial u_{r_k}})^-, (\frac{\partial \bar{J}}{\partial u_{r_k}})^+\big), \max\big((\frac{\partial \bar{J}}{\partial u_{r_k}})^-, (\frac{\partial \bar{J}}{\partial u_{r_k}})^+\big)] \subset \mathcal{R}. \tag{5.42}$$

When $m_k = n_k$, we get $(\frac{\partial \bar{J}}{\partial u_{r_k}})^- = (\frac{\partial \bar{J}}{\partial u_{r_k}})^+$, in which case $\frac{\partial \bar{J}}{\partial u_{r_k}}$ defined by the closed interval above is a singleton.

Recalling Lemma 5.3.1, we see that when $m_k = n_k$ (i.e., when action a_{r_k} is not critical and there are no critical actions between action a_{r_k} and the end of its busy period), the first condition of the theorem simply requires that $(\frac{\partial \bar{J}}{\partial u_{r_k}})^- = (\frac{\partial \bar{J}}{\partial u_{r_k}})^+ = 0$.

For $L_{r_k}(.,.)$ and $s_{r_k}(.)$, when $m_k < n_k$, generally neither $(\frac{\partial \bar{J}}{\partial u_{r_k}})^- = 0$ nor $(\frac{\partial \bar{J}}{\partial u_{r_k}})^+ = 0$ (i.e., in general, zero is not an end point of the interval defining $\partial \bar{J}_{r_k}$). Therefore, when $m_k < n_k$, the first condition of the above theorem requires that $(\frac{\partial \bar{J}}{\partial u_{r_k}})^-$ and $(\frac{\partial \bar{J}}{\partial u_{r_k}})^+$ have opposite signs (i.e., $(\frac{\partial \bar{J}}{\partial u_{r_k}})^-(\frac{\partial \bar{J}}{\partial u_{r_k}})^+ < 0$).

5.4 Direct Identification Algorithm

In this section, we will propose the direct identification algorithm for the optimization of a CHA MAS. The direct identification algorithm is able to identify the busy periods to optimize a CHA MAS.

Definition 5.4.1 *A* busy-period structure *is a partition of the actions* $a_{r_0}, \ldots, a_{r_{N_i-1}}$ *into busy periods.*

Lemma 5.4.1 *If actions* $a_{r_k}, \ldots, a_{r_{n_k}}$ *constitute a single busy period on the optimal sample path, then the optimal control* $u^*_{r_k}$ *is identical to the optimal control* $u^*(k, n_k)$*, where* $u^*_{r_k}$ *represents the optimal control of the busy period while* $u^*(k, n_k)$ *represents part of the optimal control path.*

Proof Considering Theorem 5.3.3, notice that the state equation does not propagate information across the idle period preceding the busy period containing action a_{r_k}. The control for action a_{r_k} does not depend on actions preceding the busy period. In addition, the control does not depend on actions in the following busy periods either. Thus the optimal control $u^*_{r_k}$ for a busy period is unique. ∎

Remark 5.4.1 *Lemma 5.4.1 tells us that we can solve the optimal control problem of agent i by identifying the busy periods and solve the optimal control problem for each busy period.*

Lemma 5.4.2 *If actions* $a_{r_k}, \ldots, a_{r_{k_n}}$ *constitute a single busy period on the optimal sample path, then*

1. $t^*_{r_k} < t_{r_{l_j}}$;
2. $t^*_{r_{n+1}} < t_{r^{n+1}_{l_j}}$;
3. $t^*_{r_s} \geq t_{r^s_{l_j}}$, *for every* $s = k, \ldots, n$, *where superscript* $*$ *represents that the variable is for the optimal sample path.*

Proof The proof follows Definition 5.3.4. ∎

Theorem 5.4.3 *Under Assumption 5.2.1 and Assumption 5.2.2, the busy period structure of an optimal sample path is unique in the sense that for any* a_{r_k}, $k = 0, \ldots, N_i - 1$, *the last action of the busy period containing* a_{r_k} *is unique on the optimal sample path.*

Proof The proof is by contradiction. Suppose that there exist two different busy period structures that both satisfy optimality. We can assume that the difference between the two busy period structures is in their respective first busy periods. Denote the two busy period structures by A and B. Let r_A be the last state in the first busy period on busy period structure A, and r_B be the last state in the first busy period on busy period structure B. Assume that $t_{r_A} < t_{r_B}$. Recall Lemma 5.4.1, the optimal control u_{r_k} in both busy period structures should result in the same $t^*_{r_A}$. From Lemma 5.4.2 and the busy period structure A, we have

$$t^*_{r_A} < t_{r^A_{l_j}}. \tag{5.43}$$

On the other hand, from Lemma 5.4.2 and the busy period structure B, we have

$$t^*_{r_A} \geq t_{r^A_{l_j}}. \tag{5.44}$$

This causes contradiction, the busy periods must coincide, and the proof is complete. ∎

Based on Theorem 5.4.3, for each single agent modeled using the CHA framework, the direct identification algorithm is proposed to solve the optimization problem. The idea is that we can identify the busy period structure of an agent by optimizing all busy periods.

In the algorithm, N represents the number of agent in the MAS. $Q(k, n)$ represents the sub-optimization problem of a sequence of actions that starts from state r_k and ends at state n. In the algorithm, we assume that agent 0 depends on some external events, and agent i depends on actions of previous agents, $i = 1, \ldots, N-1$. By applying the direct identification algorithm for the optimization of the CHA framework, we are able to optimize the performance and time of agents in a CHA MAS based on the given cost function.

Remark 5.4.2 *In the direct identification algorithm listed in Table 5.1, the index k always indicates the first action of all busy periods on the optimal sample path. This is because k is updated as $k = n+1$ in Step 2 only when the last action of a busy period is identified to be $a_{r_{n-1}}$.*

Table 5.1: The Direct Identification Algorithm

	for $i = 0$ to $N - 1$
Step 0	initialize $k = 0$, $n = 0$;
	while $n \leq N_i - 1$ do
Step 1	solve the sub-optimization problem $Q(k, n)$;
Step 2	identify busy periods:
	if $t_{r_n} < t_{r^n_{l_j}}$ then
	$k \leftarrow n + 1$
	end if
Step 3	increment index n
	$n \leftarrow n + 1$
	end while
	end for

5.5 Summary

In this chapter, we give an analysis of the optimization of a CHA MAS. In our study, we consider both time-driven dynamics and event-driven dynamics for the optimization of a CHA system. We have proposed some useful definitions and theorems to solve the optimization problem. An example is also given to illustrate how to define the optimization problem for a CHA. The direct identification algorithm is also introduced for solving the optimal control problem for a CHA MAS. In the next chapter, we will give examples and experiments to illustrate how to utilize the proposed framework to develop the control scheme for multi-agent systems.

Chapter 6

Experiments and Simulations

This chapter gives some experimental and simulation results for systems modeled using the proposed framework. The goal is to implement the tools we have introduced to develop the control algorithm for multi-agent systems. The feasibility of the proposed framework is illustrated through three different scenarios. It is demonstrated that the proposed framework is generic and can be applied to the control of both homogeneous and heterogeneous MASs.

6.1 The Multi-Crane Cooperation System

In this experiment, the proposed framework is applied to the homogeneous multi-agent system as introduced in Scenario 1.1.1. The control of a multi-crane system composed of two industrial overhead cranes operating in the same workspace is studied using the proposed framework. The goal of the control of this multi-crane system is to control the two cranes to move the payloads in the same workspace without any collision. The results are also reported in [33].

6.1.1 The Overhead Crane

Overhead cranes are widely used in many fields, such as the shipping, mining, manufacturing and automotive industries where the overhead cranes are used to move loads from

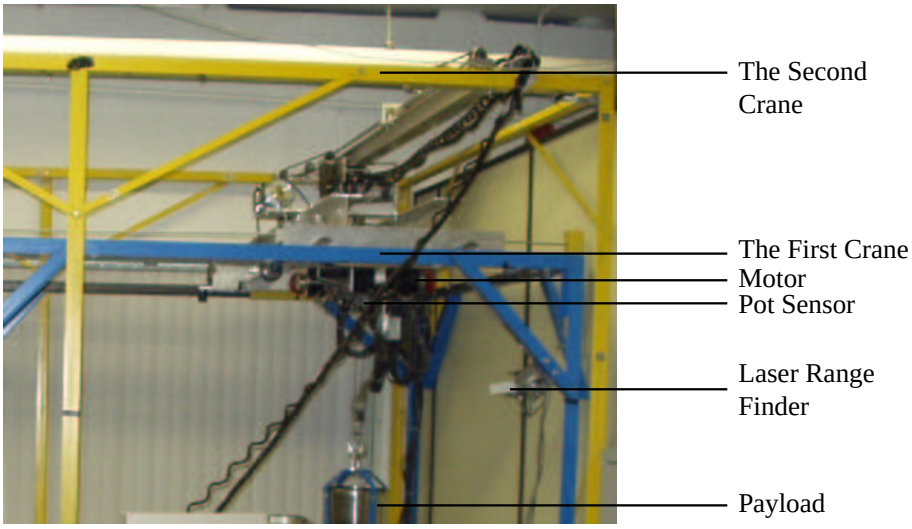

Figure 6.1: The Overhead Crane.

one place to another. The crane in this multi-agent system has three DC motors for the three dimension workspace. Two potentiometers are connected to measure the swing angles. The overhead cranes used in this study are shown in Figure 6.1. Details about the overhead crane can be found in [65].

The crane is considered as a hybrid system whose dynamics are controlled by the hybrid agent. At the initial stage, the initial discrete and continuous values for the system are set. We have discrete values such as the direction of movement and the brake status. We have the continuous values such as the speed of the trolley.

If the value of $zBrake$ turns from $true$ to $false$, $speedZ$ will jump from 0 to the desired speed (ideally). If $flagMoveToTarget$ turns from $false$ to $true$, the crane will jump from the idle state to the moving state. For the developed crane system, we also have autonomous jumps. When the limit switches are triggered, the crane will stop moving and the speed for that direction will jump to zero.

f_p determines the evolution of the continuous state. We can control the speed of the motors in order to control the speed of the payload v. The position of the payload (x_c, y_c, z_c)

in the coordinate of the workspace can be described as

$$\begin{aligned} x_c &= x + l \sin\theta_x \cos\theta_y, \\ y_c &= y + l \sin\theta_y, \\ z_c &= -l \cos\theta_x \cos\theta_y, \end{aligned} \tag{6.1}$$

where $(x_c,\ y_c,\ z_c)$ is the position of the payload in the trolley coordinate, x and y are the travel and traverse positions, l is the length of the string, θ is the swing angle of the payload that can be decomposed into the travel direction θ_x and traverse direction θ_y.

The action executor

The following actions are designed for the overhead crane: *anti-swing, pick up, move to, put down, cross over* and *stop*. The design of the actions can be found in [33].

The abstract states

For the crane, we have the following abstract states:

1_c idle;

2_c object picked up;

3_c load has been transferred to target;

4_c load has been put down;

5_c picked up without the load;

6_c backed to the initial position;

7_c put down without load;

8_c swinging stopped;

9_c move away;

10_c collision avoided.

6.1.2 Modeling the System Using the Proposed Framework

In this section, the proposed framework is applied to model the control of the multi-crane Cooperation system.

The intelligent coordination control layer

In the intelligent coordination control layer, the states of the controlled process are mapped to the abstract states. The crane starts at the initial state 1_c. The coordination

input $q^{\triangleleft} \in Q^{\triangleleft}$ and $x^{\triangleleft} \in X^{\triangleleft}$ are the abstract state $r \in R$ of the other crane and the position of the other crane. Since we only have two agents in the system, we do not need the input signature. Here $q_s \in Q_s$ is obtained from the user's graphical interface which tells if the load is ready or not. $x_s \in X_s$ determines the starting position and the target position of the payload. Based on the coordination rule base, β gives the desired action. The coordination rule base also determines the enable function g. After the desired hybrid action is executed by the action executor, the system transitions to the next abstract state. f_r outputs the continuous reference to the action executor, for example, the target position x_t. f_o and ϕ_o communicate the current position and abstract state to the other agent, respectively.

The coordination rule base

Based on the nature of the multi-crane system, the following coordination rule base is defined.

The desired choices are:

1_c (*pick up*) 2_c

2_c (*move to*) 3_c

3_c (*put down*) 4_c

4_c (*pick up*) 5_c

5_c (*move to*) 6_c

6_c (*put down*) 7_c

8_c (*move to*) 3_c or 6_c

9_c (*move to*) 3_c or 6_c

10_c (*move to*) 3_c or 6_c

The constraints are:

2_c if swing (*anti-swing*) 8_c

5_c if swing (*anti-swing*) 8_c

2_c if cross in the same region (*cross over*) 9_c

5_c if cross in the same region (*cross over*) 9_c

2_c if move into the unloading zone behind the other crane at the same time (*stop*) 10_c

5_c if move into the loading zone behind the other crane at the same time (*cross over*) 10_c

The coordination rule base defines the desired choices and the constraints for the cranes

Figure 6.2: The Mobile Robot, ATRV-Mini.

to cooperate with each other. The proposed framework was applied to develop the control system for the multi-crane system. Experiments show that the two overhead cranes can work within the same workspace without any collisions. In the experiment, the two overhead cranes transport loads within the shared workspace.

6.2 The Control of a Multi-Agent System

A more interesting and challenging case is the application of the proposed framework for the control of the heterogeneous multi-agent system as introduced in Scenario 1.1.2. The control systems involved in this system are:

1. A mobile robot, iRobot ATRV-Mini as shown in Figure 6.2, which is a flexible, robust platform for either indoor or outdoor experiments and applications.

2. An overhead crane which has been introduced above.

3. A robot manipulator, CRS F3 as shown in Figure 6.3, which can provide six degrees of freedom.

The final goal of this application is to have a cooperative action among the overhead crane, the mobile robot and the robot manipulator. As shown in Figure 6.4, the mobile robot picks up an object in the overhead crane's workspace (zone 1) and carries it to the

Figure 6.3: The Robot Manipulator, F3.

manipulator's workspace (zone 2). The robot manipulator is mounted on a track which is an extra axis of control for the robot manipulator. The robot manipulator picks up the object from the mobile robot (zone 2) and delivers it to the other end of the track (zone 3). There are four landmarks set in the workspace to guide the mobile robot to move along the desired trajectory.

6.2.1 The Mobile Robot

In this multi-agent system, the mobile robot is a nonholonomic mobile robot with kinematic constraints in the two dimensional workspace. A nonholonomic constraint for a mobile robot is a non-integrable equation involving the configuration parameters and their derivatives (velocity parameters). Such a constraint does not reduce the dimension of the space of configurations attainable by the robot, but reduces the dimension of the space of possible differential motions at any given configuration. Consider the mobile robot that is modeled in Example 5.1.1. The moving direction of the mobile robot is the discrete value while the continuous values are listed in Equation 5.11.

The Action Executor

In order to pick up an object from the crane and deliver it to the robot manipulator,

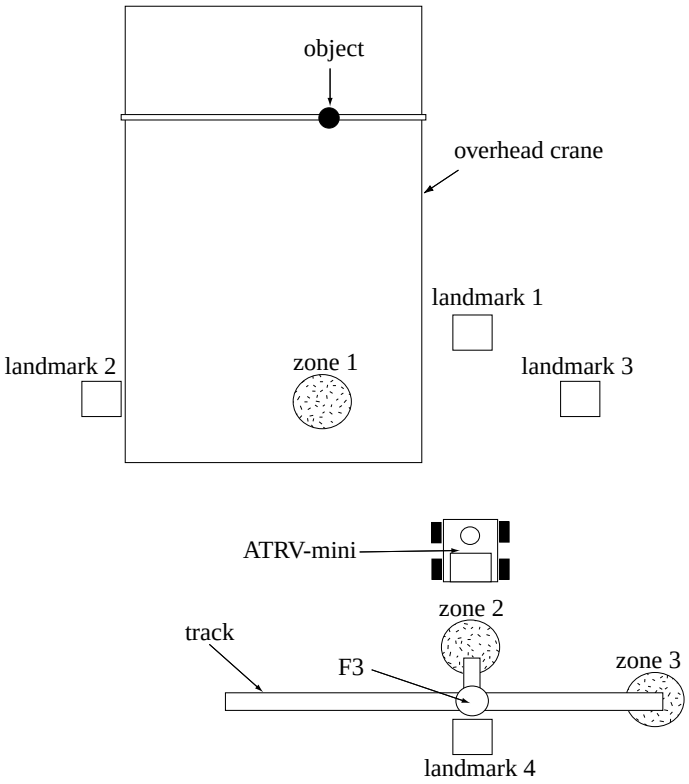

Figure 6.4: The Setup of the Multi-Agent System.

the mobile robot needs to execute the following desired actions:

(*search*) - turn the servo motor of the CCD camera to scan the environment in order to find the landmark;

(*align*) - align the robot body to the target;

(*vision navigation*) - move toward the landmark using the output of the fuzzy controller;

(*turn left*) - turn left 90^o;

(*turn right*) - turn right 90^o;

(*turn back*) - turn 180^o;

(*move back*) - move backwards into the loading area of the manipulator.

These actions are guaranteed to be executed by the hybrid action executor.

The Coordination States

For the mobile robot, we have the following coordination states:

1_m idle;

2_m first landmark located;

3_m aligned;

4_m first landmark reached;

5_m second landmark located;

6_m second landmark reached;

7_m loaded;

8_m third landmark located;

9_m third landmark reached;

10_m fourth landmark located;

11_m fourth landmark reached;

12_m ready to be unloaded.

6.2.2 The Robot Manipulator

The robot manipulator is made by CRS Robotics. The joints and tracks are illustrated in Figure 6.5. The manipulator has an interface with Windows 2000 which is called ActiveRobot. There are more than fifty functions defined in the interface for controlling the robot

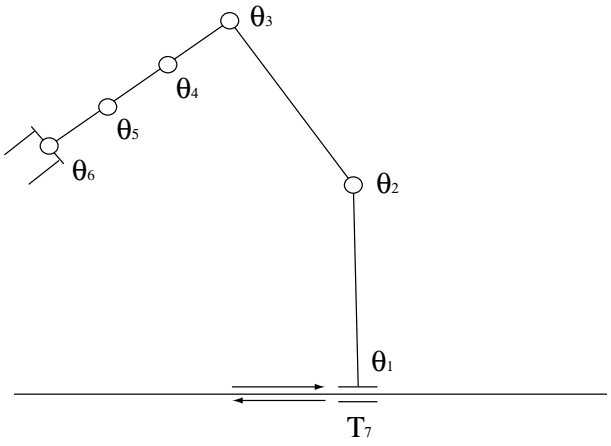

Figure 6.5: The Axis Configuration of the Robot Manipulator.

manipulator and processing data. The specification of the robot manipulator is listed in Table 6.1.

The robot is connected to the robot server that processes all the requests from the clients that command the robot manipulator. The functions that the clients can call are listed in Table 6.2 [60]. The robot server gets the control of the robot manipulator through the manipulator's controller. Then the robot server controls the joints through the functions described in the table.

The Action Executor

In order to pick up an object from the mobile robot and deliver it to the other side of the track, the robot server needs to send out the following desired actions to the robot manipulator:

(*approach*) - the tip of the manipulator approaches the object;

(*close gripper*) - the manipulator grabs the object;

(*move up*) - the tip moves up in order to pick up the object;

(*move left*) - the manipulator moves to the left end of the track;

(*turn left*) - the manipulator turns left 90^o;

(*drop*) - the gripper opens in order to drop the object;

(*turn right*) - the manipulator turns right 90^o;

Table 6.1: The Physical Limitations of the Robot Manipulator

Axis	Limit
1	$\pm 180^o$
2	-135^o to 45^o
3	$\pm 135^o$
4	$\pm 180^o$
5	$\pm 135^o$
6	51 turns or $\pm 18432^o$
Track	5000 mm

(*move right*) - the manipulator moves right and goes back to the initial position.

These actions are guaranteed to be executed by the controller.

The Coordination States

For the robot manipulator, we have the following coordination states:

1_r ready to pick up;

2_r picked up.

6.2.3 Modeling the System Using the Proposed Framework

In this section, the proposed framework is applied to model the control of the multi-agent system with the mobile robot, the robot manipulator and the overhead crane.

The Workspace

In this multi-agent system, the overhead crane, the mobile robot and the robot manipulator have three separate workspace. The mobile robot's workspace overlaps with both the crane's workspace and the robot manipulator's workspace. Thus the object could be delivered from the overhead crane to the robot manipulator.

The overhead crane's goal is the loading area. The mobile robot has short term goals as the landmarks and long term goals as the loading area in the crane's workspace and the loading area in the robot manipulator's workspace.

The Intelligent Coordination Control Layer

Table 6.2: The Functionalities of the Robot Server

Function	Input	Description
Extend	distance	Extend the arm a specific distance
Retract	distance	Retract the arm a specific distance
MoveH	distance & direction	Move the arm over the track a specific distance
MoveV	distance & direction	Move the arm UP or DOWN a specific distance
Turn	degrees & direction	Turn the base of the arm a specific angle
TurnG	degrees & direction	Turn the end effector a specific angle
TurnW	degrees & direction	Turn the wrist a specific angle
MoveTo	position	Move to a previous defined position
Learn	position	Save the current arm position in a variable
Ready	none	Move arm to the ready position
Home	none	Move arm to the home position
Gripper	position	Open/close the gripper
Speed	velocity	Set the speed

In the intelligent coordination control layer, the states of the controlled process are mapped to the coordination states. The mobile robot starts from the initial state 1_m. The coordination input $q^{\triangleleft} \in Q^{\triangleleft}$ is the coordination state $r \in R$ of the crane and the robot manipulator. We have three agents in the system, the input signature makes the mobile robot know where the coordination input is from. $x^{\triangleright} \in X^{\triangleright}$ represents the relevant position of the mobile robot in the overhead crane's workspace. The values are obtained from the sonar sensors. Based on the coordination rule base, the desired action is selected. The coordination rule base also determines the enable function g. After the desired hybrid action is executed by the action executor, the system transitions to the next coordination state. f_r outputs the continuous reference to the action executor, for example, the distance for the robot to make a turn in front of the landmarks. f_o and ϕ_o communicate the current position and discrete state to other agents, respectively.

The robot manipulator also starts from the initial state 1_r. The coordination input $q^{\triangleleft} \in Q^{\triangleleft}$ and $x^{\triangleleft} \in X^{\triangleleft}$ are the coordination state $r \in R$ of the mobile robot and the relative position of the mobile robot in the loading zone, respectively. We have three agents in the system, the input signature makes the robot manipulator know where the coordination input is from. Based on the coordination rule base, β gives the desired action. The coordination rule base also determines the enable function g. After the desired hybrid action is executed by the action executor, the system transit to the next coordination state. ϕ_o communicates the current coordination state to the other agents.

The Coordination Rule Base

Based on the nature of this multi-agent system, the following coordination rule base is defined.

The desired choices for the crane:

1_c (*pick up*) 2_c

2_c (*move to*) 3_c

3_c (*put down*) 4_c

4_c (*pick up*) 5_c

5_c (*move to*) 6_c

6_c (*put down*) 1_c

The constraints are:

If mobile robot is not at state 6_m, (*put down*) is not allowed for 3_c

The desired choices for the mobile robot:

1_m (*search*) 2_m

2_m (*align*) 3_m

3_m (*vision navigation*) 4_m

4_m (*turn left*) 5_m

5_m (*vision navigation*) 6_m

6_m (*null*) 7_m

7_m (*turn back*) 8_m

8_m (*vision navigation*) 9_m

9_m (*turn right*) 10_m

10_m (*vision navigation*) 11_m

11_m (*turn back*)(*move back*) 12_m

12_m (*null*) 1_m

The constraints are:

If crane is not at state 5_c, only (*null*) is allowed for 6_m, and the transition is prohibited

If manipulator is not at state 2_r, only (*null*) is allowed for 12_m, and the transition is prohibited

The desired choice for the robot manipulator:

2_r (*move left*)(*turn left*)(*drop*)(*turn right*)(*move right*) 1_r 1_r (*approach*)(*close gripper*)(*move up*) 2_r

The constraint for the manipulator is:

If the mobile robot is not at state 12_m, (*approach*)(*close gripper*)(*move up*) is not allowed for 1_r

Note that since there is no states defined between a series of actions for the manipulator, several actions can be executed in consequences. This series of actions can be thought as one single action.

The coordination rule base defines the desired choices and the constraints for the agents to cooperate and coordinate with each other.

Communication Mechanism

In this experiment, the server-client architecture is implemented for the communication

among agents. Communications between the server and the client are realized using the Common Object Request Broker Architecture (CORBA). Users are able to monitor and operate the agents at a remote site while the agents exchange information with others using CORBA.

In order to interface the local control to another agent at a remote side, a server and a client are constructed for the agent. The client part resides on the remote agent. The server part resides on the host machine on the local agent. The host machine is an on-line PC with Windows NT system or Linux installed. CORBA is used to handle requests involving the control of the hardware from the remote computer.

6.2.4 Results

In this section, the results of the development of this multi-agent system are given.

The Design of the Action Executor for the Mobile Robot

As an example, the design of the action executor of the mobile robot is introduced. In order to provide tolerance to the inaccuracy of the visual data, the vision navigation of the mobile robot is implemented by building a fuzzy steering controller. The turning angle of the robot is determined by differential steering method. If both the left wheels and the right wheels turn in tandem, the robot moves in a straight line. If one wheel turns faster than the other, the robot follows a curved path. So steering the robot is just a matter of varying the speed of the wheels. Since the turn radius of the robot is quite large compared with the radius of the wheels, referring to Figure 5.1, we have the following relationships:

$$\begin{aligned} S_1 &= R\phi, \\ S_m &= (R + \frac{w}{2})\phi, \\ S_2 &= (R + w)\phi, \end{aligned} \tag{6.2}$$

where S_1 and S_2 give the displacement (distance traveled) for the left and right wheel respectively, R is the turn radius of the center, w is the distance between wheels (from center-to-center along the length between the two font wheels or two back wheels), and ϕ is the angle of the turn in radians. S_m is the displacement at the center point. Once we have established the simple geometry for the differential steering system, it is easy to

develop algorithms for controlling the robot's turning angle ϕ. As the robot is considered as having a rigid body, to develop a forward kinematic equation for the differential steering system, we start by specifying a frame of reference in which an arbitrarily chosen point is treated as stationary.

By differentiating Equation 6.2, we have

$$u_R - u_L = \dot{\phi} w, \tag{6.3}$$

where u_L and u_R correspond to the forward velocity of the left wheels and the right wheels respectively. Then we have

$$u_2 = \dot{\phi} = \frac{u_R - u_L}{w}. \tag{6.4}$$

A block diagram of the fuzzy controller is shown in Figure 6.6. The desired ϕ_d is acquired by calculating the position of the target in the image representing the environment detected by the color CCD camera. The error between the command signal and the actual position, as well as the change in error of signal are calculated and fed into the fuzzy controller. From Equation 6.4, it can be seen that the difference between the velocity of the wheels determines the turning speed. The fuzzy controller is designed to output the desired u_2 to the motors to control the robot's turning angle ϕ to converge to the desired angle ϕ_d.

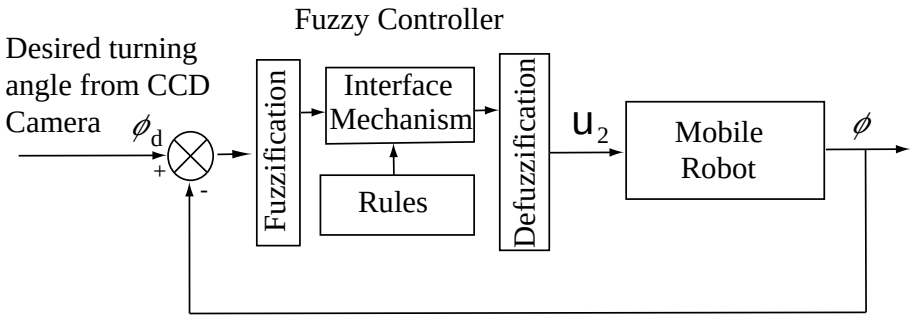

Figure 6.6: Block Diagram of the Fuzzy Logic Controller.

The fuzzification procedure maps the crisp input values to the linguistic fuzzy terms with the membership values between 0 and 1. In this study, we use five membership functions for both error $e_\phi = \phi - \phi_d$ and change in error $\dot{e}_\phi = \dot{\phi} - \dot{\phi}_d$. Figure 6.7 and Figure 6.8 illustrate the input membership functions for e_ϕ and $\dot{e}_\phi$ respectively.

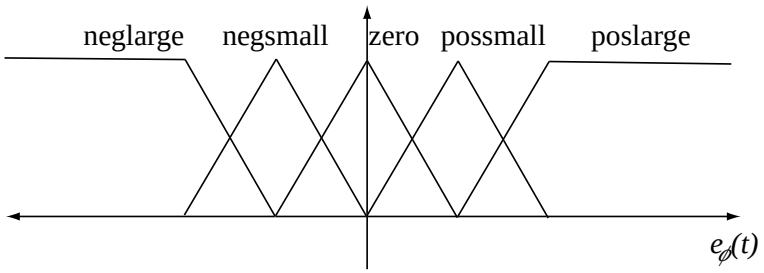

Figure 6.7: Input Membership Functions for Error e_ϕ.

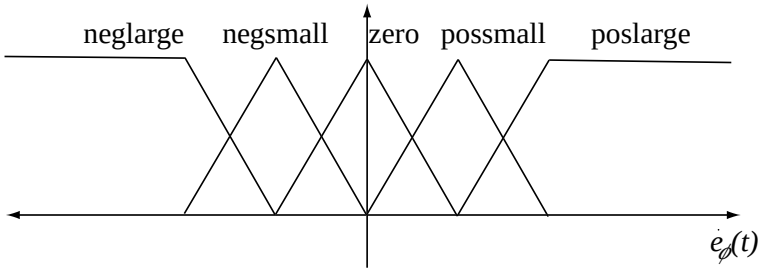

Figure 6.8: Input Membership Functions for Change in Error $\dot{e}_\phi$.

The rules governing the input output relationship of the proposed fuzzy controller are given. The inference mechanism is responsible for decision making in the control system using approximate reasoning [15]. The control rules are designed based on expert knowledge and testing. Furthermore, the control rules also meet the stability requirements derived from Lyapunov's direct method. For example, if e_ϕ is "poslarge" and is increasing rapidly (i.e., $\dot{e}_\phi$ is "poslarge"), then the left motor should be much faster than the right motor (i.e., u_2 should be "NL"). Based on this knowledge, we can obtain twenty five fuzzy rules. Table 6.3 represents abstract knowledge that the expert has about how to control the turn angle given the error and its derivative as inputs. The input and output linguistic variables are summarized in the table.

The defuzzification procedure maps the fuzzy output from the inference mechanism to a crisp signal. We use the Center Of Gravity (COG) defuzzification method for combining the recommendations represented by the implied fuzzy sets from all the rules. Let b_i denote the center of the membership function of the consequent of rule (i) and $\int \mu_{(i)} dx$ denote the area under the membership function $\mu_{(i)}$. The COG method computes the crisp value μ^c

Table 6.3: Rules for the Differential Steering System

	$u_2 - u_1$	$\dot{e}_\phi$ pos large	pos small	zero	neg small	neg large
	poslarge	NL	NL	NL	NS	ZO
	possmall	NL	NL	NS	ZO	PS
e_ϕ	zero	NL	NS	ZO	PS	PL
	negsmall	NS	ZO	PS	PL	PL
	neglarge	ZO	PS	PL	PL	PL

given by:

$$\mu^c = \frac{\sum_i b_i \int \mu_{(i)} dx}{\sum_i \int \mu_{(i)} dx}. \tag{6.5}$$

In this study it is used for computing the center of gravity of the implied fuzzy sets. Figure 6.9 shows the output membership functions.

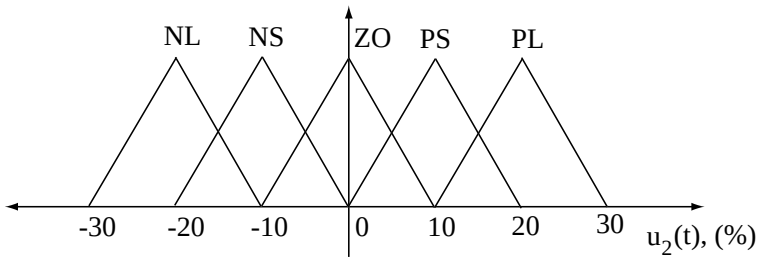

Figure 6.9: Output Membership Functions.

The fuzzy controller is developed as the action executor for the mobile robot. Figure 6.10 and Figure 6.11 show the input and the output of the fuzzy controller, respectively. The results for the mobile robot following a landmark with external disturbances are shown. For example, it is pushed to the right and then to the left. One can see that the controller is able to reduce the error to zero instantly.

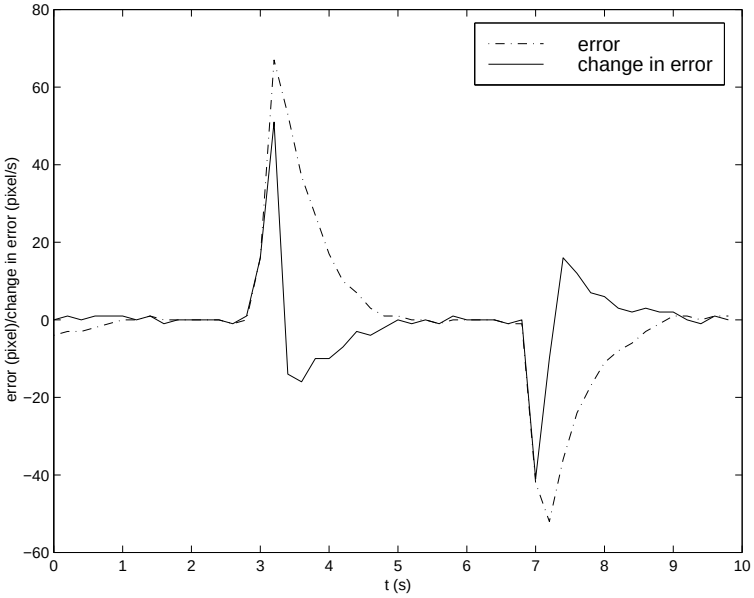

Figure 6.10: Input of the Fuzzy Controller.

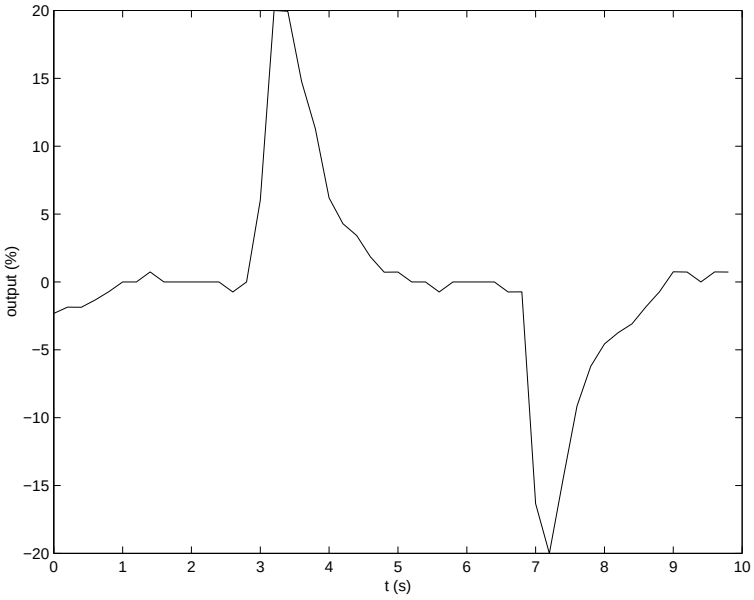

Figure 6.11: Output of the Fuzzy Controller.

Simulation and Experimental Results

In Figure 6.12, simulation results for the cooperation and coordination among the mobile robot, the overhead crane and the robot manipulator are given. In the figure, the round object represents the load of the overhead crane, while the square object represents the mobile robot. The trajectories of both the overhead crane and the mobile robot are given. From the figure, we can see that the overhead crane starts from the initial position and delivers the object to the loading area to wait for the mobile robot to pick up the object. The mobile robot follows the landmarks into the loading area and picks up the object. Then, the mobile robot turns around. For clarity, the path for the robot returning to the robot manipulator's track, which is shown as a long solid bar in the figure, is omitted. Note that even with a push, the robot can still follow the landmark and finish the desired task.

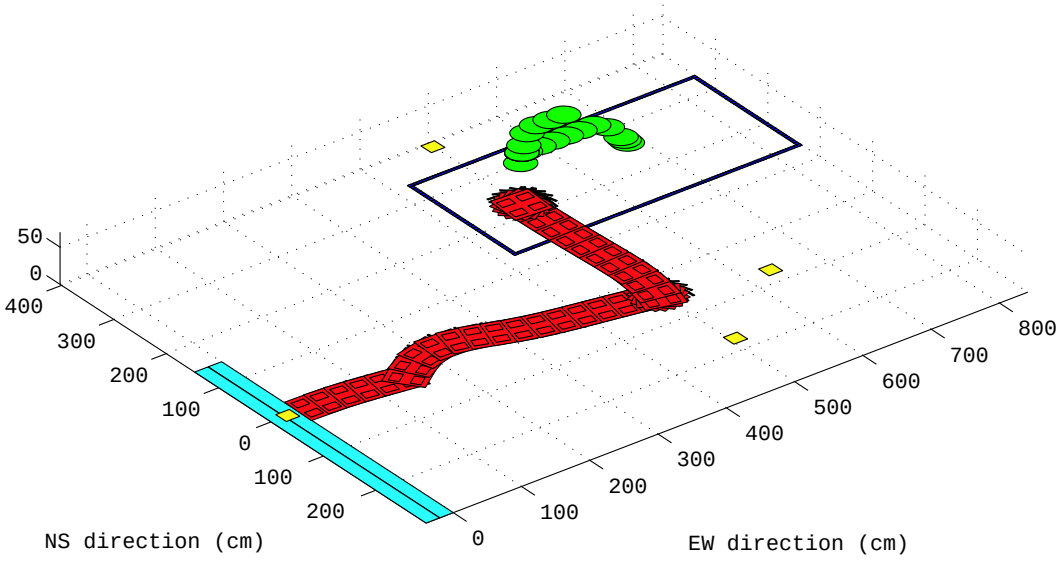

Figure 6.12: Simulation Results for the Heterogeneous Multi-Agent System.

Figure 6.13 illustrates the evolution of the continuous values of the the mobile robot, which include the position and the direction. It can be seen that the discrete state also

changes when the robot gets close to the landmarks. It makes turns at corresponding landmarks.

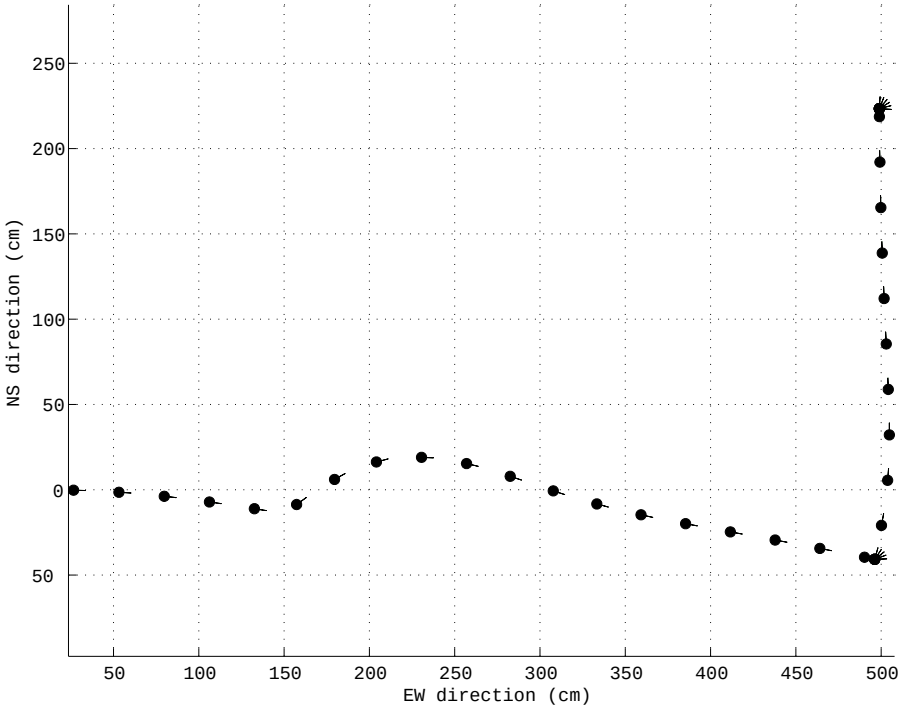

Figure 6.13: Simulation Results for the Mobile Robot.

The simulation result shows that the proposed framework can model the control of this multi-agent system. In addition, experiments also verify that the multi-agent system can achieve the desired goal successfully, for example, the overhead crane delivers an object in its workspace to the designated area, then with the vision navigation control, the mobile robot picks up the object from the crane's workspace and delivers it to the robot manipulator. The robot manipulator then picks up the object and transports it to its own workspace. The whole process involves cooperation, coordination and communication among multiple agents. By applying the proposed framework to the control of this multi-agent system, we are able to achieve coordinated control of the heterogeneous multi-agent system. The agents cooperatively work together to achieve the desired global goal.

Figure 6.14, 6.15, 6.16 and 6.17 show the four landmarks setup for the experiment. Figure 6.18 shows that the robot manipulator is able to pick up the object from the mobile robot. Figure 6.19 shows that the overhead crane is able to drop the object on top of the mobile robot. Figure 6.20 shows that the mobile robot follows the landmark. Figure 6.21 shows that the overhead crane picks up the object from its workspace. Figure 6.22 shows that the mobile robot can overcome the uneven floor when it follows the landmark.

Figure 6.14: The First Landmark.

6.3 Coordination of Multiple Mobile Robots

Since the two scenarios mentioned above do not involve extensive coordination tasks, the coordination problems can be solved by selecting desired state transitions based on the designer's own knowledge. In order to illustrate more complex coordination tasks, we apply the proposed framework to solve a multi-robot coordination problem. An intelligent planner is designed for a multi-robot planning scenario, and we use numerical simulations to illustrate this scenario.

Figure 6.15: The Second Landmark.

Figure 6.16: The Third Landmark.

Figure 6.17: The Fourth Landmark.

Scenario 6.3.1 *In this scenario, all agents are assumed to be able to finish the desired actions to move to the next state. In the simulation, there are five agents selected for the coordination problem. The environment is set to* (25×25) *grids with obstacles set in it as shown in Figure 6.23.*

The agents start from their initial positions as shown in Figure 6.23. Target positions are shown in Figure 6.24. The global goal is that all agents should try to reach their target positions without any conflicts with others. The sub-goal of each individual agent is to reach its own target without colliding with others.

6.3.1 Modeling

We assume that the agents can execute the desired actions, for example, going South, going North, going West, going East, going South East, going South West, going North East, or going North West. The problem is to develop an intelligent planner for the intelligent coordination control layer to plan the next actions for the agents. This multi-agent system is considered as a control system with multiple agents modeled with the CHA framework.

Figure 6.18: The Robot Manipulator Picks Up the Object from the Mobile Robot.

Figure 6.19: The Overhead Crane Drops the Object on Top of the Mobile Robot.

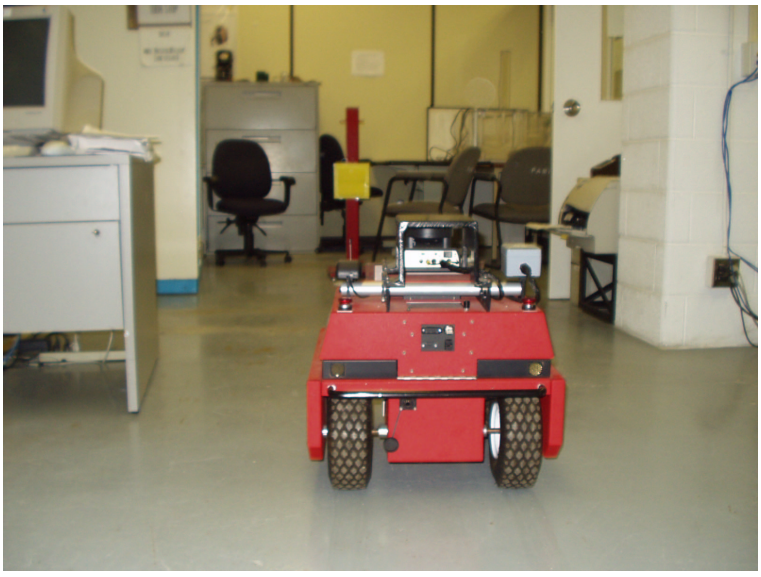

Figure 6.20: The Mobile Robot Follows the Landmark.

Figure 6.21: The Overhead Crane Picks Up the Object from the Workspace.

Figure 6.22: The Mobile Robot Follows the Landmark over the Uneven Floor.

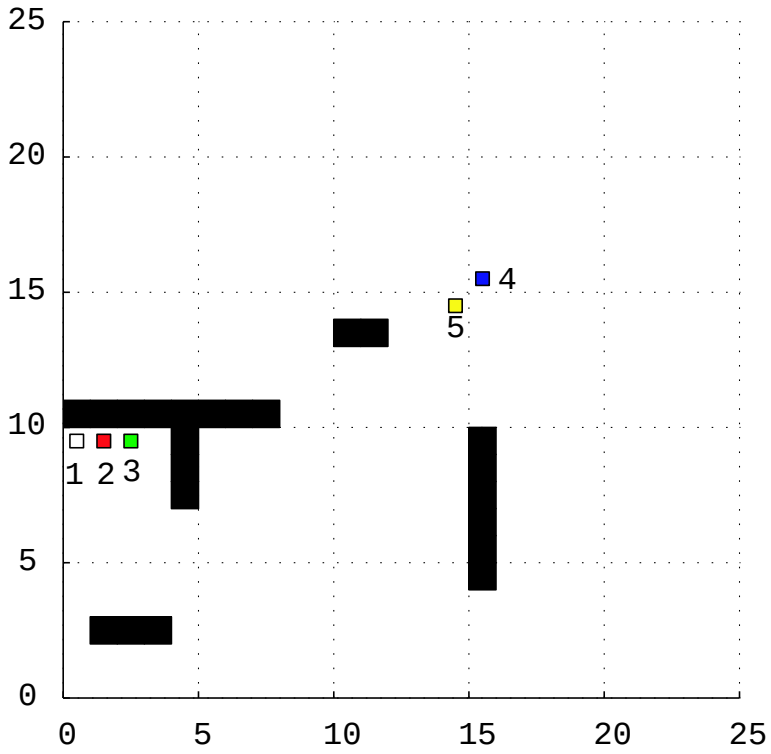

Figure 6.23: The starting positions of the agents. Large black squares represent obstacles while small squares represent the robots. White square: Agent 1; Red square: Agent 2; Green square: Agent 3; Blue square: Agent 4; Yellow square: Agent 5. The numbers are also marked beside the squares that represent the position of the corresponding agents.

Based on the objective of each agent, the intelligent planner should be able to plan the desired state trajectory that achieves the sub-goal. Action $a \in A$ for each step along the trajectory is executed and the agent moves to the next state.

A neural network based approach inspired by [68] is used for this planning problem. Only information about the goals of the multi-agent system is required for the intelligent planner to generate the coordination policies to achieve the goal. In the mean time, the constraints are defined in the coordination rule base for all the states that are not allowed which might cause collisions. For a given present state in R, denoted by r_p, the next state

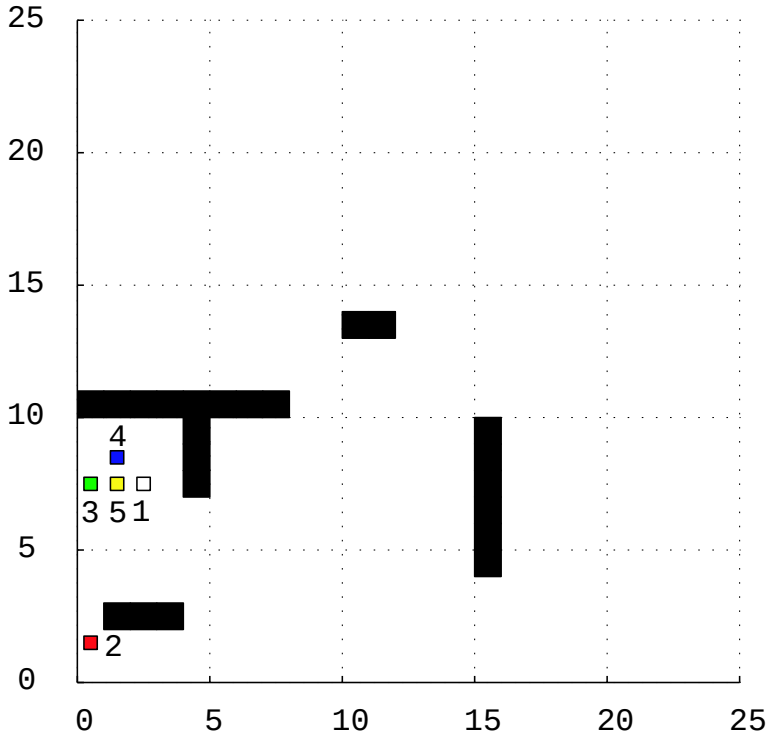

Figure 6.24: The target positions of all agents. Large black squares represent obstacles while small squares represent the robots. White square: Agent 1; Red square: Agent 2; Green square: Agent 3; Blue square: Agent 4; Yellow square: Agent 5. The numbers are also marked beside the squares that represent the position of the corresponding agents.

r_n is obtained by $r_n \Leftarrow x_{r_n} = \max\{x_i, i = 1, 2, \ldots, k\}$ where x is the degree of landscape activity of the neural network, i is the number of the neighboring states.

6.3.2 Intelligent Planning

In [68], a biologically inspired neural network approach to collision-free motion planning of mobile robots or robot manipulators is proposed. Each neuron in the topologically organized neural network has only local connections, whose neural dynamics are characterized by a shunting equation. The robot motion is planned through the dynamic activity

landscape of the neural network without any prior knowledge of the dynamic environment.

Inspired by this approach, we propose an intelligent planner for the multi-robot coordination scenario. We implement a topologically organized neural network which is expressed in a state space $\mathcal{S}$. The location of the jth neuron of the ith agent at the grid in $\mathcal{S}$, denoted by a vector $q_{ij} \in R^F$, uniquely represents a state of an agent in $\mathcal{S}$. Each neuron has local lateral connections to its neighboring neurons that constitute a subset in $\mathcal{S}$, which is called the receptive field of the jth neuron of the ith agent in neuro-physiology. The proposed dynamics of the jth neuron of the ith agent is characterized by a modified shunting equation,

$$\begin{aligned} \frac{dx_{ij}}{dt} &= -Ax_{ij} + (B - x_{ij}) \\ &\quad ([I_{ij} + \beta * Ico_{ij}^{old}]^+ + \Sigma_{k=1}^{m} w_{ijk}[x_{ik}]^+) \\ &\quad -(D + x_{ij})([I_{ij} + Ico_{ij}]^-). \end{aligned} \tag{6.6}$$

Parameters A, B and D represent the passive decay rate, the upper and lower bounds of the neural activity, respectively. Variable x_{ij} is the neural activity of the jth neuron of the ith agent, which has a bounded continuous value $x_{ij} \in [-D, B]$. t is a virtual time index that only depends on the occurrence of events. The excitatory input, $[I_{ij} + \beta \times Ico_{ij}^{old}]^+ + \Sigma_{k=1}^{m} w_{ijk}[x_{ik}]^+$, results from the target, the coordination factors determined by the states of other agents, and the lateral connections among neurons. The external input I_{ij} to the jth neuron of the ith agent is defined as $I_{ij} = E$, if there is a target; $I_{ij} = -E$, if there is an obstacle; $I_{ij} = 0$, otherwise, where E is a positive constant. β is a coordination recovery rate to adjust the recovery speed of the neural network to the inhibitory stimulus of the conflict states caused by other agents. Ico_{ij}^{old} is the stimulus of the previous conflict state. The inhibitory input $[I_{ij} + Ico_{ij}]^-$ results from the obstacles and the conflict states caused by other agents while Ico_{ij} is a coordination term determined by a coordination coefficient α. Ico_{ij} is defined as $Ico_{ij} = -\alpha \times E$, if there is another agent at that state; $Ico_{ij} = 0$, otherwise. The connection weight w_{ijk} from the kth neuron to the jth neuron of the ith agent is defined as $w_{ijk} = f(|d_{ijk}|)$, where $d_{ijk} = |q_{ij} - q_{ik}|$ represents the Euclidean distance between q_{ij} and q_{ik} in $\mathcal{S}$. Function $f(a)$ is a monotonically decreasing function, for example, a function defined as $f(a) = \mu/a$, if $0 < a < r_0$; $f(a) = 0$, otherwise, where μ and r_0 are positive constants. The neuron has only local connections in a small region

$(0, r_0)$, which is called its receptive field $\mathcal{R}_{ij}$. Parameter m is the number of neurons located within $\mathcal{R}_{ij}$.

For a given present state in $\mathcal{S}$, denoted by q_p, the next state q_n is obtained by

$$q_n \Leftarrow \Big(x_{q_n} = \max\{x_i, i = 1, 2, \ldots, k\}\Big), \tag{6.7}$$

where i is the number of the neighboring neurons including itself (i.e., all the possible next locations). The present location adaptively changes according to the varying environment.

6.3.3 Results

The neural network based intelligent planner has 25×25 topologically organized neurons with zero initial activity. The model parameters are chosen as $A = B = D = 1$, $\mu = 0.02$, $r_0 = 2$, $E = 1$, $\alpha = 0.02$ and $\beta = 0.85$.

The landscape activities of the five agents are shown in Figure 6.25. The intelligent planners are triggered by the completion of the previous task as defined as a discrete event. The collision-free trajectories of the five agents generated by the intelligent planners are shown in Figure 6.26. It can be seen that the five intelligent planners are able to plan the state trajectories dynamically that lead to the target positions.

6.4 Summary

This chapter gives some experimental and simulation results for systems modeled using the proposed framework. The feasibility of the proposed framework is illustrated by three different scenarios. The control schemes are developed using the proposed framework. It is demonstrated that the proposed framework is generic and is applicable to both homogeneous and heterogeneous multi-agent systems.

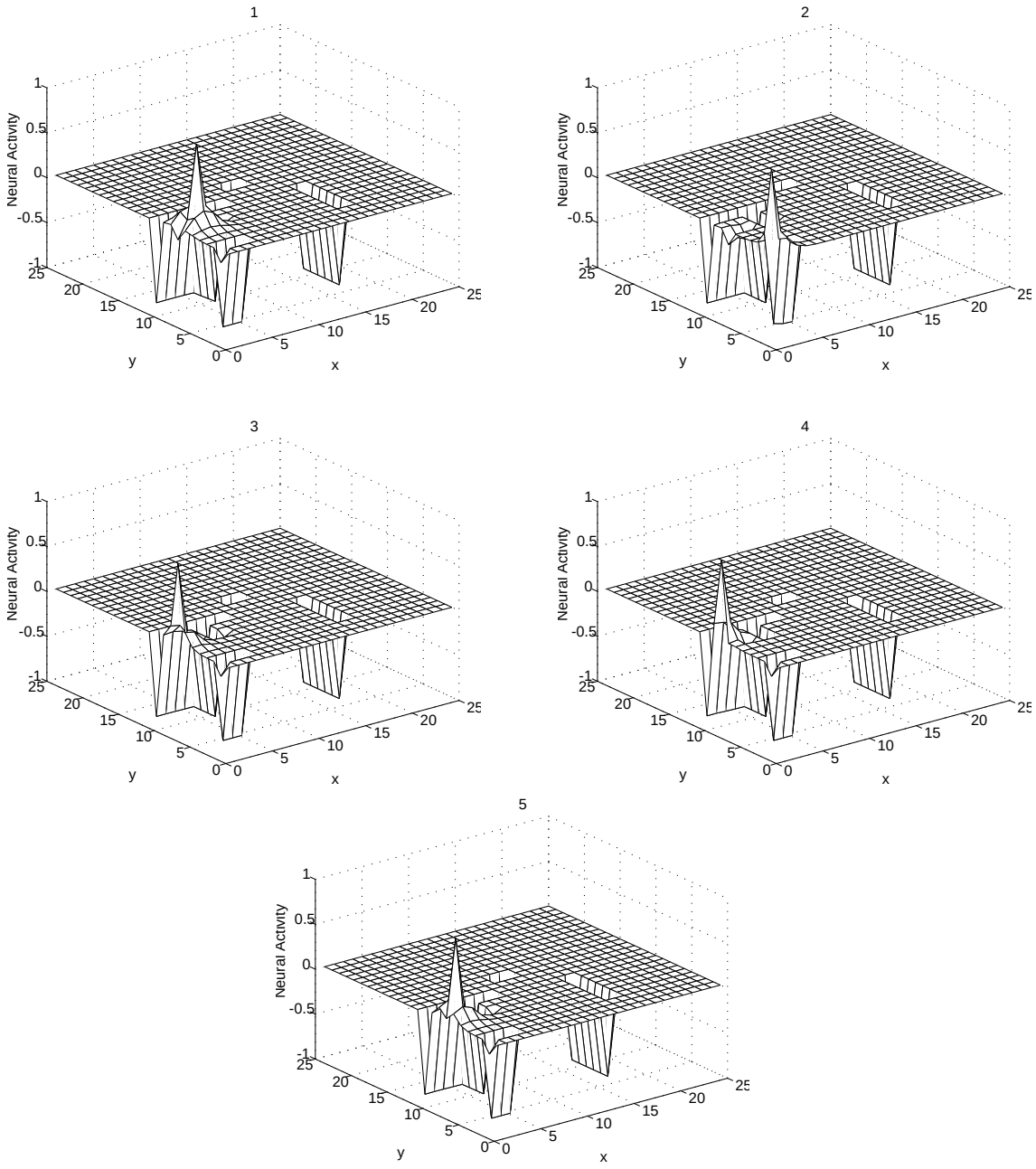

Figure 6.25: Landscape of neural activities (at the end of the simulation). 1. Neural activity of Agent 1; 2. Neural activity of Agent 2; 3. Neural activity of Agent 3; 4. Neural activity of Agent 4; 5. Neural activity of Agent 5.

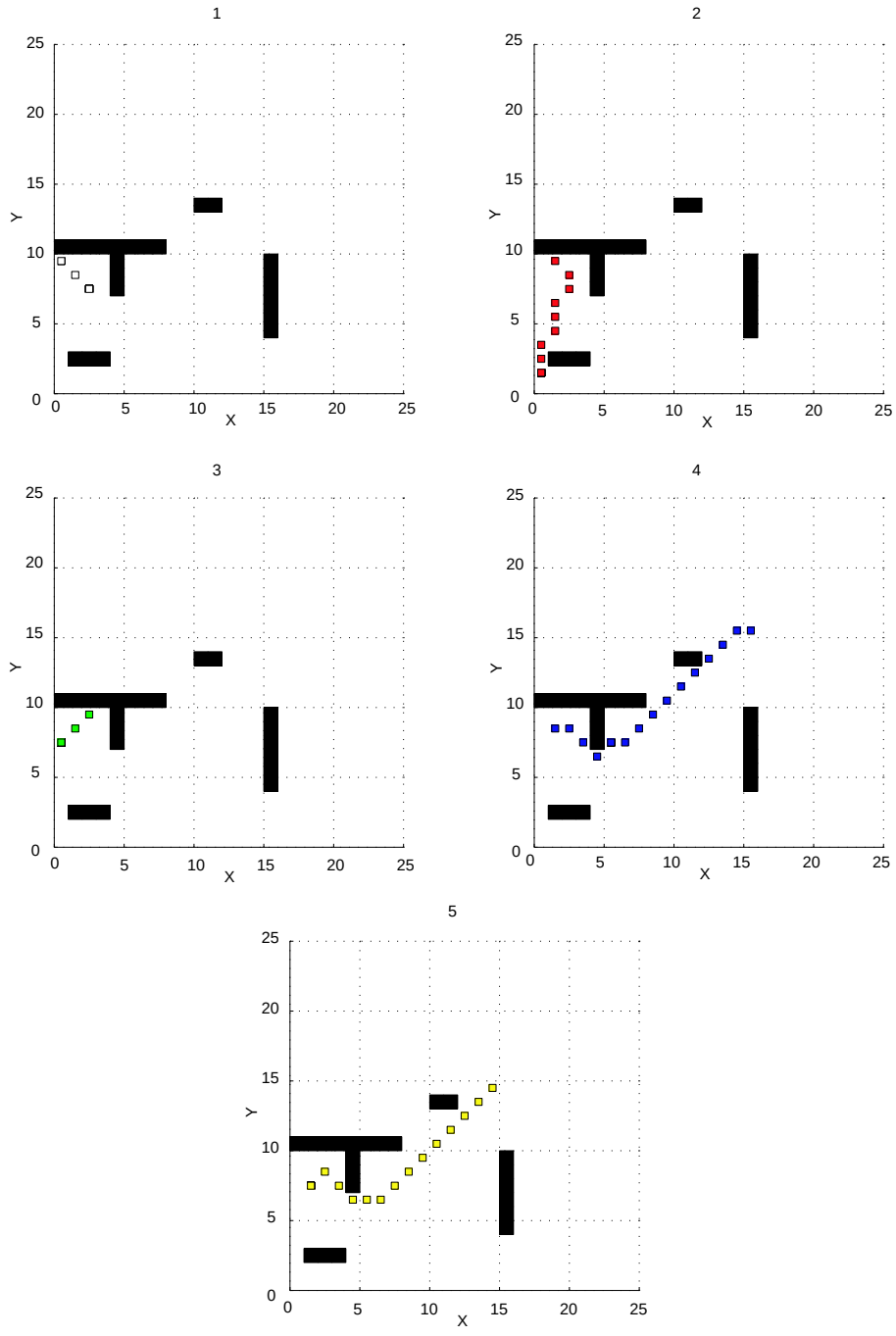

Figure 6.26: Trajectories of the Agents. Large black squares represent obstacles while small squares represent the robots. 1. Trajectory of Agent 1; 2. Trajectory of Agent 2; 3. Trajectory of Agent 3; 4. Trajectory of Agent 4; 5. Trajectory of Agent 5.

Chapter 7

Stability Analysis and Optimization Examples

This chapter illustrates stability and optimization analysis for MASs using the methodology we have proposed and developed in previous chapters. In Chapter 6, we have demonstrated that we can implement the proposed framework to develop the control algorithm for both homogeneous and heterogeneous multi-agent systems. The feasibility of the proposed framework is illustrated through three different scenarios. In this chapter, the stability of the homogeneous and the heterogeneous MASs is analyzed. In addition, we also apply the direct identification algorithm we have proposed to optimize the performance of the heterogeneous MAS.

7.1 Stability Analysis of CHA MASs

In this section, the stability of the homogeneous system introduced in Scenario 1.1.1 and the stability of the heterogeneous system introduced in Scenario 1.1.2 are analyzed.

7.1.1 Stability of the Homogeneous System in Scenario 1.1.1

Proposition 7.1.1 *Based on the definition of the stability of CHA systems, the multi-crane system in Scenario 1.1.1 is globally stable.*

Proof 1. Because of the design of the hardware and the software of the overhead cranes, the action executors can accomplish the hybrid actions. After the actions have been executed, the coordination state transitions to the next coordination state according to the coordination rule base. Each overhead crane agent is locally stable because:

- The action executor is designed such that the continuous state evolution is stable (i.e., the desired position of the payload can be reached by the overhead crane, thus the equilibrium point of each abstract state is maintained).
- The abstract state transition is achievable by the proper design of the action executor, and the reverse state transition does not happen immediately.
- All unsafe regions in the continuous space are avoided with the proper design of the action executor.

2. As described in Chapter 6 and also in [33], all the actions taken by the overhead cranes are on the allowed event trajectories $\mathcal{E}_a$ which is governed by the coordination rule base. For $r \in R$, $a \in A$, $c \in C$, we have $r \models l_o$ holds and $(a, l_o) \in C$, $r \models l_c$ holds and $(a, l_c) \in C$ respectively. Recall that for the overhead crane, coordination states 1 and 7 represent state 'idle' and state 'put down without load' respectively. The goal set is the region around state (7, 7) for (crane 1, crane 2) and the origin set corresponds to the coordination state (1, 1). $\mathcal{E}_a$ leads the system to the goal set.

3. We wish to show that for this multi-crane system, the goal set, the invariant set $R_m \subset R$ is stable in the sense of Lyapunov w.r.t. $\mathcal{E}_a$.

   We use the metric defined by the Euclidean distance between each agent (at the beginning of each abstract state) and the goal region along the allowed event trajectories $\mathcal{E}_a$, which is

$$\rho(r, R_m) = \Sigma_{i=1}^{2}\{|x_i(r_i) - \bar{x}_i| + |y_i(r_i) - \bar{y}_i| + |z_i(r_i) - \bar{z}_i|\} \tag{7.1}$$

   in which the goal region is defined as $R_m = \{(7, 7)\}$ which corresponds to $\{(\bar{x}_1, \bar{y}_1, \bar{z}_1), (\bar{x}_2, \bar{y}_2, \bar{z}_2)$. Subscript 1 is used to represent overhead crane 1 and 2 represents overhead crane 2. We choose

$$V(r) = \rho(r, R_m), \tag{7.2}$$

then we need to show that in a sufficiently small r-neighborhood of the set R_m the Lyapunov function V has the required properties.

(1) If we choose $c_2 = c_1$, it is obvious that for all sufficiently small $c_1 > 0$, when $V(r) > c_2$ for $r \in$ r-neighborhood of R_m, $\rho(r, R_m) > c_1$.

(2) Same as above, if we choose $c_3 = c_4 > 0$ as small as desired, when $\rho(r, R_m) < c_3$ for $r \in$ r-neighborhood of R_m, we have $V(r) \leq c_4$.

(3) By design, all the agents only move toward the next goal along the allowed event trajectories $\mathcal{E}_a$, they don't go backward. So we have $V(\mathcal{R}(r_0, \mathcal{E}_k, k))$ a non-increasing function for $k \in \mathcal{Z}^+$, as long as $\mathcal{R}(r_0, \mathcal{E}_k, k) \in$ r-neighborhood for all $\mathcal{E}_k$ such that $\mathcal{E}_k\mathcal{E} \in \mathcal{E}_a(r_0)$. ∎

Proposition 7.1.2 *Based on the definition of the stability of CHA systems, the multi-crane system in Scenario 1.1.1 is asymptotically stable.*

Proof We have proved that the multi-crane system is globally stable. In order to prove that the system is asymptotically stable, we need to show that the goal set, the invariant set $R_m \subset R$ is asymptotically stable in the sense of Lyapunov w.r.t. $\mathcal{E}_a$.

We have already shown that for the closed invariant set $R_m \subset R$, in a sufficiently small r-neighborhood of the set R_m there exists a function $V(r) = \rho(r, R_m)$ having all the properties of Theorem 4.2.1.

Furthermore, overhead crane 1 will move to the target location at $(\bar{x}_1, \bar{y}_1, \bar{z}_1)$; overhead crane 2 will move to the target location at $(\bar{x}_2, \bar{y}_2, \bar{z}_2)$. Thus at state $(7, 7)$, we have

$$\begin{aligned} \rho(r, R_m) &= \Sigma_{i=1}^{2}\{|x_i(r_i) - \bar{x}_i| + |y_i(r_i) - \bar{y}_i| \\ &\quad +|z_i(r_i) - \bar{z}_i|\} \\ &= 0. \end{aligned}$$

Recall that when the system terminates, we can append null actions at the end of the event trajectory $\mathcal{E}_a$. Therefore as $k \to \infty$, we have $V(\mathcal{R}(r_0, \mathcal{E}_k, k)) = \rho(r, R_m) \to 0$ for all $\mathcal{E}_k$ such that $\mathcal{E}_k\mathcal{E} \in \mathcal{E}_a(r_0)$ and for all $k \in \mathcal{Z}^+$ as long as $\mathcal{R}(r_0, \mathcal{E}_k, k) \in$ the r-neighborhood of the set R_m. ∎

Proposition 7.1.3 *Based on the definition of the stability of CHA systems, the multi-crane system in Scenario 1.1.1 is exponentially stable.*

Proof We have proved that the multi-crane system is globally stable. In order to prove that the system is exponentially stable, we need to show that the goal set, the invariant set $R_m \subset R$ is exponentially stable in the sense of Lyapunov w.r.t. $\mathcal{E}_a$.

We use the metric defined by the Euclidean distance between each agent (at the beginning of each abstract state) and the goal region along the allowed event trajectories $\mathcal{E}_a$, which is

$$\rho(r, R_m) = \Sigma_{i=1}^{2}\{|x_i(r_i) - \bar{x}_i| + |y_i(r_i) - \bar{y}_i| + |z_i(r_i) - \bar{z}_i|\} \tag{7.3}$$

in which the goal region is defined as $R_m = \{(7, 7)\}$ which corresponds to $\{(\bar{x}_1, \bar{y}_1, \bar{z}_1)$, $(\bar{x}_2, \bar{y}_2, \bar{z}_2)$. Subscript 1 is used to represent overhead crane 1, and 2 represents overhead crane 2. We choose

$$V(r) = \rho(r, R_m). \tag{7.4}$$

If we choose $c_1 = 1$ and $c_2 = 1$, it can be seen that $c_1\rho(r, R_m) = V(r) = c_2\rho(r, R_m)$, which satisfy the first condition of Theorem 4.2.3.

For the second condition of Theorem 4.2.3, assume that at the goal state, each agent has taken N_i actions. If we choose $c_3 < \frac{\max_{k=1}^{N_{i-1}} |\rho(\mathcal{R}(r_0,\mathcal{E}_{k+1},k+1)) - \rho(\mathcal{R}(r_0,\mathcal{E}_k,k))|}{\rho(r_0,R_m)}$, it can be seen that the second condition of Theorem 4.2.3, $V(\mathcal{R}(r_0, \mathcal{E}_{k+1}, k + 1)) - V(\mathcal{R}(r_0, \mathcal{E}_k, k)) \leq -c_3(\rho(\mathcal{R}(r_0, \mathcal{E}_k, k), R_m))$, is also satisfied.

Therefore, the invariant set R_m is exponentially stable w.r.t $\mathcal{E}_a$. ∎

7.1.2 Stability of the Heterogeneous System in Scenario 1.1.2

Proposition 7.1.4 *The action executor for* vision navigation *of the mobile robot in Scenario 1.1.2 can complete the task, and the continuous state is stable w.r.t. the target in the sense of Lyapunov.*

Proof Recall the model of the mobile robot shown in Figure 5.1. Since the actuators in the mobile robot we are discussing are DC motors. Based on Newton's law, we have

$$J\ddot{\theta}_m + b\dot{\theta}_m = \tau, \tag{7.5}$$

where θ_m is the displacement of the rotor ($radius$), J is the moment of inertia of the rotor ($kg \cdot m^2$), b is the damping ratio of the mechanical system ($N \cdot m \cdot s$) and τ is the motor torque ($N \cdot m$). Assume the friction on both the front wheels are same, then

$$\begin{aligned} J\frac{\dot{u}_R}{r} + b\frac{u_R}{r} &= \tau_R - f, \\ J\frac{\dot{u}_L}{r} + b\frac{u_L}{r} &= \tau_L - f, \end{aligned} \tag{7.6}$$

where r is the radius of the wheels, f is the friction torque, and u is the velocity. Hence

$$\tau_R - \tau_L = \frac{J}{r}(\dot{u}_R - \dot{u}_L) + \frac{b}{r}(u_R - u_L). \tag{7.7}$$

Put Equation 6.3 into Equation 7.7, we have

$$\tau_R - \tau_L \ = \ \frac{J}{r}\ddot{\phi}w + \frac{b}{r}\dot{\phi}w. \tag{7.8}$$

Rearrange Equation 7.8, we obtain

$$\ddot{\phi} = \frac{r}{Jw}(\tau_R - \tau_L) - \frac{b}{J}\dot{\phi}, \tag{7.9}$$

where $\tau_R - \tau_L$ should be given by the fuzzy controller and

$$\tau_R - \tau_L = \Phi(x_1,\ x_2). \tag{7.10}$$

Since we are interested in the control of the turn angle about the desired turn angle ϕ_d, we need to maintain the equilibrium by decreasing the error between ϕ_d and the actual turn angle ϕ and the change in error to zero. Let

$$\begin{aligned} e_\phi &= \phi - \phi_d = x_1, \\ \dot{e}_\phi &= \dot{\phi} - \dot{\phi}_d = x_2, \end{aligned} \tag{7.11}$$

then we have

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= \frac{r\Phi(x_1,\ x_2)}{Jw} - \ddot{\phi}_d - \frac{b}{J}(x_2 + \dot{\phi}_d). \end{aligned} \tag{7.12}$$

Assume that at the equilibrium point $\dot{\phi}_d = 0$ and $\ddot{\phi}_d = 0$. Note that the equilibrium point corresponds to $\tau_R = \tau_L$ (i.e., for the fuzzy controller $\Phi(0,\ 0) = \tau_R - \tau_L = 0$) so that the equilibrium is preserved. Choose

$$V(x) = \frac{1}{2} k\, x_1^2 + \frac{1}{2} x_2^2 \tag{7.13}$$

as the "Lyapunov function" $V : B(h) \to \mathcal{R}$ for some $h > 0$, where $B(h) = \{x \in \mathcal{R}^2 : |x| < h\}$ is a ball centered at the origin with a radius h and $|\cdot|$ is a norm on $\mathcal{R}^2$. k is a positive constant whose value can be changed to guarantee the stability, then

$$\nabla V\big(x(t)\big) = [k\, x_1,\ x_2]^T, \tag{7.14}$$

and

$$\dot{V} = [k\, x_1,\ x_2] \begin{bmatrix} x_2 \\ \frac{r\Phi(x_1,\ x_2)}{Jw} - \frac{b}{J} x_2 \end{bmatrix}. \tag{7.15}$$

We would like $\dot{V} < 0$ to prove asymptotic stability (i.e., to show that the fuzzy controller can achieve and maintain the desired turn angle ϕ_d). Hence we need

$$x_2\Big(k\, x_1 + \frac{r\Phi(x_1,\ x_2)}{Jw} - \frac{b}{J} x_2) < 0. \tag{7.16}$$

Note that $x_2 \neq 0$, then we need

$$\begin{aligned} k\, x_1 + \frac{r\Phi(x_1,\ x_2)}{Jw} - \frac{b}{J} x_2 > 0 \quad (x_2 < 0), \\ k\, x_1 + \frac{r\Phi(x_1,\ x_2)}{Jw} - \frac{b}{J} x_2 < 0 \quad (x_2 > 0). \end{aligned} \tag{7.17}$$

Rearranging the equations we see that we need

$$\begin{aligned} \Phi(x_1,\ x_2) > \frac{Jw}{r}(\frac{b}{J} x_2 - k\, x_1) \ \ (x_2 < 0), \\ \Phi(x_1,\ x_2) < \frac{Jw}{r}(\frac{b}{J} x_2 - k\, x_1) \ \ (x_2 > 0) \end{aligned} \tag{7.18}$$

on $x \in B(h)$ for some $h > 0$. As a graphical approach, we can plot the right-hand side of this equation, design the fuzzy controller $\Phi(x_1,\ x_2)$, and find $h > 0$ so that the given

inequality holds and hence asymptotic stability holds. From Figure 7.1, it can be seen that the plot for the right-hand side of Equation 7.18 is a curved plane passing the origin. This can also be proved by setting $x_1 = 0$ and $x_2 = 0$ in Equation 7.18. Note that we can always scale the Φ coordinate by choosing different value of the positive constant k. Recall the definition of the rules in Table 6.3, it can be seen that the proposed fuzzy controller $\Phi(x_1,\ x_2)$ is below the curved plane in Figure 7.1 when $x_2 > 0$ (corresponds to change in error $\dot{e}\ > 0$) and is above the curved plane when $x_2 < 0$ (corresponds to change in error $\dot{e}\ < 0$) so that the stability holds. We should also note that Figure 7.1 only shows the situation when $\phi_d = 0$. If we set ϕ_d to other values rather than 0, it can be seen that for different ϕ_d, the stability will always hold. ∎

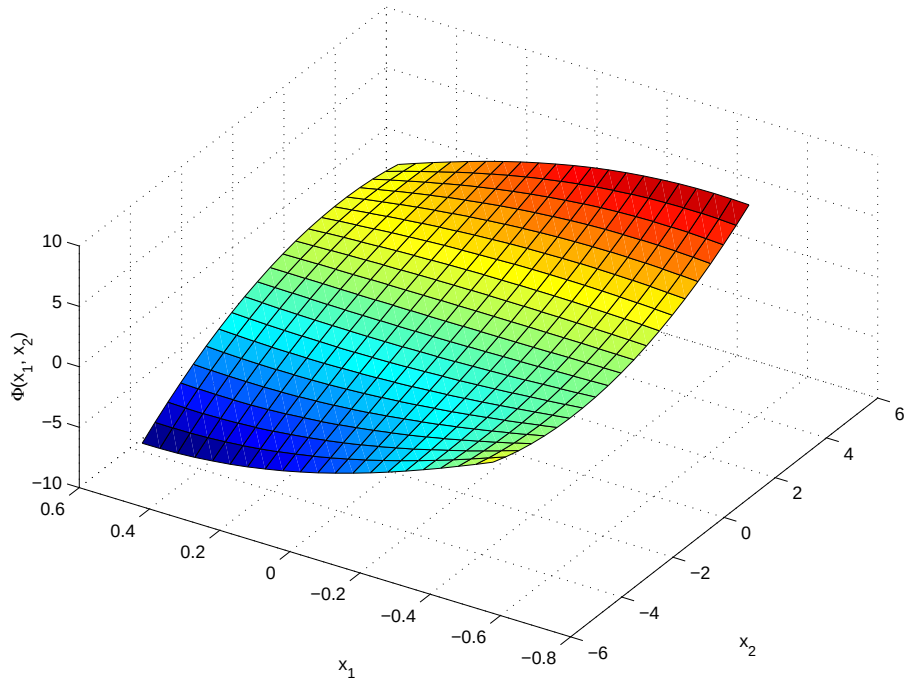

Figure 7.1: Plot of the Right-Hand Side of Equation 7.18.

Proposition 7.1.5 *Based on the definition of the stability of CHA systems, the multi-agent system with the mobile robot, the robot manipulator and the overhead crane in Sce-*

nario 1.1.2 is stable.

Proof 1. Because of the design of the hardware and the software of the overhead crane, the mobile robot and the robot manipulator, the action executors can accomplish the hybrid actions. After the actions have been executed, the coordination state transitions to the next coordination state according to the coordination rule base. The overhead crane agent is locally stable as shown above. The robot manipulator agent is locally stable because:

- The action executor is designed such that the continuous state evolution is stable (i.e., the target position of the end effector can be reached by the robot manipulator, thus the equilibrium point of each abstract state is maintained).
- The abstract state transition is achievable by the proper design of the action executor, and the reverse state transition does not happen immediately.
- All unsafe regions in the continuous space are avoided with the proper design of the action executor.

The mobile robot agent is locally stable because:

- The action executor is designed such that the continuous state evolution is stable (For the mobile robot, the vision navigation equilibrium point of each abstract state is maintained as stated in Proposition 7.1.4).
- The abstract state transition is achievable by the proper design of the action executor, and the reverse state transition does not happen immediately.
- All unsafe regions in the continuous space are avoided with the proper design of the action executor.

2. As described in Chapter 6, all the actions taken by the overhead crane, the mobile robot and the robot manipulator are on the allowed event trajectories $\mathcal{E}_a$ which is governed by the coordination rule base. For $r \in R$, $a \in A$, $c \in C$, we have $r \models l_o$ holds and $(a, l_o) \in C$, $r \models l_c$ holds and $(a, l_c) \in C$ respectively. Recall that for the overhead crane, coordination states 1 and 7 represent state 'idle' and state 'put down without load' respectively. For the mobile robot, coordination states 1 and 12 represent 'idle'

and 'ready to be unloaded' respectively. For the robot manipulator, coordination state 1 represents 'ready to pick up'. The goal set is the region around state (7, 12, 1) for (crane, mobile robot, robot manipulator) and the origin set corresponds to the coordination state (1, 1, 1). $\mathcal{E}_a$ leads the system to the goal set.

3. We wish to show that for this multi-agent system, the goal set, the invariant set $R_m \subset R$ is stable in the sense of Lyapunov w.r.t. $\mathcal{E}_a$.

   We use the metric defined by the Euclidean distance between each agent (at the beginning of each abstract state) and the goal region along the allowed event trajectories $\mathcal{E}_a$, which is

   $$\rho(r, R_m) = \Sigma_{i=1}^{3}\{|x_i(r_i) - \bar{x}_i| + |y_i(r_i) - \bar{y}_i| + |z_i(r_i) - \bar{z}_i|\} \tag{7.19}$$

   in which the goal region is defined as $R_m = \{(7, 12, 1)\}$ which corresponds to $\{(\bar{x}_1, \bar{y}_1, \bar{z}_1)$, $(\bar{x}_2, \bar{y}_2, \bar{z}_2)$, $(\bar{x}_3, \bar{y}_3, \bar{z}_3)\}$. Subscript 1 is used to represent the overhead crane, 2 represents the mobile robot, and 3 represents the robot manipulator. Note that for the mobile robot $z_2 = \bar{z}_2$. We choose

   $$V(r) = \rho(r, R_m), \tag{7.20}$$

   then we need to show that in a sufficiently small r-neighborhood of the set R_m the Lyapunov function V has the required properties.

   (1) If we choose $c_2 = c_1$, it is obvious that for all sufficiently small $c_1 > 0$, when $V(r) > c_2$ for $r \in$ r-neighborhood of R_m, $\rho(r, R_m) > c_1$.

   (2) Same as above, if we choose $c_3 = c_4 > 0$ as small as desired, when $\rho(r, R_m) < c_3$ for $r \in$ r-neighborhood of R_m, we have $V(r) \leq c_4$.

   (3) By design, all the agents only move toward the next goal along the allowed event trajectories $\mathcal{E}_a$, they don't go backward. So we have $V(\mathcal{R}(r_0, \mathcal{E}_k, k))$ a non-increasing function for $k \in \mathcal{Z}^+$, as long as $\mathcal{R}(r_0, \mathcal{E}_k, k) \in$ r-neighborhood for all $\mathcal{E}_k$ such that $\mathcal{E}_k\mathcal{E} \in \mathcal{E}_a(r_0)$. ∎

Proposition 7.1.6 *Based on the definition of the stability of CHA systems, the multi-agent system with the mobile robot, the robot manipulator and the overhead crane in Scenario 1.1.2 is asymptotically stable.*

Proof We have proved that the multi-agent system with the mobile robot, the robot manipulator and the overhead crane is stable. In order to prove that the system is asymptotically stable, we need to show that the goal set, the invariant set $R_m \subset R$ is asymptotically stable in the sense of Lyapunov w.r.t. $\mathcal{E}_a$.

We have already shown that for the closed invariant set $R_m \subset R$, in a sufficiently small r-neighborhood of the set R_m there exists a function $V(r) = \rho(r, R_m)$ having all the properties of Theorem 4.2.1.

Furthermore, the overhead crane will move to the target location at $(\bar{x}_1, \bar{y}_1, \bar{z}_1)$; the mobile robot will move to the target location at $(\bar{x}_2, \bar{y}_2, \bar{z}_2)$; the robot manipulator will move to the target location at $(\bar{x}_3, \bar{y}_3, \bar{z}_3)$. Thus at state $(7_c, 12_m, 1_r)$, we have

$$\begin{aligned} \rho(r, R_m) &= \Sigma_{i=1}^{3}\{|x_i(r_i) - \bar{x}_i| + |y_i(r_i) - \bar{y}_i| \\ &\quad +|z_i(r_i) - \bar{z}_i|\} \\ &= 0. \end{aligned}$$

Recall that when the system terminates, we can append null actions at the end of the event trajectory $\mathcal{E}_a$. Therefore as $k \to \infty$, we have $V(\mathcal{R}(r_0, \mathcal{E}_k, k)) = \rho(r, R_m) \to 0$ for all $\mathcal{E}_k$ such that $\mathcal{E}_k\mathcal{E} \in \mathcal{E}_a(r_0)$ and for all $k \in \mathcal{Z}^+$ as long as $\mathcal{R}(r_0, \mathcal{E}_k, k) \in$ the r-neighborhood of the set R_m. ∎

Proposition 7.1.7 *Based on the definition of the stability of CHA systems, the multi-agent system with the mobile robot, the robot manipulator and the overhead crane in Scenario 1.1.2 is exponentially stable.*

Proof We have proved that the multi-agent system with the mobile robot, the robot manipulator and the overhead crane is stable. In order to prove that the system is exponentially stable, we need to show that the goal set, the invariant set $R_m \subset R$ is exponentially stable in the sense of Lyapunov w.r.t. $\mathcal{E}_a$.

We use the metric defined by the Euclidean distance between each agent (at the beginning of each abstract state) and the goal region along the allowed event trajectories $\mathcal{E}_a$, which is

$$\rho(r, R_m) = \Sigma_{i=1}^{3}\{|x_i(r_i) - \bar{x}_i| + |y_i(r_i) - \bar{y}_i| + |z_i(r_i) - \bar{z}_i|\} \tag{7.21}$$

in which the goal region is defined as $R_m = \{(7, 12, 1)\}$ which corresponds to $\{(\bar{x}_1, \bar{y}_1, \bar{z}_1)$, $(\bar{x}_2, \bar{y}_2, \bar{z}_2)$, $(\bar{x}_3, \bar{y}_3, \bar{z}_3)\}$. Subscript 1 is used to represent the overhead crane, 2 represents the mobile robot, and 3 represents the robot manipulator. Note that for the mobile robot $z_2 = \bar{z}_2$. We choose

$$V(r) = \rho(r, R_m). \tag{7.22}$$

If we choose $c_1 = 1$ and $c_2 = 1$, it can be seen that $c_1\rho(r, R_m) = V(r) = c_2\rho(r, R_m)$, which satisfy the first condition of Theorem 4.2.3.

For the second condition of Theorem 4.2.3, assume that at the goal state, each agent has taken N_i actions. If we choose $c_3 < \frac{\max_{k=1}^{N_i-1} |\rho(\mathcal{R}(r_0,\mathcal{E}_{k+1},k+1))-\rho(\mathcal{R}(r_0,\mathcal{E}_k,k))|}{\rho(r_0,R_m)}$, it can be seen that the second condition of Theorem 4.2.3, $V(\mathcal{R}(r_0, \mathcal{E}_{k+1}, k+1)) - V(\mathcal{R}(r_0, \mathcal{E}_k, k)) \leq -c_3(\rho(\mathcal{R}(r_0, \mathcal{E}_k, k), R_m))$, is also satisfied.

Therefore, the invariant set R_m is exponentially stable w.r.t $\mathcal{E}_a$. ∎

7.2 Optimization of the CHA MAS

In Figure 6.12, the simulation results for the cooperation and coordination between the mobile robot and the robot manipulator are given. In the figure, the round object represents the load of the overhead crane, while the square object represents the mobile robot. The trajectories of both the overhead crane and the mobile robot are given. For clarity, the trajectory of the robot manipulator is only partially shown, as represented by a small square. From the figure, we can see that the overhead crane starts from the initial position and delivers the object to the loading area to wait for the mobile robot to pick up the object. The mobile robot follows the landmarks into the loading area and picks up the object. Then, the mobile robot turns around.

From Theorem 5.4.3, we know that the busy period structure of an optimal sample path is unique in the sense that for any a_{r_k}, $k = 0, \ldots, N_i - 1$, the last action of the busy period

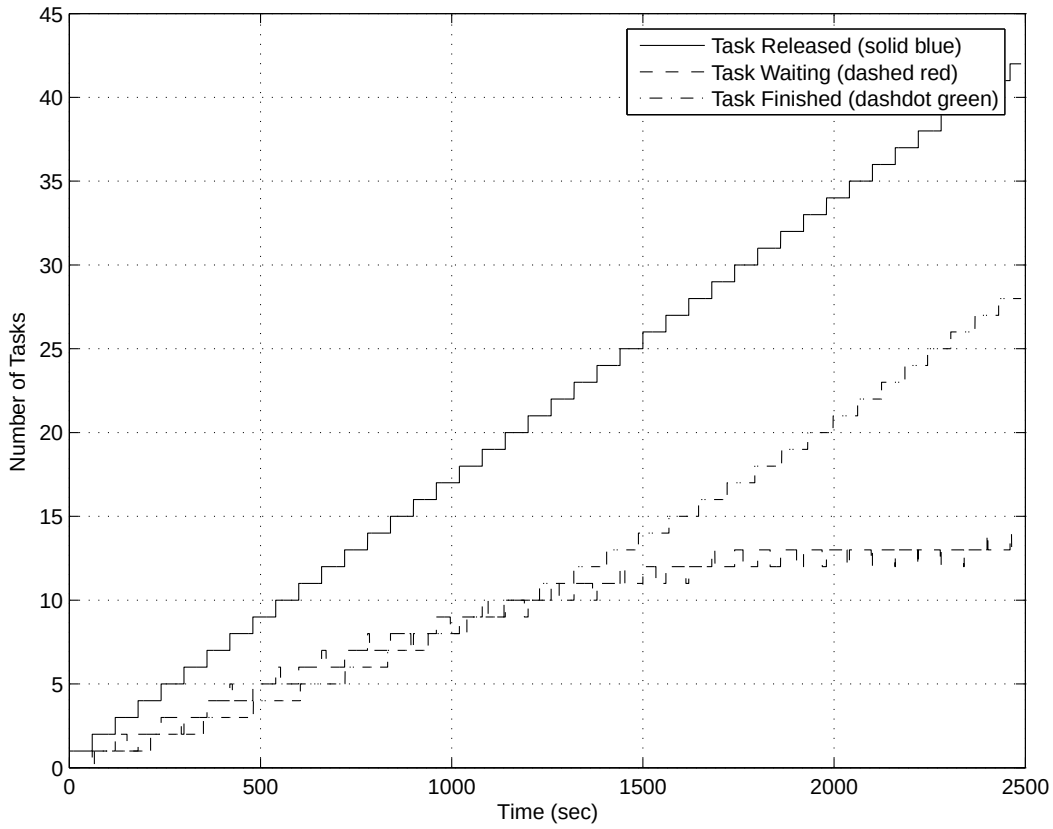

Figure 7.2: The Tasks that the MAS Has Finished During and After the Optimization Process.

containing a_{r_k} is unique on the optimal sample path. Based on the theorem, the direct identification algorithm is proposed in Chapter 5 to identify the busy period structure. In this section, the direct identification algorithm is applied to the optimization of the MAS. After we apply the direct identification algorithm, we are able to identify the busy period structure of the MAS. From Figure 7.2, it can be seen that after the direct identification algorithm is applied, the number of waiting tasks becomes a constant at about 1700s. This means that the tasks released at a constant rate can be finished by the optimized MAS while the quality of the tasks is also maintained.

The simulation results show that for a heterogeneous MAS modeled by the CHA framework as described in Scenario 1.1.2 (i.e., the overhead crane delivers an object in its workspace to the designated area, then with the vision navigation control, the mobile robot picks up the object from the crane's workspace and delivers it to the robot manipulator. The robot manipulator then picks up the object and transports it to its own workspace), through the application of the proposed direct identification algorithm to the optimization of this MAS, we are able to achieve optimized coordinated control of the heterogeneous MAS. The agents cooperatively work together to achieve the desired global goal with the optimal function satisfied. As far as we know, there is no similar research done to optimize the performance of a MAS which considers both the time-driven dynamics and the event-driven dynamics.

7.3 Summary

In this chapter, we have given some examples of the stability and optimization analysis using the methodology we have proposed. The stability of the homogeneous and the heterogeneous MASs presented in the previous chapter is analyzed. In addition, we also apply the direct identification algorithm we have proposed to optimize the performance of the heterogeneous MAS. In the next chapter, we will conclude this research and give some future research directions.

Chapter 8

Conclusions and Future Work

Multi-agent systems represent a group of agents operating cooperatively to solve common tasks in dynamic environments. In this study, a generic framework is proposed for the control of multi-agent systems.

8.1 Summary

Chapter 1 gives a brief introduction of this dissertation.

In Chapter 2, we have reviewed the background information about various areas related to this research.

In Chapter 3, a framework is proposed for the distributed control and coordination of multi-agent systems. In the proposed framework, the control of multi-agent systems focuses on decentralized control and coordination of agents. Each agent is modeled as a CHA which is composed of an intelligent coordination control layer and a hybrid control layer. The core of the proposed framework is on developing coordinated agents for the control of hybrid multi-agent systems. A robust and generic control architecture is developed to control either a homogeneous multi-agent system or a heterogeneous multi-agent system. The proposed framework is able to model the cooperation, coordination and communication among the members of the multi-agent system. The control scheme is able to control a multi-agent system where agents cooperate, coordinate and interact with each other.

In Chapter 4, we discuss the stability of MASs modeled using the CHA framework.

We are interested in both the local stability and the global stability of MASs. The local stability is used to describe each single agent's ability to maintain the stability of one entity in a MAS. The global stability of a MAS describes the ability of a group of agents' ability to achieve a desired goal. In order to achieve the global stability, the local stability of all agents has to be guaranteed. For the local stability, we analyze the stability of a CHA which is modeled as a hybrid system. In order to analyze the global stability of a MAS, each CHA can be treated as a DES at the upper layer. The dynamics of the DES evolve in time with the occurrence of events at possibly irregular time intervals.

In Chapter 5, the optimization of MASs modeled by the CHA framework is studied. We consider both time-driven dynamics and event-driven dynamics for the optimization of a CHA system. The optimization problem of the MASs is analyzed. An example is also given to illustrate how to define the optimization problem for a CHA. The direct identification algorithm is introduced for solving the optimal control problem of a CHA MAS.

Chapter 6 gives some experimental and simulation results for systems modeled using the proposed framework. The goal is to implement the tools we have introduced to develop the control algorithm for multi-agent systems. The feasibility of the proposed framework is illustrated through three different scenarios. It is demonstrated that the proposed framework is generic and can be applied to the control of both homogeneous and heterogeneous MASs.

Chapter 7 gives some stability and optimization analysis using the methodology we have proposed in the previous chapters. In this chapter, the stability of the homogeneous and the heterogeneous MASs is analyzed. In addition, we also apply the direct identification algorithm we have proposed to optimize the performance of the heterogeneous MAS.

8.2 Conclusion

In the proposed framework, the control of multi-agent systems is regarded as achieving decentralized control and coordination of agents. Each agent is modeled as a CHA which is composed of an intelligent coordination layer and a hybrid control layer. The intelligent coordination layer deals with the planning, coordination, decision-making and computa-

tion of the agent. The hybrid control layer of the proposed framework takes the output of the intelligent coordination layer and generates discrete and continuous control signals to control the overall process. The proposed framework is able to model the cooperation, coordination, and communication of MASs. In order to verify the feasibility of the proposed framework, experiments for both heterogeneous and homogeneous MASs are implemented. In addition, the stability of systems modeled using the proposed framework is also analyzed. The conditions for asymptotic stability and exponential stability of a CHA system are given. In order to optimize a MAS, a hybrid approach is proposed to address the optimization problem for a MAS modeled using the CHA framework. Both the event-driven dynamics and time-driven dynamics are included for the formulation of the optimization problem. A direct identification algorithm is also proposed to solve the optimization problem. As a conclusion, the proposed framework is able to model MASs. We have also provided a methodology for the stability analysis of MASs modeled using the proposed framework. Optimization analysis is also given for the proposed framework.

8.3 Future Work

Some areas of future research that have potential to extend the results of this research are presented in this section.

8.3.1 Fault Tolerance and Reconfiguration

MASs take the advantage of distributed control to concurrently control multiple entities. The agent-based control strategy provides more flexibility, potential for greater functionality, and unfortunately, more pieces to break. In order to solve problems when modules of a MAS fail, strategies for reconfiguration of MASs are necessary to provide fault tolerance and flexibility to the system. Reconfiguration mechanisms lead to the design of robust systems that have the capability to allow the service continuity, in the presence of a failure, on the basis of a minimal degradation of performances. A successful reconfiguration strategy can provide self-reconfigurable agents. A group of the modules can thus generate various structures and actions. Although the module itself is a simple mechanism, self-reconfiguration presents a challenging control problem due to the many combinatorial

possibilities of modular configurations in an agent. The reconfiguration strategy can be developed based on multi-module blocks to plan the overall clustering strategies and also to provide cooperative module motions. The hardware feasibility of the reconfiguration strategy should be verified through self-reconfigurable agents. Future research should be done to include reconfiguration capabilities in an agent. New modules could be included in the proposed framework to model reconfiguration of an agent or the MAS.

8.3.2 Learning of MASs

Reinforcement learning can be applied to multi-agent systems to take into account the needs and behaviors of other agents, and to learn to perform effectively. Cooperation among agents during learning is essential in improving global performance. We need to develop coordination methods based on learning, which enables an agent to learn by observing other agents and the effects on the workspace. Agents learn to coordinate their actions by including information about other agents. An agent can receive rewards from its own actions as well as receive some rewards from neighboring agents. In this way, each agent acts in a social way and the groups of agents learn to behave cooperatively without any conflicts.

8.3.3 Optimization of Abstract State Evolution

In Chapter 5, we consider both time-driven dynamics and event-driven dynamics for the optimization of a CHA system. As further study, we should also present a method to select the optimal actions. The irrelevant actions make the problem difficult. For various problems, we need a methodology to optimize the actions that agents take. It is not clear how to define a good heuristic function in order to select the optimal actions. It is obvious that searching without an accurate heuristic is out of the question. We have shown that the rule base we have proposed can limit the number of actions that an agent can take. An intelligent planner can also be developed to automatically generate the action sequence. However, we still need to construct more sophisticated heuristics that examine the available actions as well as the structure of the global goal. Then, it becomes possible to find an optimal sequence of actions for the agents.

Bibliography

[1] S. Akella and S. Hutchinson. Coordinating the motions of multiple robots with specified trajectories. In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, pages 624–631, Washington DC, USA, 2002.

[2] C. Arbib and F. Rossi. Optimal resource assignment through negotiation in a multi-agent manufacturing system. *IIE Transactions (Institute of Industrial Engineers)*, 32(10):963–974, 2000.

[3] J. Baeza, D. Gabriel, J. Bejar, and J. Lafuente. A distributed control system based on agent architecture for wastewater treatment. *Computer-Aided Civil and Infrastructure Engineering*, 17(2):93–103, 2002.

[4] G. Barrett and S. Lafortune. Decentralized supervisory control with communicating controllers. *IEEE Transactions on Automatic Control*, 45(9):1620–1638, 2000.

[5] A. H. Bond and L. Gasser. *An Analysis of Problems and Research in Distributed Artificial Intelligence*. Morgan Kaufman Publishers, 1988.

[6] R. I. Brafman and M. Tennenholtz. On partially controlled multi-agent systems. *Journal of Artificial Intelligence Research*, 4:477–507, 1996.

[7] M. Branicky, V. S. Borkar, and S. K. Mitter. A unified framework for hybrid control: model and optimal control theory. *IEEE Transactions On Autamatic Control*, 43(1):31–45, 1998.

[8] R. W. Brennan, M. Fletcher, and D. H. Norrie. An agent-based approach to reconfiguration of real-time distributed control systems. *IEEE Transactions on Robotics and Automation*, 18(4):444–449, August 2002.

[9] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[10] C. G. Cassandras, D. L. Pepyne, and Y. Wardi. Optimal control of a class of hybrid systems. *IEEE Transactions on Automatic Control*, 46(3):398–415, 2001.

[11] F. T. S. Chan and J. Zhang. A multi-agent-based agile shop floor control system. *International Journal of Advanced Manufacturing Technology*, 19(10):764–774, 2002.

[12] Y. C. Cho, C. G. Cassandras, and D. L. Pepyne. Forward decomposition algorithms for optimal control of a class of hybrid systems. *International Journal of Robust and Nonlinear Control*, 11, 2001.

[13] V. Crespi, G. Cybenko, D. Rus, and M. Santini. Decentralized control for coordinated flow of multi-agent systems. In *Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN)*, pages 2604–2609, 2002.

[14] J. B. de Sousa and F. L. Pereira. Real-time hybrid control of multiple autonomous underwater vehicles. In *Proceedings of the 1998 37th IEEE Conference on Decision and Control (CDC)*, pages 2645–2649, Tampa, FL, USA, 1998.

[15] D. Driankov, H. Hellendoorn, and M. Reinfrank. *An Introduction to Fuzzy Control*. Berlin, Springer, 1998.

[16] M. G. Earl and R. D'Andrea. Modeling and control of a multi-agent system using mixed integer linear programming. *Proceedings of 41st IEEE Conference on Decision and Control*, 1:107–111, 2002.

[17] L. Erman, F. Hayes-Roth, V. Lesser, and Reddy. The hearsay-ii speech understanding system: Integrating knowledge resolve uncertainty. *ACM Computing Surveys*, 12, 1980.

[18] B. Fan, Q. Pan, and H. C. Zhang. A multi-agent coordination framework based on markov games. In *Proceedings of CSCWD 2004 - 8th International Conference on Computer Supported Cooperative Work in Design*, pages 230–233, Xiamen, China, 2004.

[19] J. Ferber. *Multi-Agent Systems - An Introduction to Distributed Artificial Intelligence*. Addison-Wesley, 1999.

[20] M. S. Franzin, F. Rossi, E. C. Freuder, and R. Wallace. Multi-agent constraint systems with preferences: Efficiency, solution quality, and privacy loss. *Computational Intelligence*, 20(2):264–286, 2004.

[21] K. Fregene. *Distributed Intelligent Control of Hybrid Multiagent Systems*. University of Waterloo, 2002.

[22] K. Fregene, D. Kennedy, and D. Wang. Hica: A framework for distributed multiagent control. *IASTED International Conference on Intelligent Systems and Control*, pages 187–192, 2001.

[23] L. E. Garza, F. J. Cantu, and S. Acevedo. Faults diagnosis in industrial processes with a hybrid diagnostic system. In *Proceedings of MICAI 2002: Advances in Artificial Intelligence. Second Mexican International Conference on Artificial Intelligence*, pages 536–545, 2002.

[24] J. Gjerdrum, N. Shah, and L. G. Papageorgiou. A combined optimization and agent-based approach to supply chain modelling and performance assessment. *Production Planning and Control*, 12(1):81–88, 2001.

[25] J. L. Sanchez Gonzalez, M. Mediavilla Pascual, J. C. Fraile Marinero, F. Gayubo Rojo, J. Perez Turiel, and F. J. Garcia Gonzalez. Analysis of utilization and throughput in a multirobot system. In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, pages 96–101, Washington DC, USA, 2002.

[26] T. Heikkila, M. Kollingbaum, P. Valckenaers, and G. J. Bluemink. An agent architecture for manufacturing control: Manage. *Computers in Industry*, 46(3):315–331, October 2001.

[27] A. Helsinger, K. Kleinmann, and M. Brinn. A framework to control emergent survivability of multi agent systems. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 28–35, New York, NY, USA, 2004.

[28] C. Hewitt. Viewing control structures as patterns of passing messages. *Artificial Intelligence*, 8(3):323–364, 1977.

[29] M. Heymann, F. Lin, and G. Meyer. Multiuser discrete-event control with active events. *IEEE Transactions on Automatic Control*, 47(2):314–318, February 2002.

[30] T. Huntsberger, P. Pirjanian, A. Trebi-Ollennu, H. D. Nayar, H. Aghazarian, A. J. Ganino, M. Garrett, S. S. Joshi, and P. S. Schenker. Campout: A control architecture for tightly coupled coordination of multirobot systems for planetary surface exploration. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, 33(5):550–559, September 2003.

[31] Y. Indrayadi, H. P. Valckenaers, and H. Van Brussel. Dynamic multi-agent dispatching control for flexible manufacturing systems. In *Proceedings 13th International Workshop on Database and Expert Systems Applications*, pages 489–493, 2002.

[32] N. R. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75(2):195–240, June 1995.

[33] F. Karray, O. Basir, I. Song, and H. Li. A framework for coordinated control of multi-agent systems. In *Proceedings of IEEE International Symposium on Intelligent Control*, pages 156–161, Taipei, Taiwan, 2004.

[34] J. Kosakaya, A. Kobayashi, and K. Yamaoka. Distributed supervisory system with cooperative multi-agent fep. In *Proceedings 22nd International Conference on Distributed Computing Systems Workshops*, pages 633–638, 2002.

[35] H. Y. K. Lau, A. Ko, and T. L. Lau. A decentralized control framework for modular robots. In *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1774–1779, Sendai, Japan, 2004.

[36] M. K. Lim and Z. Zhang. A multi-agent based manufacturing control strategy for responsive manufacturing. *Journal of Materials Processing Technology*, 139(1-3 SPEC):379–384, August 2003.

[37] C. H. Lin and K. T. Song. Flexible real-time control of home robots using a multi-agent based approach. In *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3092–3097, Sendai, Japan, 2004.

[38] P. Lucidarme, O. Simonin, and A. Liegeois. Implementation and evaluation of a satisfaction/altruism based architecture for multi-robot systems. In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, pages 1007–1012, Washington DC, USA, 2002.

[39] J. Lygeros. *Hierarchical, Hybrid Control of Large Systems*. PhD Thesis, University of California, Berkeley, CA, 1996.

[40] N. Lynch. I/o automata: A model for discrete event systems. *Technical Memo MIT/LCS/TM-351, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA*, March 1988.

[41] N. Lynch, R. Segala, and F. Vaandraager. Hybrid i/o automata. *Information and Computation*, 185(1):105–157, August 2003.

[42] R. Madhavan, K. Fregene, and L. E. Parker. Distributed heterogeneous outdoor multi-robot localization. In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, pages 374–381, Washington DC, USA, 2002.

[43] M. J. Mataric. Learning in behavior-based multi-robot systems: policies, models and other agents. *Cognitive Systems Research*, 2:81–93, 2001.

[44] H. Nishiyama, W. Yamazaki, and F. Mizoguchi. Negotiation protocol for proof of realization of cooperative task in multi-agent robot systems. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pages 1685–1690, 3 2000.

[45] P. Ogren, E. Fiorelli, and N. E. Leonard. Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *IEEE Transactions on Automatic Control*, 49(8):1292–1302, August 2004.

[46] L. Overgaard, Nelson B. J., Khosla, and Pradeep K. Multi-agent framework for grasping using visual servoing and collision avoidance. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2456–2461, Minneapolis, MN, USA, 1996.

[47] Z. Papp and H. J. Hoeve. Multi-agent based modeling and execution framework for complex simulation, control and measuring tasks. In *17th IEEE Instrumentation and Measurement Technology Conference 'Smart Connectivity: Integrating Measurement and Control'*, pages 1561–1566, Baltimore, MD, USA, 2000.

[48] K. M. Passino and K. L. Burgess. *Stability Analysis of Discrete Event Systems*. John Wiley & Sons, Inc., 1998.

[49] K. M. Passino, A. N. Michel, and P. J. Antsaklis. Lyapunov stability of a class of discrete event systems. *IEEE Transactions on Automatic Control*, 39(2):269–279, 1994.

[50] C. D. Paternina-Arboleda and T. K. Das. A multi-agent reinforcement learning approach to obtaining dynamic control policies for stochastic lot scheduling problem. *Simulation Modelling Practice and Theory*, 13(5):389–406, 2005.

[51] D. L. Pepyne and C. G. Cassandras. Modeling, analysis, and optimal control of a class of hybrid systems. *Discrete Event Dynamic Systems: Theory and Applications*, 8(1):175–201, 1998.

[52] G. A. S. Pereira, B. S. Pimentel, L. Chaimowicz, and M. F. M. Campos. Coordination of multiple mobile robots in an object carrying task using implicit communication. In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, pages 281–286, Washington DC, USA, 2002.

[53] I. Romanovski and P. E. Caines. On the supervisory control of multi-agent product systems. In *Proceedings of 41st IEEE Conference on Decision and Control*, pages 1181–1186, Las Vegas, NV, USA, 2002.

[54] S. J. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach*. Pearson Education, Inc., 2003.

[55] P. Sellem, E. Amram, and D. Luzeaux. Open multi-agent architecture extended to distributed autonomous robotic systems. In *Proceedings of the SPIE - The International Society for Optical Engineering*, pages 170–177, USA, 2000.

[56] Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: off-line design. *Artificial Intelligence*, 73(1-2):231–252, 1995.

[57] R. Sikora and M. J. Shaw. Coordination mechanisms for multi-agent manufacturing systems: Applications to integrated manufacturing scheduling. *IEEE Transactions on Engineering Management*, 44(2):175–187, May 1997.

[58] J. A. A. Sillince. Multi-agent conflict resolution: a computational framework for an intelligent argumentation program. *Knowledge-Based Systems*, 7(2):75–90, June 1994.

[59] J. Soler, V. Julian, C. Carrascosa, and V. Botti. Applying the artis agent architecture to mobile robot control. In *Proceedings of Advances in Artificial Intelligence. International Joint Conference 7th Ibero-American Conference on AI 15th Brazilian Symposium on AI IBERAMIA-SBIA 2000*, pages 359–368, 2000.

[60] I. Song, F. Karray, and F. Guedea. An advanced control framework for a class of distributed real-time systems. In *Proceedings of the 2004 the World Automation Congress*, pages 62–67, Spain, 2004.

[61] P. Song and V. Kumar. A potential field based approach to multi-robot manipulation. In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, pages 1217–1222, Washington DC, USA, 2002.

[62] W. M. Spears, D. F. Spears, J. C. Hamann, and R. Heil. Distributed, physics-based control of swarms of vehicles. *Autonomous Robots*, 17(2-3):137–162, September/November 2004.

[63] P. Tabuada, G. J. Pappas, and P. Lima. Compositional abstractions of hybrid control systems. *Discrete Event Dynamic Systems: Theory and Applications*, 14(2):203–238, April 2004.

[64] G. W. Tan, C. C. Hayes, and M. Shaw. Intelligent-agent framework for concurrent product design and planning. *IEEE Transactions on Engineering Management*, 43(3):297–306, August 1996.

[65] J. Wang, H. Li, F. Karray, and O. Basir. Fuzzy anti-swing control of a behavior-based intelligent crane system. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, Las Vegas, NV, USA, October 2003.

[66] Z. Wang and V. Kumar. Object closure and manipulation by multiple cooperating mobile robots. In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, pages 394–399, Washington DC, USA, 2002.

[67] R. R. Yager. Penalizing strategic preference manipulation in multi-agent decision making. *IEEE Transactions on Fuzzy Systems*, 9(3):393–403, 2001.

[68] S. X. Yang and M. Meng. An efficient neural network approach to dynamic robot motion planning. *Neural Networks*, 13(2):143–148, 2000.

[69] F. Zanichelli, S. Caselli, A. Natali, and A. Omicini. Multi-agent framework and programming environment for autonomous robotics. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3501–3507, San Diego, CA, USA, 1994.

[70] H. C. Zhang and J. Li. Using fuzzy multi-agent decision-making in environmentally conscious supplier management. *CIRP Annals - Manufacturing Technology*, 52(1):385–388, 2003.