DISTRIBUTED COALITION FORMATION FOR COLLABORATIVE LARGE-SCALE MULTI-AGENT SYSTEMS

BY

PREDRAG TOŠIĆ

B.S., University of Maryland Baltimore County, 1994
M.S., University of Maryland Graduate School in Baltimore, 1995
M.S., University of Illinois at Urbana-Champaign, 1998

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2006

Urbana, Illinois

Acknowledgments

First and foremost, I would like to thank my adviser, professor Gul Agha, for his guidance and support since 2001. I would also like to sincerely thank my colleagues from professor Agha's Open Systems Laboratory (OSL) – especially those who were involved in the TASK research project. Among them, I am particularly greatly indebted to Reza Ziaei, as well as Nirman Kumar and Myeong-wuk Jang. I would also like to thank Wooyoung Kim, Amr Ahmed, Tom Brown, Nadeem Jamali, Prasannaa Thati, Smitha Reddy, Joshua Chia, Liping Chen, Soham Mazumdar, and Abhilash Patel. If I have forgotten anyone with whom I have had an interaction related to the work presented in this thesis, I hope they will forgive me. I would also like to acknowledge very helpful feedback from professors Michael Loui and Les Gasser.

My research that eventually led to this thesis was supported by the DARPA IPTO TASK program, contract # F30602-00-2-0586.

Last but not least, neither the research work summarized in this thesis, nor anything else that I have ever accomplished in my life, would have ever been possible without the unconditional love and support from my family in Serbia.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AAMAS Autonomous Agents and Multi-Agent Systems

CNP Contract Net Protocol

(D)AI (Distributed) Artificial Intelligence

(D)CO(P) (Distributed) Constraint Optimization (Problem)

(D)CS(P) (Distributed) Constraint Satisfaction (Problem)

DPS Distributed Problem Solving

(D)DTA(P) (Dynamic) Distributed Task Allocation (Problem)

(D)SC(P) (Distributed) Set Covering (Problem)

(D)SP(P) (Distributed) Set Partitioning (Problem)

MCDCF Maximal Clique-based Distributed Coalition Formation

(M)MAS (Massive) Multi-Agent System(s)

NP Nondeterministic Polynomial time (computational complexity class)

OSL Open Systems Laboratory

(PO)MDP (Partially Observable) Markov Decision Process

P2P Peer-to-Peer

TSP Travelling Salesman Problem

UAV Unmanned Aerial Vehicle

CHAPTER 1

INTRODUCTION

This thesis addresses a particular, frequently encountered coordination problem in *collaborative multi-agent systems.* Autonomous agents, as well as multi-agent systems (MAS) made of such autonomous agents, are a growing and exciting research area that spans many scientific disciplines, from economics to social sciences to distributed computing to artificial intelligence (e.g., [1, 60, 63]). Our work in multi-agent systems, an important part of which is summarized in this thesis, is a theoretically-oriented computer science research, and, as such, it chiefly overlaps theory of distributed computing with (distributed) AI.

Applications where autonomous agents reside in dynamically changing environments and are required to accomplish various tasks in real time in those environments pose a number of modeling, design and analysis challenges. From a MAS designer's standpoint, some of the major challenges include finding appropriate models for coordination among the agents, as well as for an individual agent's capabilities of adaptation (for example, via reinforcement learning) and autonomous decision making (such as selecting one among a finite set of available actions).

This thesis focuses on a particular aspect of the general problem of coordination in large-scale multi-agent systems, namely, the problem of coalition formation for the purpose of collaborative task allocation [21, 22, 43, 44, 47, 54].

1.1 Motivation

Autonomous agents have become a powerful paradigm in a great variety of disciplines, from sociology and economics to distributed artificial intelligence and software engineering to cognitive

sciences to philosophy. While different disciplines have different needs and may have different notions of *(autonomous) agents,* the agents in economics and those in distributed artificial intelligence (DAI), for example, nonetheless tend to share most of the fundamental properties. Indeed, it has become common to define an appropriate notion of *agency* by specifying the *necessary attributes* that all agents of the particular kind one has in mind are required to share (e.g., [10, 31, 35]). There has been much of debate, however, what set of properties exactly qualifies an entity, such as a single human decision maker, a firm in the market, a computer program, a robot or an unmanned autonomous vehicle, for an *autonomous* or an *intelligent* agent. Influential position papers, such as [62] for intelligent agents or [10] for autonomous agents, while trying to clarify and unify the terminology, as well as propose useful and broadly applicable agent taxonomies, also illustrate heterogeneity and lack of full agreement on the definition and the required (as opposed to optional) properties even in the case of those autonomous agents that are restricted to computer programs alone (which disallows, say, humans, firms or social insects).

It has been observed that the main division line is the one that separates the (purely) *reactive agents* [33, 35] from the more complex, capable of cognitive-like behaviors *deliberative agents* [31, 35, 62]. A reactive agent is one that is coupled to the environment and is capable of being affected by, and perhaps in turn also affecting, the environment. It need not be capable of cognitive tasks such as learning, planning or reasoning. It need not have any complicated internal structure, or any capability of complex correlations between its internal states and the states of the outer world ("symbolic representations"); it uses little or no memory, etc.

In contrast, a deliberative agent is much more complex in terms of its internal structure, is typically capable of creating and working with abstract representations of a complex outer world (e.g., by performing planning, reasoning and/or learning tasks), has some sense of its purpose (tasks, goals, utilities), usually is pro-active and adaptable, etc. Much of research in the mainstream artificial intelligence (AI) over the past twenty or more years has been focused on the design problem of such artificial deliberative agents, capable of acting in complex environments and autonomously pursuing their complex goals or tasks in such environments (see, e.g., [1, 31, 35, 60, 62] and references therein).

The focus of our research summarized in this thesis is on autonomous agents that are char-

acterized by (i) deliberateness, (ii) resource limitations, and (iii) being embedded in a dynamic, partially observable environment whose main ingredients are other, similar agents, as well as tasks that need to be serviced, and that require some amount of resources in order to be completed.

Autonomous agents and multi-agent systems studied in computer science [1, 10, 60, 62] are characterized, among other properties, by a considerable degree of autonomy of individual computing entities or processes (such as software or robotic agents) and, at the same time, the fact that each agent has a *local,* and therefore *incomplete* and *imperfect* picture of the world. Since in MAS there is either no central control, or only a limited central control, the individual agents often have to reason and act *strictly locally* (e.g., [27, 43, 50, 54, 62]). Hence, *genuinely distributed* algorithms are frequently needed so that the agents can effectively coordinate with one another. The need for the fully decentralized coordination strategies is particularly critical for very large agent ensembles – those that include from thousands up to millions of autonomous agents – in which the resources of each agent individually are severely limited. In such large-scale, bounded-resource MAS [50], any feasible and scalable coordination strategy, in addition to being decentralized, also better be based chiefly or even solely on *local* computations and communication. The coalition formation algorithm that we propose in Chapter 4 satisfies both of these two critical criteria: it is (i) fully decentralized and (ii) strictly local.

From a general distributed computing viewpoint, MAS pose a number of challenges to a distributed algorithm designer. Many of those challenges are related to various aspects of the *agent coordination problem* [1, 60, 66]. In order to be able to effectively coordinate, agents need to be able to *reach consensus (agreement)* [20, 48] . Two particularly common consensus problems are *leader election* and *group (or coalition) formation.*

Group or coalition formation has been an important issue in distributed systems in general [20], and in multi-agent systems, in particular [42, 43, 54, 66]. Given a collection of communicating agents, the goal in distributed coalition formation is that these agents, based on their local knowledge and communication with other agents *only,* decide how to effectively split up into several coalitions, so that each agent knows which coalition(s) it belongs to.

There are several critical issues that a MAS designer needs to address in the context of (distributed) coalition formation. First, what is the right notion of a coalition in a given setting? For

example, can the coalitions overlap? Second, a distributed mechanism for coalition formation needs to be provided. Third, a mechanism that enables an agent to maintain and, if need be, update its knowledge about its coalition membership needs to be provided. Fourth, how are the agents to decide, when is the right time to transform, or perhaps dissolve, the existing coalitions? These and other challenges related to autonomous agents forming coalitions have been extensively studied in the literature on multi-agent systems; see, e.g., [12, 21, 22, 23, 34, 42, 43, 44, 66]. They have also arisen in our research group's work on the parametric models and a scalable software simulation of large ensembles ($10^3 - 10^4$ agents) of autonomous unmanned aerial vehicles deployed in a multi-task mission [15, 16, 49, 50].

In this thesis, the focus will be on the second issue above, namely, on providing ensembles of cooperative agents with a fully decentralized mechanism for forming non-overlapping coalitions.

A need for a distributed, online and efficient coalition formation may arise due to a number of different factors, such as the geographical dispersion of the agents, heterogeneity of the agents' tasks and those tasks' resource requirements, heterogeneity or insufficiency of the individual agents' capabilities, and so on [21, 43, 50]. While, for the small- to medium-scale systems of robots, unmanned vehicles or other multi-agent systems, a fully or partially centralized approach to coalition formation and maintenance may be feasible and even optimal, large scale systems appear to require a fully decentralized approach [50, 54]. That is, in such systems, the agents need to be able to self-organize into coalitions without any remote run-time (human or other) intervention, and quickly reach a consensus on who is forming a coalition with whom [51, 54]. We will briefly discuss several such large-scale MAS application domains in Chapter 2.

1.2 Our Main Contributions

The main contribution of this thesis is a novel distributed graph algorithm that we propose as a coalition formation strategy in cooperative multi-agent systems. We shall motivate, describe and analyze the proposed algorithm in some detail in Chapters 2, 4 and 5, respectively. We will also show that the proposed approach to coalition formation is feasible and scalable under certain assumptions that, we argue, actually hold in a number of important large-scale MAS applications. The two most important assumptions are that (i) the underlying communication network topology

of the multi-agent system in question is sufficiently *sparse,* and that, insofar as the MAS with *dynamically changing network topologies* are concerned, these topology changes take place relatively slowly.

The basic idea behind the *Maximal Clique-based Distributed Coalition Formation* (MCDCF) algorithm, which will be described in detail in Chapter 4, is to efficiently and in a fully decentralized manner *partition* the graph representing the communication network of agents into (preferably, *maximal*) *cliques* of nodes. These maximal cliques would, in practical applications, usually also need to satisfy some additional criteria in order to form temporary coalitions of desired quality. Once formed and deployed to accomplish their tasks, these coalitions of agents are then maintained until they are no longer useful or meaningful.

We envision distributed coalition formation based on our approach to find its application in the dynamic, collaborative, bounded-resource MAS domains where the communication network topology undergoes periodic changes, and, furthermore, where the agents do not have sufficient resources or knowledge of their environment for sophisticated long-term planning. In particular, any considerable network topology changes, as will be discussed in much more detail in Chapters 4 and 5, would require the agents to re-transform the existing coalitions and form new ones. In particular, this observation implies that the proposed coalition formation algorithm, or its appropriate variants, may need to be invoked a number of times during the collaborative MAS deployment as a basic coordination subroutine.

1.3 Thesis Outline

The rest of this thesis is organized as follows. The preliminaries are covered in Chapter 2. Specifically, we introduce the problem of coalition formation in MAS, and outline some of the main challenges. We then focus on the main properties of *locally constrained collaborative multi-agent environments* [53, 54] that are critical for the applicability of our approach to coordination via coalition formation. We also include some examples of the large-scale multi-agent systems that are characterized by the *communication network topologies* that are (i) dynamic, (ii) physically *localized,* as well as (iii) *sparse* in the usual graph-theoretic sense – and are thus good candidate applications for our approach to coalition formation.

Chapter 3 provides a brief overview of some of the most relevant related work. We start with a short and informal introduction to *distributed constraint satisfaction,* as a generic problem in Distributed AI that places our work into a broader context. We then discuss some research directions found in the MAS literature on *task allocation via coalition formation,* which is the more specific problem that this thesis attempts to address.

Chapter 4 is the central part of this thesis. In section 4.1, we define a class of task allocation via coalition formation (distributed) constraint optimization problems, including the particular problem we address. We then outline the approach taken, and state the main assumptions that need to hold in order for our approach to work. Once the stage has been set, we describe in some detail our *Maximal Clique-based Distributed Coalition Formation* (MCDCF) algorithm in section 4.2, and illustrate the operation of the algorithm on a small example in section 4.3.

Chapter 5 starts with the cost analysis of the algorithm, followed by discussion of some of the main features, strengths and weaknesses of our approach to distributed coalition formation, and a brief outline of some possible extensions and generalizations of that approach.

Finally, we summarize this thesis in Chapter 6. The pseudo-code for the MCDCF algorithm is given in *Appendix.*

CHAPTER 2

PRELIMINARIES

In this chapter, we overview the theoretical background underlying the problem of coalition formation in multi-agent systems. We first briefly survey different types of MAS and their environments. Insofar as the nature of relationship among the agents, we discuss whether those agents are *collaborative* or *competitive,* and how does the nature of this relationship affect the coalition formation among the agents. Insofar as the nature of the agents' environment is concerned, we focus on the issue of whether, informally speaking, the assumption that the bigger a coalition of agents, the better, holds or does not hold.

Once we have classified the fundamental types of relationships among the agents and their environments, we shall focus on the scenario that has been the main focus of our work on multi-agent systems thus far (as of early 2005), namely, the collaborative, locally constrained MAS [15, 49, 50, 52, 51, 53, 54]. In that context, we shall briefly overview the basics of the *cooperative N-person game theory* [19, 32], as well as of the classical *decision theory* [27, 30], and discuss the practical limitations of those theories when applied to MAS, as well as mention some modifications that are required in order for those classical mathematical theories to become applicable in the bounded-resource, bounded-rationality multi-agent domains.

Last but not least, we will also provide some examples of concrete collaborative MAS characterized by (i) the bounded computational, communication and/or energy resources of each agent, where the limits on these critical resources need to be explicitly taken into account, and (ii) the sparse communication network topologies. It is the application domains in which both properties (i) and (ii) hold that we find particularly well-suited for our MCDCF-based approach to coalition formation and task allocation [50, 53, 54].

2.1 On Coalition Formation in Large-Scale Multi-Agent Systems

We have argued in Chapter 1 that distributed coalition formation is of a considerable interest for many different kinds of autonomous agents and multi-agent systems, including both those MAS where all agents share the same goal and thus strive to optimize an appropriately defined *global utility* function, as well as those where the agents do not share a global utility function, and where each agent generally acts in its own, individual self-interest. Based on whether each agent strives to optimize its individual utility or payoff, or all agents share the same objective and thus optimize a joint utility function, the MAS domains are generally divided into *competitive* and *cooperative.* However, both types of agents usually need to be able to effectively coordinate with each other – including, but not limited to, coordination via coalition formation.

In case of the competitive, self-interested agents, the critical issue is how are the benefits of forming a coalition going to be distributed among the individual agents; likewise, whether an agent joins a coalition that other agent(s) propose to it, depends strictly on whether this particular agent, insofar as its individual utility is concerned, is going to be better off if it joins the proposed coalition than if it does not. These issues do not arise in the purely collaborative, *distributed problem solving* (DPS) multi-agent domains, where there is only one utility function to optimize, namely, that of the entire agent ensemble. Since our present work does not address the issue of how are the coalitional payoffs to be split up among the individual agents [54], the solution that we propose for the coalition formation and the coalition-to-task mapping problems is directly applicable to cooperative and collaborative distributed problem solving domains. We will further reflect on these issues in Chapter 5, after we present our maximal clique based distributed coalition formation algorithm; for more details, we also refer the reader to our work in [50, 53].

Another important distinction insofar as the individual agents and the relationships among them are concerned, is the one between *homogeneous* and *heterogeneous* agents. Homogeneous MAS are those where all agents are identical, or at least very similar, to each other in terms of their roles, capabilities, and other properties. Heterogeneous MAS are characterized by the presence of and coordination among the agents that, typically, significantly differ in terms of their roles, properties, and/or capabilities. Since we are primarily interested in studying multi-agent coordination in large-scale *peer-to-peer* (P2P) systems, we shall focus on *homogeneous* MAS. In particular, our *Maximal*

Clique based Distributed Coalition Formation (MCDCF) algorithm in Chapter 4 treats all agents equally: it is a genuinely P2P, leaderless coordination mechanism. However, our agents may differ in terms of their *capability vectors* [43, 50] – but these differences in capabilities do not affect the basic fact that, during the coalition formation process itself, all agents play the same role and, moreover, execute the same code.[1]

Insofar as the nature of the coalition formation strategies and the allowable coalition structures are concerned, the following distinctions are made in the MAS literature on multi-agent coordination via coalition formation (e.g., [12]):

– on-line (dynamic) vs. off-line (static) coalition formation;

– centralized vs. distributed coalition formation;

– overlapping vs. non-overlapping coalitions.

In static coalition formation, the agents form coalitions *prior* to deployment; that is, first the agents reach the consensus on the coalition structure (or, in the centralized setting, an outside central authority forms these coalitions), and then the already formed coalitions are deployed, i.e., mapped to various tasks, or perhaps sets or sequences of tasks [12].

The more challenging scenario is when the agents have to form the coalitions *dynamically* at the time of deployment – especially when the coalition formation is carried out in a decentralized manner, and thus this process itself requires reaching *distributed consensus* among the agents [20, 48]. The additional challenges in this, dynamic decentralized coalition formation scenario arise when the agents are severely limited in terms of their resources, such as the computational time, communication bandwidth, or available energy for transmitting messages. Our MCDCF algorithm is designed for the purpose of dynamic, fully decentralized and highly resource-aware multi-agent coalition formation; that is, we make a modest attempt to address agent coordination via coalition formation in its most difficult setting with respect to the above criteria.

Insofar as the allowable coalition structures, we consider in this work *non-overlapping* coalitions. In particular, viewed as a (distributed) graph algorithm, our MCDCF approach to coalition formation is designed to *partition* (rather than just *cover*) the set of vertices of the underlying graph. We discuss the relevant algorithmic and computational complexity theoretic aspects of the

[1] For the pseudo-code for MCDCF algorithm, see *Appendix.*

set partitioning and set covering problems in Chapter 5.

One of the emphases of our work on agent coordination via coalition formation as summarized in this thesis, as well as on modeling and analyzing large-scale MAS in general (see, e.g., [50]), has been on the limited communication and computational resources of individual agents. In order to have a chance of being practically useful, an algorithm for coordinating, to give some examples, thousands of micro-UAVs, mobile software agents or smart sensors, has to respect the intrinsic knowledge and resource limitations of such agents. It also has to scale well with the problem size (including, but generally not limited to, the number of agents and the number of tasks in the system), even if this is at the expense of optimality. In short, *scalable, resource efficient* and *fully distributed* algorithms for coalition formation are desperately needed in a number of multi-agent application domains. Such algorithms should use a few communication rounds among the agents, place a modest computational burden on each agent, be aware of limited communication range and bandwidth, and ensure that a distributed consensus among the agents on who is forming a coalition with whom is effectively, reliably and efficiently reached.

We propose in this work a distributed coalition formation algorithm based on the idea that, in peer-to-peer (in particular, *leaderless*) MAS, an agent would prefer to form a coalition with those agents that it can communicate with directly, and, moreover, where every member of such a potential coalition can communicate with any other member directly. That is, the preferable coalitions are *(maximal) cliques.* It is well-known that finding a maximal clique in an arbitrary graph is **NP**-hard in the centralized setting [8, 11]. This implies the computational hardness that, in general, each node faces when trying to determine the maximal clique(s) it belongs to. However, if the degree of a node is sufficiently small, then finding all maximal cliques this node belongs to may become computationally feasible. If one cannot guarantee that, or *a priori* does not know if, each node in a given underlying MAS interconnection topology is of a small degree, then one has to impose additional constraints in order to ensure that the agents are not attempting to solve an intractable problem.

We describe our distributed maximal clique based coalition formation algorithm in some detail in Chapter 4. This algorithm, or its appropriately adjusted variants, can be fine-tuned to meet the coordination needs of a great variety of MAS, such as the following:

– various cooperative *distributed problem solving (DPS)* agents characterized by non-trivially bounded computational and communication resources (e.g., [21, 22, 43, 47]);

– different kinds of self-interested, strictly competing or competing-and-cooperating agents [34, 42] where the concepts, paradigms and tools from the *N-person game theory* have found many applications; and, more generally,

– various bounded-resource, imperfect-knowledge autonomous agents that are acting in complex dynamic environments [27, 50, 62], where these environments are only *partially accessible* to any agent; such autonomous agents are thus characterized by the property of *bounded rationality* [46].

As we have already pointed out, our current focus is on the cooperative MAS such that the individual agents are characterized by the very limited communication and computational resources, as well as bounded rationality insofar as each agent's knowledge about the world is concerned. Extending the existing framework to competitive MAS domains is left for the future work.

2.2 On Locally Constrained Collaborative MAS Environments

Large ensembles of autonomous agents provide an important class of examples where the agents' capability to coordinate and, in particular, self-organize into groups or coalitions, is often of utmost importance for such systems to be able to accomplish their tasks.

One can distinguish between two general, broad classes of such autonomous agents. One is the class of agents deployed in the context of *distributed problem solving* (DPS). The agents encountered in the DPS contexts typically share their goal(s). For instance, DPS agents most often have a joint utility function that they all wish to maximize as a team, without any regard to (or perhaps even a notion of) individual payoffs. This joint utility or, more generally, the goal or set of goals assigned to DPS agents, is usually provided by their designer. However, it may not be always feasible – or even possible – that the designer explicitly specify in detail how are the agents to divide-and-conquer their tasks and resources, how are they to form coalitions and elect leaders of those coalitions, etc. Due to the scalability issues, incomplete *a priori* knowledge of the environments these agents may encounter, and possibly other considerations, instead of "hard-wiring" into his DPS agents explicitly how are the agents to be coordinated, the system designer may choose to enable the autonomous agents with only the very basic coordination primitives, and leave to the agents to

self-organize and coordinate without any central control or other forms of outside interference, and to do so dynamically, as required by changes in their environment. Hence, in many situations the DPS agents may be required to be able to effectively form groups or coalitions in a fully distributed manner. An example are heterogeneous human or robotic agents deployed in a hard real-time, no-outside-assistance rescue mission in a disaster area: while such agents are collaborative as there is an overarching, global goal at the system level, each agent also has its own preferences and a local, individual notion of goal(s).

The second basic type of agents, the *self-interested agents*, are a kind of agents that do not share their goals (and, indeed, need not share their designer). In contrast to the DPS agents, each self-interested agent has its own agenda (e.g., an individual utility function it is striving to maximize), and no altruistic incentives to cooperate with other agents. However, even such self-interested, individual utility driven agents, may nonetheless still need to cooperatively coordinate and collaborate with each other in many situations. One class of examples are those agents (such as, e.g., autonomous unmanned vehicles) that, if they do not coordinate in order to resolve possible conflicts, they risk mutual annihilation. Another class of examples are the agents with bounded resources: individually, an agent may lack resources to accomplish any of its desired tasks – yet, if this agent forms a coalition with one or more other agents, the combined resources and joint effort of all agents in such a coalition may provide utility benefits to everyone involved. An example of such coordination via coalition formation among the selfish agents are coalitions among corporations in, for instance, airline or automobile industry.

Of an increasing importance and ubiquity, especially in the collaborative large-scale MAS domains, are those autonomous agents that, while collaborative, are also *locally constrained* in various ways. These constraints on each agent's local knowledge about the world (that have *bounded rationality* [46] as a consequence), as well as on the available local computational resources, communication bandwidth, memory storage, and/or energy resources, limit the agent's ability insofar as with whom it can interact, how much information can be exchanged in such an interaction, and the like. A paradigmatic example of a severely locally constrained collaborative large-scale MAS are the sensor networks of the near future that may include $10^3 - 10^4$ or more *smart sensors.* We briefly discuss the sensor network example in subsection 2.2.1.

Both coalition formation and task selection are examples of an autonomous agent's *decision making.* The mathematical framework for modeling and analysis of how an agent makes decisions on what action to select, to which other agents to propose forming a coalition, and similar problems is provided by *decision theory* [4, 30]. Decision theory is defined in [27] as *a means of analyzing which of a (typically finite) set of available options should be selected when it is uncertain what exactly will be the outcome of taking that option.* The goal of decision theory is to provide an agent with the means of selecting *the best* available option, where "the best" usually means the option that maximizes *the expected payoff* [27]. Thus, probability theory and utility theory are the necessary mathematical ingredients of decision theory. Since the crucial issue of designing autonomous agents is how to provide those agents with the ability to select the best action from a set of possible actions, and since, in general, the state of the world depends on factors other than the actions taken by an individual agent, decision theory provides an appropriate and rather general mathematical framework for designing autonomous agents and analyzing their behavior [27].

The emphasis in decision theory is on the decision making of a *single* autonomous agent. This agent observes, and acts in response to, the changes in its environment. In the context of MAS, this environment also includes other agents. *Game theory* [24] is a close relative of decision theory that studies interactions among several decision making entities that are, in general, autonomous and self-interested. In fact, decision theory can be viewed as a special case of game theory, namely, the one that studies the so-called *games against nature* where a single rational agent plays a formal game against an environment that is not self-interested or rational, but, rather, typically (appears to) act randomly [27].

Insofar as the large-scale MAS are concerned, the relevant subarea within game and decision theories is the *n-person game theory* [19, 14]. For the DPS collaborative MAS domains, the relevant type of n-person games are the *cooperative games* [24, 32]. While in the strictly DPS domains an individual agent has no notion of individual utility or payoff, in those DPS domains that are characterized by the aforementioned properties of *bounded rationality* and bounded computational and communication resources, each agent will, in general, still have some local, individual notion of preferences about those aspects of the environment that are accessible to that agent. Hence, the concepts and models from cooperative n-person game theory do apply. In particular, collabo-

rative coalition formation among distributed problem solving agents has been studied in the MAS literature from a cooperative n-person game theory perspective for a relatively long time (e.g., [42]).

An important game-theoretic consideration in the context of coalition formation in n-person games has to do with the nature of the environment, and, in particular, whether the environment is *super-additive, sub-additive,* or neither. The nature of the environment and its implications in the MAS context are formalized and discussed, e.g., in [38, 42]. Super-additive environments typically characterize collaborative domains. In such environments, assuming there are no obstacles or constraints to arbitrary coalitions being formed, the best coalition for all agents in the system is the *grand coalition* that includes all agents. However, is such grand coalition possible, or even desirable, in a scenario where there are, say, 10^4 or more agents in the system? In most such massive MAS, however, (i) an agent can efficiently and reliably communicate with typically only a handful of other agents, and (ii) each basic task would usually require a coordinated effort of only a relatively few agents anyway. It is precisely the broad class of large-scale MAS applications that satisfy properties (i) and (ii) above that we have in mind, and that we would like to provide with a novel coalition formation based coordination strategy.

It has been observed in the MAS literature that the classical game theory with its various notions of optimal strategies, equilibria, and optimal coalitions is not particularly algorithmic-minded [43]. In the classical game theory, little or no attention is paid to the computational costs associated with determining appropriate coalition structures, game equilibria, or other notions of a game solution. Moreover, if a prescription of *how* to achieve the desired coalition structure or game equilibrium is offered at all, the proposed solutions are tacitly assumed to be computed off-line and in a centralized manner. In particular, the classical game theory's procedures for determining and finding the most desirable coalitions are not well suited for the large-scale, *massive* multi-agent systems, where the solutions to a given n-person game problem has to be found in an online and distributed manner, in which the agents have to respect severe constraints on their computational and communication resources, and where the time available for any sort of an agent's deliberation (such as, e.g., computing appropriate game equilibria, evaluating the "goodness" of all candidate coalitions, etc.) is typically rather limited [50, 53].

Upon some reflection on the first version of our distributed coalition formation algorithm [51],

where the entire emphasis was on efficiently forming maximal-clique-like coalitions whose sizes would never be allowed to exceed a pre-specified threshold, we have realized that we had tacitly assumed what we shall call *locally super-additive environments.* A locally super-additive agent environment is one where the traditional, game-theoretic super-additivity holds but only subject to the appropriate *locality constraints.* In [51], the main constraints in the maximal clique context are stemming from the communication network topology that restricts which agents can communicate with each other *directly* (as opposed to via multiple hops). We assume this general framework in the present work, as well. However, we shall also discuss some coalition quality criteria going beyond the mere communication network topology properties. A more detailed discussion on some possible alternative criteria for a coalition's quality will follow in Chapter 5; we also refer the interested reader to [53, 54].

We have been attempting to address the issue of which agents will select which tasks to serve [49, 50], or, alternatively, which coalitions of agents will form in order to serve some particular task or set of tasks [54]. To that end, what is critical are an agent's or agent coalition's *capabilities* that enable the agent(s) to meet the tasks' resource demands. Each agent is assumed to possess a vector of capabilities or available resources; likewise, each task has a tuple of resource requirements. An agent coalition can serve a particular task if and only if each component (i.e., individual capability or resource) of its joint vector of capabilities is greater than or equal to the corresponding resource requirement vector entry of that task. An agent can belong to *only one* coalition at a time, and work with its fellow coalition members on one task at a time. That is, we are (at present) interested in coalition formation solely for the purpose of the coalition-to-task mapping, but we are not concerned with *how* are then these agent coalitions exactly going to perform the tasks they have been mapped to. For these reasons, the framework as described in [50] and outlined at the beginning of Chapter 4 (cf. in section 4.1) suffices for our purposes.

To summarize, for the present purposes, the MAS environments will be assumed *locally super-additive* in the sense discussed above. We shall return to these game-theoretic issues pertaining to the nature of the MAS environment in Chapter 5.

2.2.1 Some Examples of The Candidate Large-Scale MAS Applications

One may ask, why would the maximal-clique based approach that we propose be promising for the very large scale (or *massive*) multi-agent systems (*MMAS*) that may be made of anywhere from thousands up to millions of agents? The underlying network of such MMAS is bound to be very large, thus rendering even many typically feasible (i.e., polynomial-time in the number of agents) graph algorithms obsolete due to their prohibitive cost – let alone allowing for the distributed coordination strategies that are based on the graph algorithms which are, in the centralized setting, known to be **NP**-hard in general.

However, there is one critical observation that saves the day of our approach: even if the underlying graph is indeed very large, in many important MMAS applications this graph will also tend to be *very sparse.* That is, a typical node in such a graph will tend to have only a handful of neighbors. Therefore, a distributed algorithm where agents reason and communicate strictly locally, where no *flooding* of the network is ever performed, and where each agent needs to store and work with only the data pertaining to its near-by agents, can still be designed to be sufficiently efficient. In other words, it is the underlying communication network's *density,* rather than its absolute size, that is the critical scalability parameter in the context of decentralized coalition formation in MMAS.

Some examples of the engineering, socio-technical and socio-economic systems and infrastructures that can be modeled as MMAS and that are also characterized by the aforementioned *sparseness* of the underlying network topology, include the following:

(i) *Large-scale* ($10^3 - 10^4$) *ensembles of micro-UAVs* or other similar autonomous unmanned vehicles deployed, for example, in a surveillance or a search-and-rescue mission over a sizable geographic area. Unlike the scenarios where dozens of *macro-UAVs* are deployed, where a centralized control and/or one human operator per UAV are affordable and perhaps the most efficient and robust way of deployment, in a very large scale system of autonomous *micro-UAVs* no central control is feasible or even possible, and the run-time human intervention is either minimal or nonexistent. Such micro-UAV ensembles need to be able to coordinate, self-organize, and self-adapt to the changing environments in a truly decentralized, dynamic and autonomous manner. Specifically, any coordination among these UAVs, such as the coalition formation or the coalition leader

election, necessarily has to be fully distributed. For more on the design and simulation challenges of such large-scale ensembles of micro-UAVs, we refer the reader to the OSL group's work on the DARPA-funded TASK project, and the related publications [15, 16, 49, 50].

(ii) *Smart sensor networks* [18] that include anywhere from thousands to millions of tiny sensors, each of which could be only a few millimeters in size, and of a rather limited computational power and communication range and bandwidth. In particular, the RAM memory of such smart sensors is currently (year 2004–2005) typically of the order of Kilobytes, the flash memory range is about 1 MB, the communication bandwidth is $10^2 - 10^3$ Kilobits/second, and a typical battery life span is anywhere from a few hours up to a week.

Due to their energy limitations, such smart sensors usually communicate via *local broadcasts* with very limited ranges. The main "communication mode" of the agents in our algorithm in Chapter 4 are precisely the local broadcasts. Due to their small memory capacities and low power consumption requirements, smart sensors need to simultaneously minimize both the amount of local processing, and the amount of communication – the latter pertaining to both how often they communicate messages to other sensors, and how much data is communicated in each message. Sensor networks, although of more modest sizes, have already been used as an example to which *dynamic distributed constraint satisfaction/optimization* formalisms can be fruitfully applied (e.g., [22]). We shall briefly discuss distributed constraint satisfaction (DCS) as a general modeling framework for distributed coalition formation in Chapter 3.

(iii) *Social networks,* and, in particular, various variants of the *small-world networks* where, in addition to the strictly local connectivity in the communication network topology, a relatively few random long-range connections are also present [58, 59]. A typical node in such a social network will have only a handful of neighbors it can directly communicate with, and, moreover, most of these neighbors in the network of social contacts will also tend to be the neighbors in the usual, physical proximity sense.

(iv) Various *socio-technical infrastructures,* such as, e.g., the transportation systems, the power grids, etc. For a simple concrete example, consider a car driver participating in the traffic in a large city. While there may be millions of such drivers on the road at the same time, the decision-

making of a driver-agent is based, for the most part[2], on a few properties of the agent's local environment, i.e., the relevant actions of a typically small number of near-by agents. Likewise, her own actions will usually directly affect only a handful of other agents in the system – for instance, the driver right behind her, or the pedestrian trying to cross the street right ahead of her. Thus, the underlying network of agents, while of a very large size, will be in general quite sparse, and a typical node in such a network will be adjacent to only a small number of other nodes.

An ambitious project on realistic modeling and simulation of very large-scale infrastructures (such as the traffic system of a big city), called *TRANSIMS,* is described at the URL [55] and in the documents found therein.

Insofar as the practical *grounding* of our approach to dynamic decentralized coalition formation, our MCDCF algorithm was originally chiefly motivated by the agent characteristics of the MAS applications (i) and (ii) above; we refer the reader to our early work [50, 51] for more details.

[2]There are exceptions of course; for instance, when an imminently approaching tornado is announced on the radio.

CHAPTER 3

RELATED WORK

A variety of coalition formation mechanisms have been proposed in the MAS literature in the context of both the DPS agents that are all sharing the same goal (as in, for example, [21, 22, 43, 44, 47]), and the self-interested agents where each agent has its own individual agenda (as in, e.g., [42, 66]). In particular, the problems of distributed task and resource allocation, and the issue of how is task or resource allocation coupled to what coalition structures are most desirable in a given scenario, are intimately related to the problem of distributed coalition formation itself [21, 22, 42, 43, 44].

A considerable body of the MAS literature casts distributed task and resource allocation into the general distributed constraint satisfaction and/or optimization (DCS/DCO) terms (e.g., [21, 22, 23, 42]). The importance of DCS for MAS in general is discussed in [65]. In order to provide a broader context for the problem addressed in this thesis, we will briefly (and very informally) review the most fundamental DCS and related concepts in the next section. We will then outline some existing research approaches to the central issue that this thesis is attempting to address, namely, the problem of dynamic, distributed task allocation via coalition formation in multi-agent systems, and compare and contrast those approaches with our own approach as described in Chapter 4 and references [51, 53, 54].

3.1 Distributed Constraint Satisfaction and MAS

A *(centralized) constraints satisfaction problem* (CSP) has the goal of finding a consistent assignment of values to a set of variables [65]. A CSP is defined by a set of variables, each associated with

a finite domain, and a set of constraints on the allowable values of those variables. A solution to a CSP instance is a value assignment to the variables which satisfies all the constraints [21]. A wide variety of problems in combinatorics, combinatorial optimization, traditional computer science and AI can be formulated as CSPs. One classical example is the problem of placing n chess queens on an $n \times n$ chessboard, so that no two queens attack each other. Another example is the *CNF satisfiability* problem: given a Boolean formula in the conjunctive normal form, determine if there exists a satisfying truth assignment to the variables in the formula. A CSP phrasing of the same problem is, given a set of clauses, find an assignment to the Boolean variables appearing in those clauses so that each clause is satisfied.

A *distributed constraint satisfaction problem* (DCSP) is a CSP in which the variables and the constraints are distributed among multiple agents, and where the problem has to be solved in a decentralized manner. DCSP in MAS can be informally defined as follows: given multiple agents in a shared environment, and some set of (in general) both local and inter-agent constraints, find a consistent combination of agent actions so that all local and inter-agent constraints are satisfied [65]. Again, a broad variety of problems in distributed AI can be formalized as DCSPs. A formal mathematical framework for DCSPs, as well as the main algorithmic techniques for solving these problems, can be found in [65].

A *constraint optimization problem* (that also can be either centralized or distributed) is a problem where, in addition to a set of constraints that simultaneously need to be satisfied, there is also an *objective function* that needs to be either minimized or maximized. For instance, the *Hamilton cycle* problem [11] on graphs can be thought of as a constraint satisfaction problem: "Given an undirected graph, assign to each node its successor, so that there is an edge in the graph from a given node to its successor, so that no two nodes have the same successor node, and so that every node is someone's successor". The standard optimization version of the *Travelling Salesman Problem* TSP problem [11], on the other hand, can be naturally viewed as a constraint optimization problem: "Among all Hamilton cycles in a weighted graph, find one of the lowest cost."

Our approach to coalition formation in MAS is based on determining (maximal) cliques in a graph in a distributed manner. Partitioning a set of vertices of a given graph so that each resulting group of vertices is a (maximal) clique is a natural CSP. The same applies to the slightly modified

problem: given a vertex of a graph, find a maximal clique that this vertex belongs to. In our MCDCF algorithm in Chapter 4, each agent is, in a sense, attempting to solve this latter version of the maximal clique problem. However, the agents are acting concurrently, and what one agent does, in general, can affect other agents. Each agent has only a local, partial view of the graph; however, agents can communicate with each other and exchange information. Thus our problem framework fits into the general DCSP setting. Moreover, our problem is also dynamic: once some of the agents form a coalition, the rest of the agents better take note of that, because those agents that have already joined a coalition are no longer available for further coalition formation with other agents.

In this thesis, therefore, we will approach coalition formation in MAS as a *dynamic distributed constraint satisfaction* problem *with communication* [12]. While dynamic DCSPs have been studied and formalized in the MAS literature (e.g., [12, 21, 22]), we will argue in the next section that both our problem formulation, and the MCDCF algorithm as a proposed heuristic solution to that problem, are genuinely different from the existing frameworks in several important respects.

3.2 Task Allocation via Coalition Formation in MAS

DCS and DCO formalisms have been applied to a great variety of problem domains within distributed AI. Some of those DAI and MAS problem domains are *multistage negotiation* [9], *argumentation* [17], *distributed software design* [28], and *team robotics* [40]. Perhaps the most frequent problem domains where the DCS- and/or DCO-based formalisms and techniques have been utilized, however, are the distributed task allocation and the distributed resource allocation domains [21, 22, 23, 34, 42, 43, 47, 57, 66].

Of a particular relevance to our work in this thesis are the references [43, 44] and [21, 22]. We shall briefly outline the main contributions of each, as well as where our approach differs from the approaches taken in those two lines of work.

Modi et al. in [22] offer perhaps the most complete formalization and taxonomy of various distributed resource and/or task allocation problems in cooperative MAS domains, as well as the general mappings from those problems to the appropriate types of *(dynamic) distributed constraint satisfaction* (or, when appropriate, *optimization*) problems. The authors' ambitious agenda was

to be able to classify most distributed task/resource allocation problem formulations found in the MAS literature, and then to find the appropriate variant of DCS or DCO for each particular version of the distributed allocation problem.

At a first glance, our entire problem framework, whose preliminary version can be found in [51], and that has been subsequently elaborated in [53, 54] as well as in Chapter 4 of this thesis, appears to be just a special case that fits well in Modi et al.'s distributed allocation problem taxonomy as elucidated in [21, 22]. Therefore, it would seem that all one needs to do is to use their formal mappings to the appropriate DCS/DCO variants, and then obtain her problem's solution "off the shelf". However, we argue that the framework of Modi et al. [21, 22], general as it may be, is not quite suitable for classifying our problem formulation given in Chapter 4. There are at least two reasons why, that we briefly discuss below.

One, an agent's capabilities (called *operations* in [21, 22]) are *quantified* in our framework, as opposed to being Boolean-valued in a sense that an agent either has a particular capability, or does not have it. Therefore, instead of an agent being characterized by a discrete set of operations or capabilities that either are or are not present, but have no *numerical value* assigned to them that would capture to what extent an agent possesses a particular capability, in our setting each agent has its vector (that is, an ordered tuple) of nonnegative real-valued capabilities. This difference has considerable implications insofar as the partial ordering of the individual agents and agent coalitions with respect to their capabilities is concerned. Another, related difference between the modeling framework in [21, 22] and our work, is the underlying model of tasks, and how the tasks' resource or capability requirements relate to agents and agent coalitions. Because of these different assumptions, only the very special case of our MCDCF problem formulation as described in Chapter 4, and originally addressed in [51], can fit into the Dynamics DCS framework of Modi et al. in [21, 22].

Two, the agents in [22] are strictly cooperative, share the same goals, and, as such, are not endowed with any notion of individual utilities or preferences. While our MCDCF algorithm is primarily envisioned for the cooperative MAS domains, as well, one of our main assumptions is that, due to a large scale of the system, and a high dynamism and unpredictability of the changes in the environment, no shared or global knowledge about the environment is maintained. In particular,

each agent has its own individual preferences over (the locally accessible aspects of) the states of the world. That is, even though the agents in our problem formulation are not, strictly speaking, self-interested (at least not in the usual sense of that term as defined in economics), these agents still have individual preferences of how would they like the world to be. We emphasize that it is critical for our MCDCF algorithm in Chapter 4 that each individual agent has a preference partial order defined on the set of all its candidate coalitions.

The task allocation modeling framework in [21, 22] deserves credit for identifying the shortcomings of the classical DCSP framework related to DCSP not capturing the *dynamic* aspects of most MAS environments. We also try to address the issue of dynamic changes in MAS environments in this thesis, but, unlike Modi et al., we do not provide a formal mathematical framework for capturing the dynamics. In particular, our MCDCF approach can be combined with the simple *predicate-satisfaction* model of [21] whose purpose is to capture if a particular task (and the constraints it imposes onto the agents) is *active* at a particular time. We leave formally extending MCDCF to a framework with *explicitly dynamic* constraints as in [21, 22] for a future work. However, we point out that some aspects of dynamic changes in MAS environments are already captured in MCDCF, as will be discussed in Chapters 4 and 5.

The problem formulation that is the closest to ours is that of Shehory and Kraus in [43, 44]. Both the framework in [43, 44] and our own in Chapter 4 of this thesis can be viewed as addressing distributed task allocation in collaborative MAS via an appropriate set partitioning based distributed coalition formation. Moreover, the similarities go well beyond the general problem definition and approach to solving it. In particular, the simple models of both agents with their capability vectors, and tasks with their vectors of resource or capability requirements, are essentially identical in these two frameworks. The close similarity between the two models extends to the notion of coalitions, as well as what it means for a coalition of agents to be able to service a task. Likewise, both frameworks assume *constrained super-additive environments*. A multi-agent environment is super-additive if, given two different coalitions of agents, it is *always* beneficial that those two coalitions merge together into a single coalition. A constrained super-additive environment is one where, while super-additivity generally holds, it may be prohibitively costly (or forbidden on some other grounds) to always merge any two coalitions that, in principle, could be merged. In our case,

the constraints on the super-additivity of the multi-agent environment are explicitly attributed to the locality properties of the agents' communication network topology, and the cost of (multi-hop) communication. That is why we refer to the environment in our coalition formation for task allocation setting [53, 54] as to *locally constrained super-additive*. As a consequence of constrained super-additivity, both in [43] and in our work, the infeasibility of the general problem of searching for an optimal coalition structure in cooperative domains is avoided by imposing an upper bound on the allowed number of agents in any coalition [39].

However, there are some differences in the two problem formulations (that of Shehory and Kraus, and our own), as well. For instance, in [43] an agent's capabilities are transferable to other agents, whereas we do not allow such transfers. Also, in our framework (see Chapter 4) the agents evaluate tasks directly, and then rank by preference their candidate coalitions with respect to this evaluation of tasks, rather than assigning monetary units to the capabilities themselves, as in [43].

The most important distinction between the approach in [43] on one hand, and our approach in [51, 53, 54] and this thesis, on the other, is in the algorithms themselves, i.e., in how agents go about forming coalitions. In [43], the initial *candidate coalitions* are the trivial ones: each agent is initially a single-member candidate coalition. Then the agents start negotiating and, step by step, forming coalitions that are *growing* in size, until some threshold in maximal allowed coalition size is reached. In contrast, our MCDCF algorithm in Chapter 4 starts with each agent assuming that its entire neighborhood in the underlying graph is a candidate coalition. In the subsequent rounds, an agent that has not yet reached a consensus on what coalition to form with some of the other agents, is going to consider candidate coalitions that are, in the sense outlined below, *smaller* than the previously considered candidates, instead of *larger* as in [43, 44].

To be able to contrast the two algorithms more formally, we need to consider the lattice of subsets of the set of all agents, $\{A_1, ..., A_n\}$, where the partial order on this lattice is provided by the set inclusion. In [43], each agent during its quest for a coalition to form starts from the singleton candidate coalition, and then climbs up the lattice of subsets, until it reaches a subset that becomes the coalition this agent is joining. In contrast, in our MCDCF heuristic, each agent starts with the maximal possible candidate coalition – that made of all of its neighbors, including

itself.[1] Subsequently, an agent can only go either *down* the lattice of subsets, or else *across* this lattice. In the first case, if the agent's coalition proposal C has not been accepted, in the next round the agent considers some other candidate coalition, C', such that $C' \subset C$ holds (and the set inclusion is proper). In the latter case, the next candidate coalition C' is neither a subset nor a superset of C. The reader will have to wait until Chapter 4 for the full description of our algorithm and all the missing details, or can consult reference [54].

This difference between [43] and our approach in terms of an agent's search process for its "right" coalition may have profound implications for the resulting coalition structure, as well as the two algorithms' costs (and especially the rather different trade-offs between the local computation and communication). Thus, while the two problem formulations are very similar, the proposed solutions are considerably different. Which approach to task formation via coalition formation is more appropriate or efficient ultimately depends on the properties of the application domain.

[1] This is the maximal possible candidate coalition, since, in the DCS terms, our MCDCF problem framework in Chapter 4 requires that each agent has to respect a constraint that the coalition the agent eventually joins ought to be a clique in the underlying communication network.

CHAPTER 4

THE MAXIMAL CLIQUE-BASED DISTRIBUTED COALITION FORMATION ALGORITHM

We present in this Chapter our maximal clique based distributed coalition formation (MCDCF) algorithm. The purpose of MCDCF is to enable collaborative agents in a large-scale MAS to form coalitions of modest sizes in a distributed, peer-to-peer manner. This algorithm is designed for ensembles of cooperative autonomous agents to repeatedly use (as needed) as a basic coordination subroutine. It is fully decentralized and local, very resource-aware, and, under certain assumptions to be discussed in the sequel, scalable and efficient.

The proposed algorithm is a graph algorithm. The underlying undirected graph[1] captures the communication (ad hoc) network topology among the agents, as follows. Each agent is a node in the graph. The necessary requirement for an edge between two nodes to exist is that the two nodes be able to *directly* communicate with one another. That is, an unordered pair of nodes $\{A, B\}$ is an edge of the underlying graph if and only if A can communicate messages to B, or B can communicate messages to A, or both. We point out, however, that this definition of the graph edges can be made tighter by imposing additional requirements, such as, e.g., that the two nodes, if they are to be connected by an edge, represent agents that are also compatible in terms of their capabilities, such that each provides some resource(s) that the other agent needs, and/or the like.

[1]For simplicity, we assume the graph is undirected, even though the communication from one node to another is clearly directional, and the communication links need not be symmetric. However, since the nodes eventually need to reach a mutual consensus, which requires that *all* members of a coalition agree to the same coalition and *notify* all other coalition members of the agreement, the assumption about the edge (non-)directionality is inconsequential, in a sense that only those coalitions that indeed are cliques in the corresponding *directed graph* will ever be agreed upon by the agents.

The basic idea behind MCDCF is to efficiently and in a fully decentralized manner partition this graph into (preferably, *maximal*) *cliques* of nodes. These maximal cliques would usually also need to satisfy some additional criteria in order to form temporary coalitions of desired quality. These coalitions are then maintained until they are no longer preferred by the agents – that is, when they are no longer sufficiently useful or, in case of highly dynamic MAS, no longer meaningful. For instance, the coalitions should be transformed or, if appropriate, dissolved when the interconnection topology of the underlying graph (that is, the inter-agent communication network) considerably changes, either due to the agents' mobility, or because perhaps some of the old links have died out whereas some new, different links have formed, and the like. Another possible reason to abandon the existing coalition structure is when the agents determine that the coalitions have accomplished the set of tasks that these coalitions were formed to address. Thus, in an actual MAS application, the proposed coalition formation algorithm may need to be invoked a number of times as a coordination subroutine.

The rest of this Chapter is organized as follows. In section 4.1, we first state the problem addressed and briefly discuss the main assumptions made in our MCDCF algorithm. We then describe the algorithm itself in some detail in section 4.2, followed by a simple example of how the algorithm works in section 4.3.

4.1 Problem Formulation and Main Assumptions

Before we formally state the problem that our MCDCF algorithm is proposed to solve, we first provide an overview of the necessary general concepts and assumptions. We assume a multi-agent, multi-task dynamic environment. The agents have certain capabilities that (may) enable them to service the tasks. Similarly, the tasks have certain resource or capability requirements, so that no agent or coalition of agents whose (joint) capabilities do not meet a particular task's resource requirements can serve that task. Tasks are assumed *mutually independent*. In addition, each task is of a certain value to an agent. For simplicity, given a task, we will assume that all agents ascribe the same value to that task. A slightly modified version of our MCDCF algorithm actually works even when each agent has its individual view of a task's value, so that different agents may evaluate the same task differently. However, in this thesis we will not specifically concern ourselves with

this, more general framework.

Agents are assumed capable of communicating, negotiating and making agreements with each other [61]. Communication is accomplished via exchanging messages. This communication is not free: an agent has to spend time and effort in order to send and receive messages.

Thus far, our model is identical to that in [43]. An important difference that we introduce with respect to the model in [43] is that we assume that an agent's resources are not transferable to other agents. Thus, the only way for an agent A_i to use the internal resources of agent A_j for the purpose of servicing some task is that A_i and A_j join the same coalition, and then jointly attack that task.

4.1.1 Constraint Optimization Problem Formulation

We now mathematically formalize the concepts introduced in this Chapter thus far. There is a set of n autonomous agents, $\{A_1, A_2, ..., A_n\}$. Each agent A_i has a vector of real-valued, nonnegative internal resources or capabilities, $R(i) = [R_1(i), ..., R_s(i)]$. We also assume that there is some finite number of tasks in the environment. Each task requires a certain nonnegative amount of each of the s resources in order to be serviced.

At any time, each agent is aware of some subset of the currently active (that is, serviceable) tasks. A task T_j is completely characterized by the following two properties: (i) its nonnegative value, $V(j)$, and (ii) its vector of resource requirements, $R^j = [R_1^j, R_2^j, ..., R_s^j]$.

An agent coalition is any nonempty subset of the set of agents $\{A_1, ..., A_n\}$. A coalition $C_I = \{A_i : i \in I\}$ can service (that is, complete) a task T_j if and only if, for each individual resource R_k, $1 \leq k \leq s$, the following holds:

$$\sum_{i \in I} R_k(i) \;\geq\; R_k^j \tag{4.1}$$

Let's consider the "one-shot" problem where each agent just needs to pick a single task only once. In that case, there is no need for any form of long-term planning. Assuming the sole purpose of an agent's capabilities is to serve the selected task, so that the agents have no incentives to hold on to and thus try to "save" their capabilities, a self-interested rational agent that is not allowed to cooperate with other agents would simply greedily pick the most valuable task for which this

agent's capabilities are sufficient. That is, an individual agent's goal can be expressed as a simple *constraint optimization* problem

$$Max_j\ V(j) \quad subject\ to \quad R_k(agent) \geq R_k^j \quad for\ \forall k : 1 \leq k \leq s \tag{4.2}$$

The above scenario would hold, for instance, in *strictly competitive* MAS domains, or in those domains where agents are not capable of communicating and collaborating with one another. Let us now assume that the agents are capable of communication and are allowed to collaborate, and hence that they can form coalitions. If the task values are not divisible but, instead, each agent participating in a coalition that services a task receives the full value of that task as its reward, then the optimization problem that each such self-interested agent $A_\star$ faces can be phrased as follows:

Find coalition $C \subseteq \{A_1, ..., A_n\}$ *such that* $A_\star \in C$ *and*

$$Max_j\ V(j) \quad subject\ to \quad \sum_{A_i \in C} R_k(i) \geq R_k^j \quad for\ \forall k : 1 \leq k \leq s \tag{4.3}$$

is maximized.

Notice that the overall optimization problem strives to optimize over the set of all possible coalitions $C \subseteq \{A_1, ..., A_n\}$ such that $A_\star \in C$. If there are no restrictions on the coalitions that agent $A_\star$ is allowed to consider, then one globally optimal solution is always going to be *the grand coalition* made of *all* agents: $C^{optimal} = \{A_1, ..., A_n\}$. This was to be expected given that, under all assumptions that we have made, we know from elementary n-person game theory that the underlying environment in the above problem is (unrestrictedly) super-additive; we also refer the reader to the related discussion in Chapter 2.

As discussed at some length in Chapter 2, in a typical large-scale MAS domain, the coalitions that the agents would desire to form are going to be constrained in various ways, and, thus, will typically be of relatively small sizes. Explicit constraints on the allowable coalitions, as appropriate for a given problem domain, can be readily added to the optimization problem formulation (4.3). In particular, if the agents' communication ranges are bounded, thereby inducing a communication network topology on this ensemble of agents, and furthermore if each agent only considers those candidate coalitions that are *cliques* with respect to that network topology, we arrive at the desired

problem formulation from the perspective of a single self-interested agent:

Find coalition $C \subseteq \{A_1, ..., A_n\}$ *such that* $A_\star \in C$ *and* $Max_j\ V(j)$ *subject to*

$$\forall k\ (1 \leq k \leq s) : \sum_{A_i \in C} R_k(i) \geq R_k^j \quad and \quad \forall i, j : A_i, A_j \in C \implies \{A_i, A_j\} \in E(G) \tag{4.4}$$

is maximized.

In the individual agent coalition selection problem (4.4) above, $E(G)$ denotes the edge set of the undirected graph that captures the communication network topology of the underlying multi-agent system (see discussion at the beginning of this Chapter). Clearly, the constraint on every candidate coalition's *cliqueness* can be expressed in terms of the parameters pertaining to the agents' basic properties and capabilities such as, for instance, the radii of the agents' communication ranges [50].

All three constraint optimization problems thus far are stated from the perspective of an individual self-interested agent. However, when it comes to collaborative distributed problem solving MAS, as discussed at length in Chapter 2, the goal is to optimize the behavior of the entire system, and not that of an individual agent. Thus, in the DPS domains, the goal is to determine the coalition structure so that some joint, global utility function is optimized. The simplest such joint utility function in our problem framework is the sum of values of those tasks that are serviced by a particular set of agent coalitions.

Hence, assuming the coalitions are required to be *non-overlapping,* the individual agent's constraint optimization problem (4.3) can be converted into finding the optimal coalition structure from the perspective of the entire system:

Find coalitions $C_1, C_2, ...C_q \subseteq \{A_1, ..., A_n\}$ such that

$C_I \neq C_J \implies C_I \bigcap C_J = \emptyset$;

$\bigcup_I C(I) = \{A_1, ..., A_n\}$; and

$\sum_{I=1}^{q} V^I$ *is maximized*

where V^I *are the respective values of the q independent, distinct tasks* $\{T^I : 1 \leq I \leq q\}$ *that can be simultaneously serviced by these particular coalitions* $\{C_I : 1 \leq I \leq q\}$, *so that each coalition* C_I *is mapped onto a unique task* T^I.

Moreover, as discussed in Chapter 2, the goal is to solve the coalition formation problem as

formulated above in a *fully decentralized* manner. Also, in dynamic MAS environments, coalition formation (phrased as the above distributed constraint optimization problem) would need to be performed *repeatedly,* as demanded by the changes in the tasks and the agents' communication topology.

As a special case, one can consider the scenario where (i) each task $T(j)$ has a single resource requirement, R^j, (ii) each task's value $V(j)$ satisfies $V(j) = R^j$, (iii) each agent has a single capability needed for servicing tasks, and (iv) this capability value is equal to 1 for all agents in the system. Then, given some set of such tasks, each agent would strive to form a legal coalition of a size at least equal to the value of the most valuable available task. If only those coalitions corresponding to the cliques in the underlying agents' communication topology graph are considered to be legal, and if the environment is a "soup" in which there is an abundance of tasks of all possible positive integer values, then this very restricted special case of the distributed constraint optimization via coalition formation problem reduces to the basic *distributed maximal clique* problem, originally addressed in [51].

After the basic mathematical formalisms that cast the variants of (distributed) coalition formation and coalition-to-task mapping problems into the *(distributed) constraint optimization* terms, the stage is now set for presenting our coalition formation algorithm. Therefore, we now shift the emphasis to the algorithmic issues pertinent to coalition formation as a special case of the distributed consensus problem. For simplicity, in the sequel, we will not explicitly take into consideration the agents' tasks, nor those tasks' values and resource requirements. Likewise, the MCDCF algorithm itself will nor explicitly concern itself with the distributed constraint optimization problem of mapping coalitions to tasks in order to maximize an underlying joint utility function of the cooperative DPS multi-agent system. Since we would like to focus now on how agents can effectively reach distributed consensus on the coalition structure, all these other considerations will be abstracted via an appropriately defined *coalition value function.* In the framework formalized earlier in this subsection, the maximal value of any task that a given coalition can service would be a natural example of how to define this coalition value function. Regardless of how exactly is this coalition value function defined, an agent is assumed to have sufficient computational resources to be able to efficiently evaluate each candidate coalition, and assign values to those candidate

coalitions. Once that is done, each agent sorts coalitions with respect to their values, and then engages in the distributed coalition formation algorithm. When the only allowable coalitions are those corresponding to the cliques in the underlying graph, we finally arrive at the sought-after *maximal clique based distributed coalition formation* (MCDCF) problem framework.

The MCDCF algorithm will be described in some detail in section 4.2. Before we dwell into the algorithm description, however, we need to specify the necessary assumptions for that algorithm to be applicable and effective.

4.1.2 Assumptions Underlying MCDCF Algorithm

Our distributed maximal clique based coalition formation algorithm is described in the next section. For this algorithm to be applicable, the following assumptions additional to the ones already discussed need to hold:

– In every possible scenario, the resulting coalition structure is a *partition* of the set of agents into some number of mutually disjoint subsets (coalitions), where each such coalition is nonempty. In particular, an agent can belong only to one coalition at a time.

– A coalition can service only one task at a time, and the coalition value is a function of a single most valuable task that this coalition can complete.

– Communication bandwidth availability is assumed sufficient.

– Each agent has a sufficient local memory (including the message buffers) for storing all the information received from other agents.

– Communication is reliable during the coalition formation, in the following sense: if an agent A_i sends a message to another agent A_j, then either agent A_j gets *exactly* the same message that A_i has sent, or else the communication has failed, so that A_j does not receive anything from A_i at all. In particular, we assume no *scrambled* or otherwise modified messages are ever received by an agent. Of course, once the coalitions have been already formed, the above assumption on communication reliability need no longer hold.[2]

– Each agent has a unique global identifier, 'UID', and the agent knows its UID.

[2]As this requirement is still restrictive, and considerably limits the robustness of our algorithm, we will try to relax this assumption in our future work, and enable the agents to effectively form coalitions even in the presence of some limited amount of communication noise during the coalition formation process itself.

– There is a total ordering, $\prec$, on the set of UIDs, and each agent knows this ordering.

– Each agent has, or else can efficiently obtain, a reliable knowledge of which other agents are within its communication range.

– The *veracity* assumption holds, i.e., an agent can trust the information received from the neighboring agents.

On the other hand, an agent need not *a priori* know the UIDs of other agents, or, indeed, how many other agents are present in the system at any time.

We remark that, while we assume the sufficient communication bandwidth and local memory storage for the algorithm to be able to execute, our MCDCF algorithm is actually *highly resource-aware.* In particular, MCDCF heuristically attempts to simultaneously minimize both the local computation and memory requirements, and the overall amount of communication. A fairly detailed quantitative cost analysis of the algorithm will be given in Chapter 5.

After the preliminary discussion, some basic mathematical formalisms specifying the problem addressed, and an explanation of the assumptions that we make, we are now ready to present, analyze and discuss our distributed coalition formation algorithm. The *Maximal Clique based Distributed Coalition Formation* (MCDCF) algorithm is presented in section 4.2. An example of a simple network of agents, and how the algorithm works when applied to this network, is given in section 4.3. The algorithm's cost analysis and a thorough discussion of its main properties, strengths and weaknesses will follow in Chapter 5.

4.2 The MCDCF Algorithm Description

In this section, we describe the MCDCF algorithm itself. We first informally define the maximal clique based approach to distributed coalition formation for task allocation, as follows. The candidate coalitions are going to be required to be *cliques* of uniformly bounded sizes. This requirement, in practice, means one of the two possibilities. One possibility is that it is *a priori* known that the underlying MAS communication network topology is such that it can be guaranteed that there are no cliques that are prohibitively large. Otherwise, the system designer, based on the application at hand and the available system resources (local computational capabilities of each agent, bandwidth of the agent-to-agent communication links, etc.), *a priori* chooses a threshold, $K = K(n)$, such

that only the coalitions of sizes up to K are considered.

Agents themselves subsequently form coalitions in a fully distributed and online manner, as follows. Each agent (i) first learns of who are its neighbors, then (ii) determines the appropriate *candidate coalitions,* that the agent hopes are (preferably maximal, but certainly of sizes bounded by K) cliques that it belongs to, then (iii) evaluates the utility value of each such candidate coalition, measured in terms of the joint resources of all the potential coalition members, then (iv) chooses the most desirable candidate coalition, and, finally, (v) sends this choice to all its neighbors. This basic procedure is then repeated, together with all agents updating their knowledge of (a) what are the preferred coalitions of their neighbors, and (b) what coalitions have already been formed.

As mentioned in the previous section, each agent has its unique global name, called *UID.* Since we will identify an agent with the corresponding *vertex* in the underlying communication network graph of this MAS, we shall denote agents by v_i in the reminder of this Chapter, as well as Chapter 5. Moreover, we often won't explicitly distinguish between an agent and its name or UID.

In addition to its globally unique identifier UID, which we assume is a positive integer, and the vector of capabilities, each agent also has two local flags that it uses in communication with other agents. One of the flags is the binary "decision flag", which indicates whether or not this agent has already joined some coalition. Namely, $decision \in \{0, 1\}$, and the value of this flag is 0 as long as the agent still has not irrevocably committed to what coalition it is joining.

The second flag is the "choice flag", which is used to indicate to other agents, how happy is the agent with its current tentative choice or proposal of the coalition to be formed. That is, the choice flag indicates the level of an agent's urgency that its proposal for a particular coalition to be formed be accepted by the neighbors to whom this proposal is being sent. Initially, each agent v_i broadcasts $choice(i) = 3$, to indicate that it is sending its neighborhood list as the "ground zero" coalition proposal. During the subsequent rounds, if an agent v_i sends to its neighbors the choice flag value $choice(i) = 0$, that means that this agent has no satisfactory alternatives to its currently proposed coalition. The choice flag value of 1 indicates that an agent can afford to change its coalition choice, but that each of the available alternative coalitions is strictly less preferred than the current proposal. Finally, $choice(i) = 2$ indicates that agent v_i has alternative choices that are of equal preference as the currently proposed coalition.

We remark that any *candidate coalition,* that is, a subset of the set of all neighbors of an agent, such that the agent currently considers this subset to be a possible choice of a coalition that this agent would like to join, need not be a clique, let alone a *maximal clique.* Indeed, based on its strictly local knowledge, an agent, in general, does not know which of its candidate coalitions of sizes greater than three are actually cliques, if any. However, only those candidate coalitions that indeed *are cliques* will ever be agreed upon by the participating agents, and therefore have a chance of possibly becoming the *actual* coalitions.

Moreover, in case of the candidate coalitions with fewer than K members that actually end up being agreed upon by the participating agents, it can be shown that these agent coalitions indeed do form *maximal cliques,* in the following sense. These cliques can be possibly made larger in only two ways: by adding new members so that the threshold K is exceeded, and/or by taking away one or more agents that already belong to (an)other, already formed coalition(s).

We split the MCDCF algorithm into six stages. Four of these six are iteratively repeated until the consensus on coalition formation is reached. We point out, however, that *each agent executes these stages asynchronously and in parallel with the other agents.* The only assumption about the coarse-grain synchronization among the agents, that we chiefly make for the purposes of analyzing the algorithm's complexity (see Chapter 5), is that an agent does not begin another round of *Stages 2 – 5* before its neighbors are done with the previous round. The practical way of enforcing some level of coarse-grain synchronization is to use time-outs: should an agent fail to receive the update from one of its neighbors within some pre-specified time interval, the agent assumes that its neighbor is no longer available for the coalition formation, and deletes this neighbor's UID from all the appropriate lists (see below). In the sequel, we won't bother distinguishing between an agent or a communication network node, v_i, and this agent's (alternatively, node's) UID, i; the intended meaning in any given situation will be clear from the context.

Stage 0: Each agent, asynchronously and in parallel with all other agents, broadcasts the basic personal data to all its immediate neighbors. The entries in a tuple of that data are (i) the agent's UID, (ii) the agent's list of immediate neighbors, L(i), that includes i, (iii) the value of the choice flag that indicates that the list sent is the neighborhood list, and (iv) the value of the decision flag. In the more general scenarios where each agent is equipped with a set of capabilities (see the

discussion in the previous section, as well as references [43, 54]), an agent also broadcasts its vector of capabilities.

Each agent also receives the corresponding tuples (possibly including the capability vectors) from all of its neighbors. Those neighbors whose messages have not been received within the allotted time are discarded from the future coalition considerations.

Stage 1: Each agent v_i locally computes, asynchronously and in parallel with all other agents, the overlaps of its neighborhood list with the neighborhood lists that it has received from its neighbors, $C(i,j) \leftarrow L(i) \cap L(j)$.

Each agent repeats *Stages 2 – 5* until it either reaches a consensus on what coalition it is joining, or else is left with no choice but to form the trivial single-member coalition.

Stage 2: During the first round, each agent at this stage evaluates all of its candidate coalitions, and then sorts these coalitions in the nonincreasing order with respect to those coalitional values. If the coalition size is the only criterion, then the agent sorts its candidate coalitions in nonincreasing order with respect to their size, and breaks the ties either arbitrarily, or according to some fixed rule that uses the agents' UIDs. For instance, the agent may linearly order all coalitions of the same size with respect to the lexicographic ordering of those coalitions' members' UIDs (see the example in section 4.3).

If the combined *capabilities* of each tentative coalition are the main criterion, then each agent first evaluates or estimates the *coalition value* with respect to its local knowledge of the existing tasks and their demands in terms of the coalitional capabilities. The agent then orders the candidate coalitions based on these estimated coalitional values, breaking the ties according to any rule that gives priority to the coalitions of, first, the greatest size, and second (in the framework with capability vectors), whose capability vector is the greatest with respect to the usual partial order on tuples. These tie-breaking restrictions ensure that, intuitively speaking, even when several candidate coalitions containing agent v_i and having different sizes and/or capabilities also have the same task of maximal value that is serviceable by them, a smaller / less capable coalition is never going to be chosen by v_i over a larger / more capable one. This way, the desired monotonicity property in exploring the lattice of subsets of the agent set $\{A_1, ..., A_n\}$ is ensured. In the DPS problem domains where the resources are depletable, and where the marginal utilities matter [38,

60], this tie-breaking would be sub-optimal. However, we purposefully choose a model where the marginal utilities are not a concern, and we want to ensure that each agent, in its quest for a coalition to join, performs a *monotonic search* of the lattice of candidate coalitions; see also discussion in Chapter 3 on the comparison and contrast of our problem framework with that in [43].

During the subsequent rounds, an agent v_i looks for the information from its neighbors on whether any of those neighbors has joined a coalition "for good" during the previous round. Those neighbors that have are deleted from the neighborhood list $L(i)$; the intersection lists $C(i,j)$ and the candidate coalition lists $C(i)$ are also updated accordingly, and those $C(i,k)$ for which v_k is deleted from the neighborhood list $L(i)$ are also deleted. Likewise, the coalition values $val[C(i,k)]$ are appropriately updated, and the remaining candidate coalitions are re-sorted, as needed.

Stage 3: An agent v_i picks one of the most preferable lists $C(i,j)$; let $C(i) \leftarrow chosen\ [C(i,j)]$. If the group or coalition size is the main criterion, then one of the candidate coalitions of maximal size is chosen. If the combined *capabilities* of each tentative coalition for servicing various tasks are the main criterion, then the agent picks as its current coalition proposal one of the candidate coalitions with the highest coalition value. Since the assumption is that the capability vector of each agent has all its entries nonnegative, this *monotonicity property* ensures that no proper subset of a candidate maximal clique coalition is ever chosen – except in the cases when the clique size exceeds the threshold, $K(n)$.

Stage 4: Each agent sends its tuple with its UID, the tentatively chosen list $C(i)$, the value of the choice flag, and the value of the decision flag, to all its neighbors. Since we assume that an agent's capabilities are neither renewable nor depletable during the agent's deliberation (i.e., while the agent negotiates on coalition formation, as opposed to servicing a task), there is no need to re-broadcast the vector of capabilities.

Likewise, each agent receives the corresponding tuples from its current neighbors.

Stage 5: An agent v_i compares its chosen list $C(i)$ with the lists $C(j)$ received from its neighbors. If a satisfactory clique that includes the node i exists, and all members of this clique have selected it at this round as their current coalition of choice (that is, if $C(i) = C(j)$ for all $j \in C(i)$), this will be efficiently recognized by the agents that are forming this particular clique. The decision flag

of each agent $j : j \in C(i)$ is then set to 1, the coalition is formed, and this information is broadcast to all of the neighbors. In particular, the agent v_i locally broadcasts its agreed-upon coalition, and decision flag $decision(i) = 1$, to all of its still remaining neighbors. Else, if no such agreement is reached, then agent v_i, based on its UID and priority, and its current value of the *choice flag,* either does nothing, or else changes its mind about its current coalition of choice, $C(i)$. The latter scenario is possible only if $choice(i) > 0$, meaning that there are other choices of *nontrivial* candidate coalitions $C(i)$ that have not been tried out yet and are still potentially available to agent v_i.

As already remarked, each agent uses *time-outs* in order to place an upper bound on for how long it may be waiting to hear from another agent during any stage of the algorithm. If an agent v_p has sent a message to another agent v_q, and the latter is not responding, then there are four possibilities: (i) the agent v_q has failed; (ii) the communication link from v_q to v_p has failed; (iii) while v_q is in v_p's communication range, the converse does not hold (but v_p may not know it), and (iv) either the agent v_q, or the communication link from v_q to v_p, is too slow. In each case, once v_p has waited sufficiently long to hear from v_q, agent v_p will simply consider v_q unavailable for the joint coalition formation, and will delete v_q from its candidate coalition lists. Thus, the case (iv) above will be treated by the agent v_p in exactly the same way as the other three cases.

In order to ensure that the algorithm avoids cycling in every possible scenario, once an agent, v_i, changes its mind about the preferred coalition $C(i)$, it is not allowed through the remaining rounds of the algorithm to go back to its old choice(s). Once no other choices are left, this particular agent sticks to its current choice, and waits for other agents to settle to their choices. This requirement ensures the ultimate convergence to a coalition structure that all agents (locally) agree on. That is, under the assumptions stated in the previous section, the agents will reach consensus on the coalition structure after a finite number of rounds through the *Stages 2 – 5.* Moreover, if the maximum size of any $L(i)$ is sufficiently small, then the convergence will be relatively fast. A much more detailed algorithm cost analysis, as well as a discussion of MCDCF's main properties, will follow in Chapter 5.

Once all the agents exit the iterated execution of the *Stages 2 – 5,* each formed coalition will indeed be a clique. Moreover, those agent coalitions whose sizes do not exceed the pre-specified

threshold, K, are also maximal in a sense that, given such a coalition C, no agent(s) outside of this coalition can be added to it, so that the following requirements *simultaneously* hold: (i) each of the new agents is already adjacent to all the "old" coalition members of C, (ii) if more than one new agent is added, then all the added agents are also pairwise neighbors to each other, (iii) the newly added agent(s) do not belong to a coalition (or coalitions) that have already been formed, and (iv) the new size of the augmented coalition C^{new} is still at most K.

However, it is easy to construct examples of the underlying graphs and the particular runs of the algorithm such that, once every agent joins a coalition and the algorithm terminates, several agents end up in trivial coalitions, such as the coalitions of size 1 or 2.

It is therefore reasonable, in many application contexts, to introduce an *optional Stage 6* of the algorithm during which some of these small and, therefore, potentially not sufficiently useful coalitions, may be merged together. Thus, if some two small coalitions, or one small and one bigger coalition, are adjacent to each other,[3] they can be merged together. For an illustration, we refer the reader to the worked out example in section 4.3. Obviously, the connectivity of the coalitions that result from such mergers, and therefore their tolerance to the subsequent communication link failures, are in general going to be lower than that of those coalitions that are *genuine cliques.*

4.3 How MCDCF Algorithm Works: A Simple Example

To show how the MCDCF algorithm works, we use a simple example. The interconnection topology of a MAS made of seven agents is given in *Figure* 4.1. For simplicity, we assume that the only value associated with each coalition is the coalition's size. We also assume that no agent falls behind by too much, i.e., that all agents complete each round of *Stages 2 – 5* within the allotted time.

We notice that the largest clique that any agent in this example belongs to is a 3-clique. However, several agents belong to multiple triangles, so this toy example is instructive insofar as how the agents break ties, avoid deadlocks, and reach consensus on the coalition structure.

First, the initialization stage takes place, during which each agent locally broadcasts its list of neighbors to each of its neighboring agents. Then each agent, asynchronously and in parallel with

[3]That is, if there exist node x in the first coalition and node y in the second such that x and y are adjacent in the underlying graph.

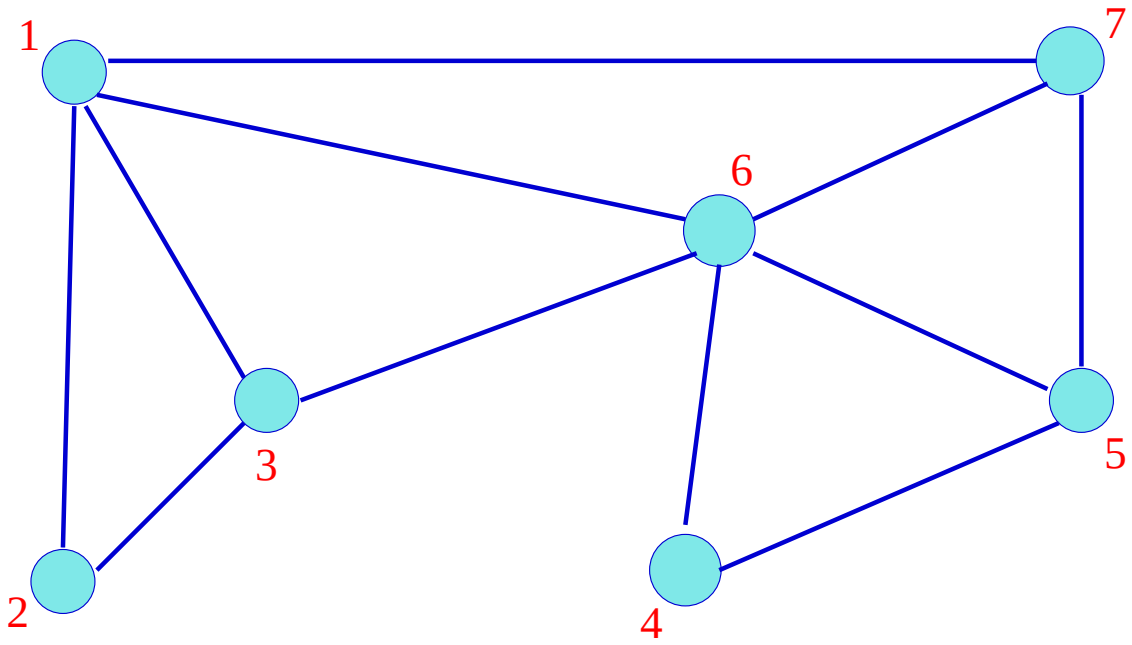

Figure 4.1: An example of a communication network topology for a small agent ensemble

the other agents, forms the initial candidate coalitions by computing the pairwise neighborhood list intersections. However, not all of thereby formed coalitions are *reachable:* if $C(i,k) \subsetneq C(i,j)$ then there exists a *maximal reachable* candidate coalition $C(i,\cdot)$, $C(i,k) \subseteq C(i,\cdot) \subsetneq C(i,j)$, such that for some agent v_p : $p \in C(i,\cdot)$, there exists an agent v_q such that $q \in C(i,j)$, and v_p and v_q are not adjacent to each other. Clearly, this agent v_p will never agree on forming the bigger coalition $C(i,j)$, since this coalition would include at least one agent that v_p cannot directly communicate with. Hence, all such sets of nodes $C(i,j)$ that *properly include* other prospective coalitions can be safely deleted from the list of candidate coalitions.

In our example, agent v_1 can safely delete the set $\{1,3,6,7\}$, as this coalition can never be agreed upon by all four agents. Concretely, v_3 will never agree to form a coalition that includes v_7 (and vice versa). Furthermore, based on the information received from v_3 and v_7, agents v_1 and v_6 can readily and safely *infer* that the coalition $\{1,3,6,7\}$ cannot be agreed on.

Once all such *unreachable coalitions* are deleted, the column denoted *"candidate coalitions* $C(i,j)$*"* in *Table* 4.1 is obtained. After that, each agent picks one of the available choices. Under the *monotonicity assumption* discussed earlier, each agent will select one of the *maximal sets* (potential cliques) that it belongs to. Each agent also appropriately sets the value of its *choice flag.* When

Table 4.1: The coalition formation configuration at the end of the first round

Node	neighborhood $L(i)$	neighborhood overlaps $L(i) \cap L(j)$	*candidate coalitions* $C(i,j)$	*chosen* $C(i)$	**choice flag**
v_1	$\{1,2,3,6,7\}$	$\{1,2,3\}$, $\{1,2,3,6\}$, $\{1,3,6,7\}$, $\{1,6,7\}$	$\{1,2,3\}$, $\{1,3,6\}$, $\{1,6,7\}$	$\{1,2,3\}$	2
v_2	$\{1,2,3\}$	$\{1,2,3\}$	$\{1,2,3\}$	$\{1,2,3\}$	0
v_3	$\{1,2,3,6\}$	$\{1,2,3,6\}$, $\{1,2,3\}$, $\{1,3,6\}$	$\{1,2,3\}$, $\{1,3,6\}$	$\{1,2,3\}$	2
v_4	$\{4,5,6\}$	$\{4,5,6\}$	$\{4,5,6\}$	$\{4,5,6\}$	0
v_5	$\{4,5,6,7\}$	$\{4,5,6\}$, $\{4,5,6,7\}$, $\{5,6,7\}$	$\{4,5,6\}$, $\{5,6,7\}$	$\{4,5,6\}$	2
v_6	$\{1,3,4,5,6,7\}$	$\{1,3,6\}$, $\{1,3,6,7\}$, $\{4,5,6\}$, $\{1,5,6,7\}$, $\{4,5,6,7\}$,	$\{1,3,6\}$, $\{1,6,7\}$, $\{4,5,6\}$, $\{5,6,7\}$	$\{1,3,6\}$	2
v_7	$\{1,5,6,7\}$	$\{1,6,7\}$, $\{1,5,6,7\}$, $\{5,6,7\}$	$\{1,6,7\}$, $\{5,6,7\}$	$\{1,6,7\}$	2

the coalition size is the criterion of that coalition's value, agent v_1, for example, would set its *choice flag* to 2, since it has more choices that are just as preferable as the selected candidate coalition, the set $\{1,2,3\}$. Once this step is completed by all agents, the overall configuration is reached as depicted in *Table* 4.1, where each row represents the corresponding agent's current local knowledge of the neighborhood structure and of that agent's choice of a tentative coalition.

For simplicity, we have assumed in this example that, in case of a tie, each agent picks the lexicographically lowest coalition. Hence, the agents v_1, v_2 and v_3 immediately reach the agreement on forming the coalition $\{1,2,3\}$. The other four agents, however, do not reach an agreement after the first round. Let us assume that, among several agents with the same value of the choice flag, $choice > 0$, the ones with the lowest index among their neighbors are required to change their proposed coalitions. In our example, this means that, in the second round, v_5 drops $\{4,5,6\}$ and selects $\{5,6,7\}$ instead. Upon doing so, agent v_5 also adjusts its value of the choice flag, and broadcasts these changes to agents v_4, v_6 and v_7. Also, since agents v_1 and v_3 are no longer available, agents v_6 and v_7 delete v_1, v_3 from their neighborhood lists, and update the candidate coalitions accordingly.

After each of the agents v_5, v_6 and v_7 performs the described updates, and then locally broadcasts its new configuration to all of the remaining neighbors, the overall configuration at the end of the second round is as in *Table* 4.2.

Table 4.2: Coalition configuration at the end of the second round

Node	*candidate coalitions* $C(i,j)$	*chosen coalition* $C(i)$	*choice flag*	*decision status*
v_1	$\{1,2,3\},\ \{1,3,6\},\ \{1,6,7\}$	$\{\mathbf{1},\mathbf{2},\mathbf{3}\}$	...	**done**
v_2	$\{1,2,3\}$	$\{\mathbf{1},\mathbf{2},\mathbf{3}\}$	...	**done**
v_3	$\{1,2,3\},\ \{1,3,6\}$	$\{\mathbf{1},\mathbf{2},\mathbf{3}\}$	...	**done**
v_4	$\{4,5,6\}$	$\{4,5,6\}$	0	busy
v_5	$\{5,6,7\}$	$\{5,6,7\}$	0	busy
v_6	$\{4,5,6\},\ \{5,6,7\}$	$\{4,5,6\}$	2	busy
v_7	$\{5,6,7\}$	$\{5,6,7\}$	0	busy

Table 4.3: The final coalition configuration; the only unhappy node is v_4

Node	*candidate coalitions* $C(i,j)$ *at the time of agreement*	*chosen coalition* $C(i)$	*choice flag*	*decision status*
v_1	$\{1,2,3\},\ \{1,3,6\},\ \{1,6,7\}$	$\{\mathbf{1},\mathbf{2},\mathbf{3}\}$	...	**done**
v_2	$\{1,2,3\}$	$\{\mathbf{1},\mathbf{2},\mathbf{3}\}$	...	**done**
v_3	$\{1,2,3\},\ \{1,3,6\}$	$\{\mathbf{1},\mathbf{2},\mathbf{3}\}$	...	**done**
v_4	$\{4\}$	$\{\mathbf{4}\}$	0	**doomed**
v_5	$\{5,6,7\}$	$\{\mathbf{5},\mathbf{6},\mathbf{7}\}$	0	**done**
v_6	$\{5,6,7\}$	$\{\mathbf{5},\mathbf{6},\mathbf{7}\}$	0	**done**
v_7	$\{5,6,7\}$	$\{\mathbf{5},\mathbf{6},\mathbf{7}\}$	0	**done**

After the second round is completed, the only agent that still has some "maneuvering room" is v_6; since there is still no agreement, and $choice(v_6) > 0$, whereas $choice(v_i) = 0$ for all $i \neq 6$ such that v_i is not done yet, in the next round v_6 changes its coalition proposal to $\{5,6,7\}$, thereby reaching the consensus with v_5 and v_7, and yielding the final configuration as in *Table* 4.3.

The coalition structure that results from the MCDCF algorithm invoked on the original network of agents is depicted in *Figure* 4.2. In the end, three coalitions are formed: two with three agents in each, and one trivial, singleton coalition. The two three-member coalitions are also *maximal cliques* in the original network; see *Figure* 4.1. Thus, in the end, each of the agents, except for v_4, has joined (one of) the coalition(s) optimal for it. Since agent v_4 has ended up left out, and since it is adjacent to the coalition $\{v_5, v_6, v_7\}$, it can be merged with this coalition to form a larger coalition $\{v_4, v_5, v_6, v_7\}$ in the optional *Stage 6* (see discussion in section 4.2). Given the assumed *monotonicity* and (locally constrained) *super-additivity* of the multi-agent environment, any coalition arising from such a subsequent merger clearly cannot be a clique.

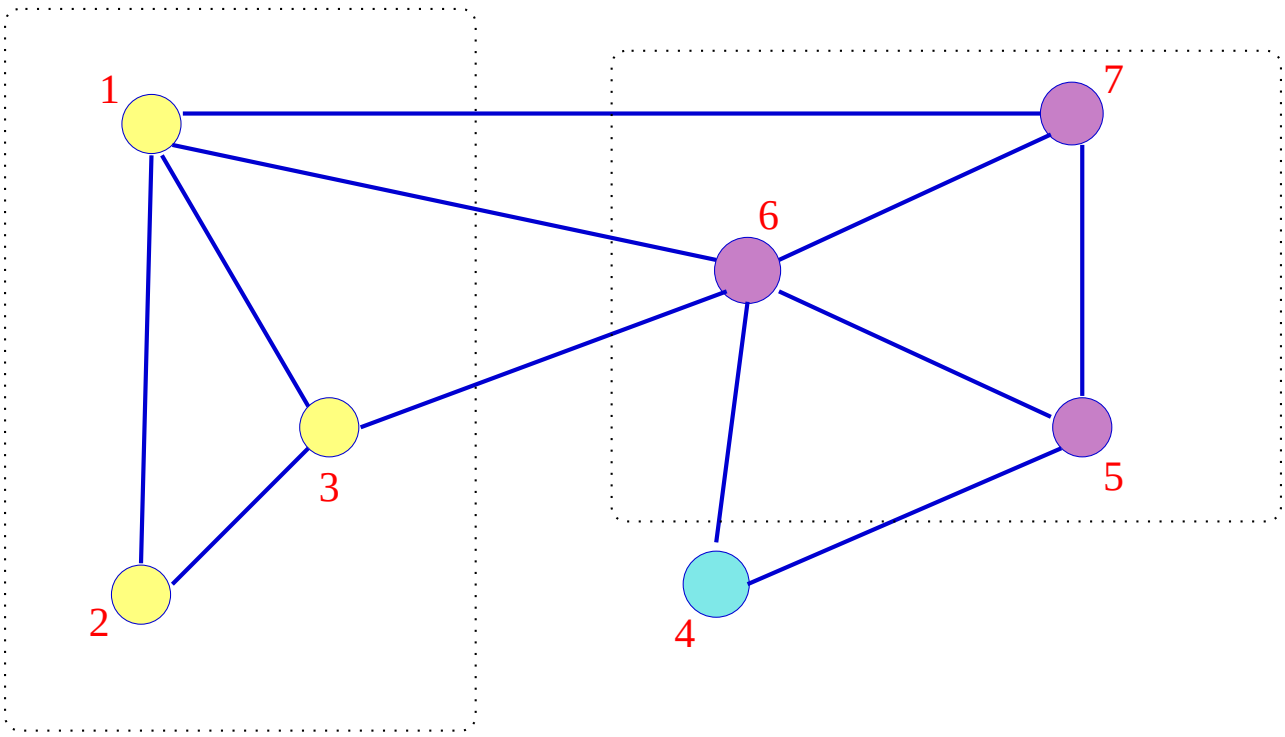

Figure 4.2: Final configuration in our example with three formed coalitions

CHAPTER 5

ALGORITHM ANALYSIS AND DISCUSSION

After having presented the MCDCF algorithm, we now analyze its costs and discuss its main features in some detail. We start with a thorough analysis of computational and communication complexity of MCDCF. We show that, for sufficiently sparse underlying communication networks, this algorithm is indeed feasible, scalable and of very modest computational and communication costs. We also briefly review the general graph algorithmic issues surrounding our MCDCF approach. We then revisit some of the game-theoretic issues whose discussion we initiated (in much more general terms) back in Chapter 2. This game theoretic discussion of MCDCF is followed by an overview of the major strengths, as well as some of the noticeable limitations and weaknesses, of the MCDCF-based approach to distributed coalition formation. We then indicate how can our approach be adjusted and extended so that it can also be applicable to the non-DPS multi-agent domains. Last but not least, we briefly outline some ideas for the future research that would take a generalized version of MCDCF as its starting point.

5.1 Cost Analysis of MCDCF Algorithm

The overall cost of a distributed algorithm can be measured in terms of several parameters. The overall worst-case execution time, the necessary local memory storage at each agent, the overall worst-case amount of local computation per agent, the maximum bandwidth requirement during any stage of the inter-agent communication, and the maximum possible energy consumption per agent, are some of the resource parameters that are most frequently explicitly taken into account

in such an algorithmic complexity analysis. In the sequel, we will focus on the main resource requirements per agent: the total amount of local computation of each agent, how much data each agent has to locally store, and how much data each agent is required to send to and receive from other agents. In particular, we will analyze under what assumptions will the required amounts of computation time, memory storage and communication be feasible so that an agent would want to venture into participating in the coalition formation based on our algorithm. However, the MCDCF cost analysis below will not address the issues that arise, nor the effects those issues would have on the overall algorithm's cost, once the communication (un)reliability, (in)sufficient bandwidth or non-negligible network delays are quantified and then explicitly taken into account.

The first and foremost cost requirement is that the algorithm be of a feasible computational complexity when it comes to its overall worst-case running time. At the very least, this running time needs to be *polynomial in the number of agents, n.* Moreover, for the MMAS made of anywhere from thousands possibly up to millions of agents, the asymptotic upper bound on the total number of elementary computational steps better be close to *linear* in n. We show that, under a restrictive yet reasonable assumption about the sparseness of the underlying network of agents, this goal can be attained. We shall focus our algorithmic complexity analysis to that pertaining to *the worst-case computation time.* We will subsequently also briefly discuss the asymptotic costs related to the memory storage and communication per agent.

Let us estimate how much computation is done by each agent, and what is the asymptotic upper bound on the total number of steps of the entire algorithm. We assume for now that the time to send and receive messages is not increasing the asymptotic upper bounds on local computations. We will show at the end of this section that the amounts of data that the agents locally broadcast in the algorithm is fairly modest. The high-level assumption that the time required for all message sends and receives does not affect the "Big-Oh" estimate of the asymptotic execution time of a distributed algorithm can be argued to be justifiable when the following more concrete assumptions hold:

– the sufficient availability of communication bandwidth;

– sufficiently big buffers for the arriving messages; and

– bounded network delays.

We shall split the time complexity analysis into two parts. First, we will estimate the maximum

number of rounds, i.e., how many times an agent may need to iterate through the *Stages 2 – 5* (see subsection 4.2). Second, we will show that the amount of computation per agent within a single round is relatively small. A feasible upper bound on the total number of rounds, combined with a modest amount of computation per round, will together ensure the overall feasibility of the algorithm. Just like we did in Chapter 4, we will continue to use the terms *agent* (in a multi-agent system) and *node* (in the underlying graph capturing the communication topology of this MAS) interchangeably throughout this Chapter, as well.

Let $K = K(n)$ be a nonnegative, monotonically nondecreasing function of n, and let the class of the underlying graphs of agents be such that, for any positive integer n, the size of any clique in the graph is bounded by $K(n)$. To show under what conditions is our algorithm going to take at most polynomially many rounds, we establish the following result:

Proposition 5.1. *Let an undirected graph with n nodes be such that the bound on the maximum node degree in this graph is given by $K(n) \leq c \cdot \log n$, for some positive constant c. Let us assume each node in the graph is an agent with sufficient computational and communication resources. Then the maximum number of rounds of the* MCDCF *algorithm will be polynomial in the number of agents, n.*

Proof. Let $K(n) \leq c \cdot \log n$. Let v_i be an arbitrary agent. Since v_i has at most $K(n)$ neighbors, the maximum number of candidate coalitions that would include agent v_i is at most $2^{K(n)}$, which is bounded from above by $2^{c \log n} = n^c$. During each round of the algorithm, either at least one new coalition is formed, or at least one agent has to change its mind about its coalition proposal, or possibly both these two events take place. When an agent changes its mind and abandons its current choice of the proposed coalition, that old coalition choice is permanently discarded. Therefore, at each round, the total number of the remaining candidate coalitions in the entire system is reduced by at least one. Since there are at most $n \cdot 2^{K(n)} \leq n \cdot n^c$ possible coalitions at the beginning, the total number of rounds is $O(n^{c+1})$. □

Hence, if the underlying network topology of a multi-agent system is such that $K(n) = O(\log n)$, then our algorithm will run in time polynomial in n. Moreover, if $K(n) \leq c \log n$ holds for $c = 1$, then the number of rounds is at most quadratic in n. We shall assume that the bound

$K(n) \leq c \log n$ asymptotically holds for some positive, real constant c close to 1. We shall also assume that, whatever the criteria of "goodness" for these clique-based coalitions may be, given the necessary data about its neighbors, an agent can efficiently compute the candidate coalition's *value,* $val[C(i)]$, for any such potential coalition $C(i)$. In particular, we shall assume in the sequel that, for any agent v_i, a single evaluation or value estimation of the value of any coalition $C(i)$ of size m takes $O(m^2)$ steps; see the proof of Proposition 5.2 for details.

Proposition 5.2. *Let the assumptions from the previous discussion hold. Then the amount of local computation that any agent has to perform during any round of our* MCDCF *algorithm is polynomial in the size of the data structures involved (cf. lists $L(i), C(i)$ and $C(i,j)$). In particular, assuming that the encoding of all information about a single agent (its UID, list of available resources, etc.) is bounded by $O(\log n)$, the total number of elementary bitwise operations of our algorithm is bounded by $O((\log n)^4)$.*

Proof. Under the stated assumptions, $K(n) \leq c \cdot \log n$, and therefore for an arbitrary agent v_i, the lists $C(i), L(i), C(i,j)$ that this agent operates with are also of sizes bounded by $O(\log n)$. Since *Stages 0 – 1* are executed only once, and *Stage 4* includes communication *only,* we just need to estimate the amounts of local computation during *Stages 2, 3 and 5.*

In *Stage 2,* a number of operations on individual lists and pairs of lists are performed. Sorting a list with K elements takes time $O(K \log K)$. Finding an element and deleting it from a list with at most K elements takes time $O(K)$. When both lists are of size $O(K)$, comparing two sorted lists, computing their intersection, and testing if one sorted list is a subset of the other, each takes at most $O(K \cdot \log K)$ operations. An agent v_i may need to perform up to K such pairwise list comparisons, intersections and similar list operations – one for each $j \in L(i)$, where $|L(i)| \leq K$. Each of these operations is done at the granularity level of a single list element, which we have assumed is encoded by $O(\log n)$ bits. Hence, the total number of list operations at the bit level is bounded by $O(K^2 \log K \log n)$, which, given our assumptions, is just $O((\log n)^3 (\log \log n))$.

During the first round, *Stage 2* also entails each agent v_i evaluating the quality (that is, the desirability) of each of its candidate coalitions $C(i)$. We shall denote this measure of a candidate coalition's quality by $val[C(i)]$. During the subsequent rounds, these values $val[C(i)]$ of the still remaining candidate coalitions may need to be updated. We therefore need an upper bound on the

computational cost of evaluating different candidate coalitions, and then maintaining and updating these coalitional values.

Let's assume that the time to evaluate any $val[C(i)]$ is bounded by some function $T(m)$, where $m = |C(i)|$ is the size of coalition $C(i)$. Let's also assume that, at each node, the candidate coalitions are *sorted* in a non-increasing order with respect to their values $val[C(i)]$. Then evaluating the coalitional values takes at most $O(\log n) \cdot (T(c \cdot \log n) + c \cdot \log(c \log n) + O(1))$ elementary steps. When $T(m) = O(m^2)$, this simplifies to $O((\log n)^3)$. Since each step is assumed to require no more than $O(\log n)$ bit operations, we arrive at $O((\log n)^4)$ bitwise operations overall. As updating and maintaining the coalition values during the subsequent rounds of the algorithm execution is no costlier than originally computing them from the scratch during the first round, the upper bound of $O((\log n)^4)$ remains valid. This observation completes the asymptotic estimate of the worst-case computation cost of a single execution of *Stage 2.*

Similar analysis applies to *Stage 5,* where the costliest operations are the pairwise list comparisons. During any round, an agent v_i has to carry out at most $|C(i)| \leq K(n) = O(\log n)$ pairwise list comparisons, where each of those lists has $O(\log n)$ entries.

Thus *Stage 2* and *Stage 5,* under the stated assumptions about the cost of calling the subroutine for determining the candidate coalitions' values $val[C(i)]$, together perform $O((\log n)^3)$ basic operations per list element. Since *Stage 3* involves only simple operations on the already sorted lists, and evaluation of a pair of conditionals, it is the *Stages 2 and 5* that dominate the overall amount of local computation at each agent. Therefore, again, the total number of elementary operations per list element per round is $O((\log n)^3)$, which, assuming each list element (whether it is an agent's UID or a value of an agent's resource) is encoded by $O(\log n)$ bits, translates into $O((\log n)^4)$ bitwise operations per round, overall.

To summarize, under the stated assumptions, the overall number of the basic list element operations per agent during a single execution of *Stages 2 – 5* is at most cubic in the maximum size $K(n) = O(\log n)$ of the lists involved, and therefore $O((\log n)^3)$ in the total number of nodes in the graph. Hence, it is only $O((\log n)^4)$ in terms of the total number of bitwise operations.

□

The two *Propositions* above establish that, under the network sparseness assumption, the

MCDCF algorithm is feasible insofar as the total amount of computation per each node is concerned. When it comes to the memory requirements, that is, the required amount of storage per node, in addition to the assumptions we have already discussed, we also additionally assume that the numerical values of each of the agent's capabilities, as well as the candidate coalition values $val[C(i)]$, are each stored with at most $O(\log n)$ bits of precision. Under all these assumptions, it can be readily verified that the total storage per agent is *polylogarithmic* in the total number of nodes n.

Namely, the data an agent needs to store include (i) the basic information about the individual neighboring agents, such as those neighbors' UIDs, decision and choice flags, and resource vectors, and (ii) a collection of the neighboring agents' neighborhood and proposed coalition lists. The greatest amount of data is send and received by each agent during *Stage 0;* the subsequently communicated data then enables an agent to update, i.e., partially overwrite, what it has stored at the initial stage of the algorithm. The total amount of data that needs to be stored at an agent v_i at any time is, therefore, bounded by

$$memory(v_i) \;\leq\; |C(i)| \cdot \sum_{j \in C(i)} \{size(V(j)) + size(L(j)) + size(R(j)) + size(val[C(j)]) + O(1)\} \quad (5.1)$$

which is asymptotically bounded from above by $a \cdot K^3 \cdot \log n$ for some constant $a \geq 1$. Under the assumption that $K \leq c \cdot \log n$, it follows that each agent needs to store only $O((\log n)^4)$ bits of data overall.

A similar analysis holds for the communication cost of the MCDCF algorithm. In case of communication, one needs to establish an upper bound on the amount of data sent and received per round, and then the product of this upper bound with the upper bound on the total number of rounds provides an upper bound on the overall amount of communication per agent. We observe that the communication during *Stage 0,* constituted of local broadcasts of each agent's data that include the resource vectors, takes place only once. Similarly, the communication in *Stage 5* (when the *THEN* branch is taken; see *Appendix* for the pseudo-code) also takes place only once for each agent – namely, when a particular agent has reached consensus with a subset of its neighbors on the coalition that is to be formed. It can be readily shown that the total amount of data exchanged

in both situations in only *polylogarithmic* in the number of agents, n.

Hence, the only communication that takes place in every round is the message exchange during *Stage 4.* During that stage, each agent sends at most $K(n) \cdot O(\log n)$ bits, which, under the same assumptions as before, is only $O((\log n)^2)$. Similarly, insofar as the amount of data an agent receives from its neighbors during *Stage 4,* this amount is at most $K \cdot (K \cdot O(\log n)) = O((\log n)^3)$. Therefore, the total number of bits communicated by each agent *per round* is bounded by $O((\log n)^3)$, or $O(n^{c+1} \cdot (\log n)^3)$ altogether.

5.2 Discussion

The proposed distributed coalition formation algorithm is based on two main ideas. One idea, familiar from the literature (see, e.g., [43] and references therein), is to formulate a distributed task and/or resource allocation problem as a *(distributed) set covering problem, (D)SC,* in those scenarios where the coalition overlaps are allowed, or a *(distributed) set partitioning problem, (D)SP,* when no coalition overlaps are allowed. Two coalitions overlap if there exists an agent that belongs to both of them. It is well-known that the decision versions of the classical, centralized SC and SP problems are **NP**-complete [11, 25]. Consequently, what is needed are efficient distributed heuristics so that the agents can effectively apply DSC- or DSP-based strategies for coalition formation. Several such efficient heuristics are already known and readily available; more details can be found, e.g., in the discussion and references in [43], or in [6]. From a graph-algorithmic standpoint, our MCDCF can be viewed as another such heuristic for distributed set partitioning.

The second main idea, and one of the main motivations behind our approach, is to ensure that the formed coalitions of agents meet the robustness and fault tolerance criteria, which are particularly important in applications where there is a high probability of the subsequent node and/or communication link failures [49, 50]. Clearly, the most robust coalitions of agents of a particular size are those that correspond to *cliques* in the underlying interconnection topology of the agents' communication network. Moreover, the larger such a clique, the more robust the coalition of agents in the clique with respect to the node and/or link failures.

Hence, our goal has been to find a scalable, fully decentralized way to determine (preferably, maximal) cliques in a given graph. Alas, the *Maximal Clique* problem is also well-known to be

NP-hard in the centralized setting [8, 11]. This hardness stems from the fact that an agent may need to test for the "cliqueness" exponentially many candidate subsets that it belongs to. However, in those graphs where the maximum degree of each node is bounded by $c \log n$, the number of subsets that each node belongs to is $O(n^c)$, i.e., *polynomial in the total number of agents.*

We remark that there are also other important and frequently encountered in practice special cases, in terms of the structural properties of the underlying graphs, where the *Maximal Clique* problem turns out to be computationally feasible. In particular, the $O(\log n)$ uniform upper bound on all node degrees is sufficient, but not necessary, for the computational feasibility of the *Maximal Clique* problem [51].

Thus, the intrinsic tension between "the bigger (a coalition), the better" and "the smaller, the more scalable" is resolved in our MCDCF approach by imposing an appropriate upper bound $K = K(n)$ on the maximal allowable coalition size. Each agent that participates in MCDCF coalition formation then locally and greedily tries to find, in a fully decentralized manner, the biggest clique that it belongs to, yet whose size does not exceed the upper bound K. Given the assumption about the collaborative, DPS nature of the underlying MAS domain, and the above discussion on the limitations on the coalition sizes stemming from the elementary algorithmic complexity considerations, the choice of *locally constrained super-additive* multi-agent environments as an appropriate setting for the MCDCF approach to coalition formation appears not only very natural, but, without some considerable modifications to the MCDCF algorithm, actually *necessary.* Some of the possible modifications to and extensions of the basic MCDCF (as presented in Chapter 4) are going to be briefly discussed in the sequel, as we return to the game-theoretic discussion of coalition formation that we have begun in Chapter 2.

The proposed algorithm can be used as a subroutine in many multi-agent system scenarios where the system needs to periodically reconfigure itself in a dynamic and fully distributed manner, and where it is important for the agents to agree *efficiently* on what coalitions are to be formed, rather than spending precious resources on complex negotiation protocols in order to achieve an "optimal" coalition structure, whatever the particular notion of *optimality* in a given setting may be.

However, it is important to point out that, in case of the self-interested agents, some form of negotiation among the agents is typically unavoidable, irrespective of the particular mechanism

employed for the purposes of generating a desired coalition structure. In particular, self-interested agents that would use a variant of the MCDCF algorithm, at the very least, would still need to negotiate on how are the spoils to be divided among them.

According to T. Sandholm in *Chapter 5* of [60], a complete coalition formation process is made of three stages: (i) coalition structure generation, (ii) optimization within each coalition, and (iii) payoff (or utility) distribution; see also [38]. In case of the DPS agents, stages (i) and (ii) generally suffice, but when the agents have their individual preferences over the states of the world in general, and different individual evaluations of the desirability of various coalitions and tasks in particular, addressing (iii) becomes, indeed, necessary for any coalition formation strategy to be complete.

We observe that, in our approach, the stages (i) and (ii) are not separated from each other. When (i) and (ii) are separated, as is common in much of the MAS literature on the subject (see, e.g., [38, 60]), then forming coalitions strictly causally and temporally precedes those coalitions then attempting to solve an appropriate optimization problem such as, e.g., distributed task allocation. Instead, in our work stages (i) and (ii) are *intertwined.* That is, the desired coalition structure emerges as a result of the agents solving simultaneously a coordination and a distributed constraint optimization problem.

We briefly outline our view on why, in many MAS applications, stages (i) and (ii) in the Sandholm's classification scheme ought not to be separated as in *Chapter 5* of [60]. Namely, most often *coordination* among the agents *is not a goal in itself,* but, rather, a capability needed for, or an effective approach to, solving an underlying optimization problem, such as (distributed) resource or task allocation. Hence, the good ways to coordinate are those that enable the agents to effectively solve the underlying optimization problem. Thus the *coalition quality* of the particular set of coalitions and/or the overall coalition structure (i.e., stage (i)), cannot be separated from the problem of *optimization within each coalition* (stage (ii)): to optimize well *precisely means* to form good coalitions with respect to the agents' tasks and/or resources.

We have emphasized in Chapter 2 that our approach to coalition formation, from the game theoretic perspective, assumes the underlying *locally constrained super-additive environments.* In the basic MCDCF algorithm [51], each agent merely tries to form a coalition that is required to be a clique in the underlying graph, and the agent prefers that this coalition (that is, clique) be as large

as possible. Assuming there is no upper bound on the maximal coalition size, the super-additivity property clearly holds for this simplistic problem.

Locally constrained super-additivity still holds when the coalition quality is measured with respect to the *joint capabilities* of the coalition members, as long as each agent's capability vector contains all nonnegative entries, which we have assumed in section 4.1. Other tacit assumptions that we have made, and that suffice to ensure the desired form of the environment's super-additivity, are that either a "single shot" coalition-to-task mapping is performed, or, alternatively, that each agent's resources/capabilities are *renewable* after each task is completed. Hence, in that framework, insofar as the coalition-to-task mapping is concerned, all that matters is that the total resources of agents in the coalition exceed the resource requirements of the desired task – but *by how much* is treated as irrelevant. Once the agent resources are depletable, and an agent is expected to participate in completing several tasks (with each agent or coalition still restricted to being able to work on only one task at a time), on the other hand, these agents would be interested in long-term *planning* [4], in computing the *marginal utility* of servicing each task [37], etc. In these, more general scenarios, relatively simple greedy strategies like the one that our MCDCF algorithm is based upon, need not suffice. Studying the scenarios where super-additivity need not hold and where each agent needs to perform nontrivial planning tasks, however, is beyond our current scope.

Similarly, super-additivity in general does not hold in the non-DPS domains, where each agent has its own individual preferences over the states of the world, and where different agents' preferences need not be compatible with each other. Once each agent has an individual notion of utility, the question of how desirable is a particular candidate coalition becomes more complex: each agent needs to evaluate whether it individually is better or worse off if it joins that coalition [19, 27, 32]. This evaluation will be based upon *what share* of the coalition's overall projected or expected payoff this agent can hope to get. Therefore, once the issue of how is the payoff for servicing the tasks to be distributed among the respective coalition members is brought into the picture (stage (iii) in the aforementioned Sandholm's classification), even the locally constrained super-additivity assumption, in general, becomes hard(er) to justify. However, as already mentioned, in the present work we chiefly focus on the purely cooperative, distributed problem solving MAS domains, and, in particular, we thereby avoid having to address the issue of the payment disbursements altogether.

5.2.1 Main Properties and Strengths of The MCDCF Approach

What are the main properties of the coalition structures likely to arise when MCDCF is invoked on an arbitrary large but sufficiently sparse graph that satisfies the aforementioned assumptions? Once the coalitions are formed according to the algorithm, these coalitions of agents will be tight (as everyone in the coalition can communicate with everyone else *directly*), and therefore as robust and fault-tolerant as possible. This is a highly desirable property involving the coalitions or teams of agents operating in the environments where both the agent failures and the agent-to-agent communication link failures can be expected once these agent coalitions are deployed to service their appropriate tasks.

One example of such a collaborative large-scale MAS application domain are the micro-UAVs deployed in a potentially hostile environment (see subsection 2.2.1). Various aspects of the Open Systems Laboratory's work on modeling, simulating, and designing coordination strategies for micro-UAVs can be found in references [15, 16, 49, 50]. Another application domain where the robustness and fault-tolerance in the presence of individual agent failures are critically important are the large-scale smart sensor networks [18, 21]. This is particularly true when a large number of smart sensors is spread out over a sizable geographic area, and when due to rough terrain, weather conditions, limited battery life and/or other factors, it is to be expected that quite a few smart sensors will simply die during their deployment – yet the entire sensor net as a system needs to keep functioning reliably in spite of the individual sensor failures. Thus, both micro-UAVs and smart sensor nets are among the application domains where high system fault tolerance can be expected to be of utmost importance. We have outlined some of the basic properties of the micro-UAV and the sensor net domains from a MAS perspective in subsection 2.2.1.

The proposed MCDCF algorithm can be used as a subroutine in many multi-agent system scenarios where, at various points in time, the system needs to reconfigure itself, and the agents need to form new coalitions (or transform the existing ones) in a fully distributed manner, where each agent has to reason, act and coordinate with other agents *strictly locally,* and where it is important for the agents to be able to reach consensus on these coalitions efficiently.

Insofar as the resource consumption of MCDCF is concerned, as we have seen in section 5.1, under an appropriate assumption on the network sparseness both the total amount of computation

and the total communication per each agent are going to be low-degree polynomial in the number of agents, and the local memory storage requirements are only polylogarithmic in the number of agents. That is, MCDCF is a highly resource-aware algorithm that, for sufficiently sparse underlying networks, strives to simultaneously minimize all three main cost parameters: the local computation, the local storage, and the amount of communication per agent. Moreover, if an even tighter bound on the network sparseness is imposed, namely, if $K(n)$ from section 5.1 is required to satisfy $K(n) = O(1)$, then it can be seen that each of the three critical algorithmic complexity parameters depend only on the underlying network's *density,* and not on the total number of agents.

Among the strengths of MCDCF, we also mention the following:

– the algorithm is strictly local and, in particular, no flooding of the network is required;

– an agent need not know either the absolute or relative *positions* (that is, locations) of other, near-by agents: all that is required is that each agent can correctly identify those neighboring agents that are within its communication radius;

– in fact, for the purposes of the MCDCF-based coalition formation alone, an agent need not know its own location, either;

– an agent need not know *a priori* anything about other agents, including their total number, or their UIDs: all each agent has to know, is that every agent in the system has to have its unique UID, as well as that those UIDs are totally ordered by the relation $\prec$, and that all agents respect this total order;

– MCDCF is a genuinely dynamic, online algorithm: there is no necessity for any kind of a pre-deployment, static planning stage (see the discussion in Chapter 2 and also reference [12]).

Our final observation on the effectiveness of an MCDCF-based approach to decentralized coalition formation is that the proposed algorithm as a coordination subroutine in large-scale MAS can be expected to be truly useful only when the time scale of significant changes in the inter-agent communication topology is much coarser than the time scale for the coalitions of agents, first, to form according to the algorithm, and, second, once formed, to accomplish something useful in terms of servicing those agents' tasks. In particular, these potential topology changes better be taking place at a pace considerably slower than $2^{K(n)}$, where $K(n)$ is the maximum allowed coalition size. For a more detailed discussion, we refer the reader to [50, 51].

5.2.2 Some Limitations and Weaknesses of MCDCF-based Approaches

While the proposed coalition formation algorithm has several features that are highly desirable in cooperative bounded-resource MAS domains of our interest, it also has certain limitations and weaknesses. Among the fundamental limitations of MCDCF, we emphasize the following:

– MCDCF is scalable only for the sufficiently sparse underlying graphs, or else if an upper bound (that may be quite artificial) is imposed on a maximal coalition size;

– in those dynamic environments, where the communication network topology changes at a pace comparable to or faster than about $2^{K(n)}$ (where, as before, $K(n)$ is the upper bound on the maximal coalition size, that induces an upper bound on the number of MCDCF rounds), the coalition formation process is too slow to keep up with these topology changes, and hence MCDCF cannot be effectively applied;

– while an agent need not know either its own or other agents' location, each agent has to be uniquely identifiable via its UID, which limits the applicability of MCDCF to certain very large scale MAS domains, such as, e.g., those based on the *smart dust* paradigm;

– the amount of communication per agent, although polynomial in the total number of agents, may still be too high and therefore infeasible in certain domains where, due to the available bandwidth and/or energy limitations, one cannot afford the amount of communication to scale linearly or super-linearly with the number of agents;

– there is, in general, no guarantee that the coalition structure resulting from an application of MCDCF is going to be balanced (for instance, in terms of the ratio of the biggest and the smallest resulting coalitions' sizes); to obtain such guarantees, additional criteria need to be introduced insofar as the tie-breaking process is concerned (see Chapter 4).

Insofar as the major weaknesses of the MCDCF coalition formation are concerned, while the algorithm, as observed in the earlier sections, is quite robust with respect to individual agent or link failures, it is, at the same time, rather sensitive to malicious behavior of individual agents. In particular, once the veracity assumption is dropped (see the assumptions in Chapter 4), MCDCF's robustness with respect to Byzantine attacks [20] turns out to be fairly low.

For the illustrative purposes, we show an example where there is a single bad agent that tries to purposefully mislead several good agents into forming coalition with it, thereby abandoning

potentially good coalitions with other good agents. For simplicity, we assume that the only criterion of a coalition's goodness, in addition to being made of "good" agents only, is that it be a clique of as big a size as possible. Consider the scenario in *Figure* 5.1: the central agent, marked by b (for 'bad' or 'byzantine'), is trying to convince the good peripheral agents it can directly communicate with (as indicated by the dashed lines) that they should join the coalition with it. If b succeeds in luring both the agents $\{v_1, v_2\}$ and the agents $\{v_5, v_6\}$, then agent b would succeed in simultaneously creating two fake coalitions, namely, $C_1 = \{v_1, v_2, b\}$ and $C_2 = \{v_5, v_6, b\}$. The first of those two coalitions prevents good three-member coalitions such as $\{v_1, v_2, v_3\}$ and $\{v_8, v_1, v_2\}$ from being created, thereby not only fooling the agents v_1 and v_2 into joining a coalition that will subsequently turn out to be useless, but it also causes v_8 and v_3 to have to settle for what is sub-optimal for each of them individually. Similarly, the second of the two fake coalitions not only misleads v_5 and v_6 into forming a coalition with the bad agent b, but also adversely affects the agents v_4 and v_7. Moreover, given the communication topology, none among the good agents can tell that agent b is a cheater. Indeed, the badness of b cannot be deduced during the coalition formation process itself, but would presumably only be discovered once the coalitions have been already formed and are subsequently deployed, and either both v_1 and v_2, or else both v_5 and v_6, determine that b is cheating (say, by a majority vote). Furthermore, given the topology, no good agent can tell that the resulting coalition structure is not a valid partition of the set of agents: v_1, v_2, v_3 and v_8 think that b has joined C_1; these four good agents know nothing of C_2. Similarly, v_4, v_5, v_6 and v_7 think b has joined C_2, and they know nothing of C_1.

In order for the good agents to be able to discover a "bad apple" such as the agent b in our example, they would need to communicate to each other not only the information about their nearest neighbors, but also about those nearest neighbors' neighbors. However, the increase in robustness to Byzantine attacks that would result from this modification to the original MCDCF algorithm comes at a cost: each agent would now need to both communicate and locally store quadratically more data than in the original MCDCF.

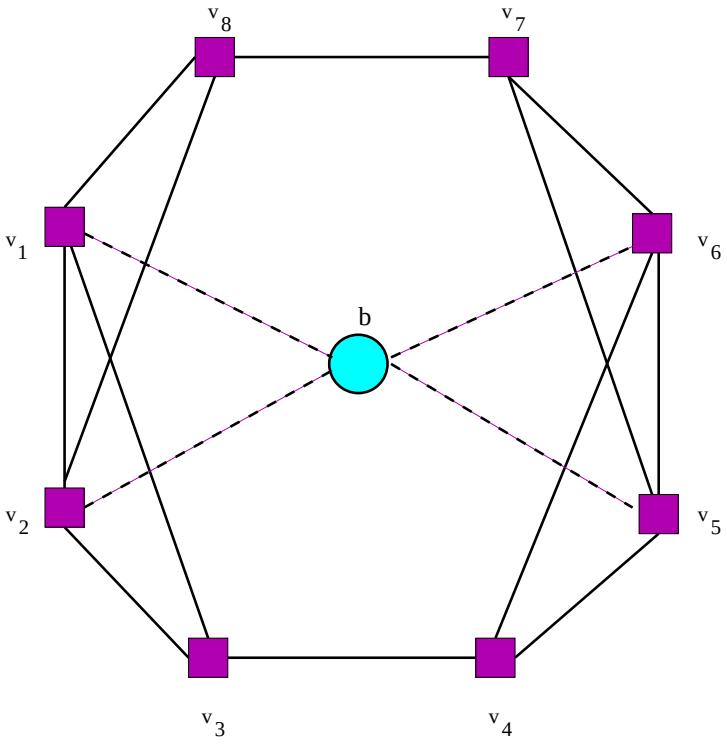

Figure 5.1: An example where a single Byzantine, malicious agent, b, manages to create a major havoc among an ensemble of good agents v_i

5.3 Some Possible Extensions and Generalizations

In this section, we outline some possible directions for the future work.

Insofar as the basic version of MCDCF is concerned, where the only criteria of a coalition's quality stem from the underlying communication network topology, there are several possible extensions to consider. One direction is to require that, instead of necessarily having to be cliques, the legal coalitions have to be connected subgraphs of the underlying communication topology graph that satisfy some other properties. For instance, consider the scenario where only the subgraphs that are k-connected for the specified value of an integer parameter k are allowed as the candidate coalitions. Since, in such subgraphs, there are at least k paths between any two nodes, the resulting agent coalitions would be guaranteed to tolerate the failures of any up to k communication links.

Another possibility is to generalize the "first-neighbor" criterion that characterizes MCDCF (since, in a clique, every node is every other node's first neighbor) to the extended, multi-hop neighborhoods. However, if the candidate cliques are those subgraphs where any two nodes are at most $h \geq 1$ hops apart from each other, then it appears reasonable that the appropriately modified

MCDCF algorithm would require each agent to communicate and locally store the data about all agents that are within h hops from it. This leads to an increase in the communication and storage requirements that is, in general, exponential in the parameter h. For example, for $h = 2$, each agent needs to send messages to, as well as receive and then store messages from, not only its first neighbors, but also those neighbors' first neighbors. If the bound on a maximum neighborhood size is some function $K = K(n)$, this means that the upper bound on the total number of candidate coalitions an agent may have to consider would now be 2^{K^2}.

As already discussed in the earlier chapters, there are many possible notions of *desirable coalitions* that are based on criteria other than those stemming from the communication network topology. In the most general setting, in addition to the communication topology graph of a MAS, one can define what we call an *affinity (hyper)graph* for this system. Such an affinity (hyper)graph would be capturing an *affinity relation* among the agents; this relation could be binary (in the ordinary graph case) or of a more general arity (the hypergraph case). Two or more agents would then constitute a (hyper)edge in this affinity (hyper)graph if and only if they are related by the affinity relation, i.e., intuitively, if they have an affinity for each other to work together as a team or coalition. Moreover, the affinity (hyper)graphs could also be made *weighted,* thereby *quantifying the affinities.* The purpose of quantified affinities would be to capture the fact that, in human societies and many other multi-agent systems, not all affinities among pairs or groups of agents that share an affinity are equally strong.

In this very general framework provided by the *affinity (hyper)graphs,* the constraint optimization formulation of the coalition formation problem would entail two sets of constraints: one related to the physical proximity and bounded communication ranges of agents, and the other, functional, that would capture for any given agent which other agents have an affinity to collaborate with that agent. In particular, the previously discussed issue of mutual compatibility of two (or more) agents' capabilities is one special case of this general notion of affinity among the agents. We leave the study of how to generalize MCDCF to such, more general problem settings for the possible future work.

Our final remark is that an appropriate *weighted hypergraph* version of the MCDCF problem framework, if indeed the underlying hypergraph is allowed to represent an arbitrary, general notion

of affinity among two or more agents, would then provide one of the most general and broadly applicable coalition formation frameworks in the MAS literature.

CHAPTER 6

THESIS SUMMARY

This thesis addresses the general problem of scalable and resource-aware models for agent coordination in collaborative large-scale multi-agent systems, as well as the more specific problem of *dynamic distributed task allocation via coalition formation* as a particular, frequently encountered approach to agent coordination in such systems.

We have motivated, introduced, described and analyzed in some detail a fully decentralized algorithm for coalition formation based on a distributed computation of (maximal) cliques of appropriately bounded sizes in the underlying communication network of agents. We named this algorithm MCDCF, which stands for *Maximal Clique based Distributed Coalition Formation.* We consider the MCDCF algorithm, together with its possible extensions and generalizations that we have briefly addressed in the present report, as well as a detailed discussion about the candidate MAS application domains where an MCDCF-based approach could be fruitfully applied, to be the main contributions of this thesis.

Indeed, it is our hope that the MCDCF algorithm, or one of its appropriately fine-tuned variants, will turn out to be a genuinely useful coordination subroutine in some of those multi-agent system applications, where the interconnection topology of the agents may often change, so that the system needs to dynamically reconfigure itself *repeatedly,* yet where these topology changes are on a time scale that allows agents to (i) form their coalitions, and (ii) do something useful while participating in those coalitions. In particular, while our problem formulation does capture some aspects of the dynamics in MAS environments, MCDCF as presented in this thesis also requires, informally speaking, that these environmental changes take place at a relatively slow pace. In particular, the agent coalitions as formed via MCDCF need to have a chance to perform useful tasks, before

the underlying communication topology of the system changes so much as to render the formed coalitions ineffective or even obsolete.

As for the future work, in addition to extending and generalizing the MCDCF coalition formation framework along some of the directions indicated in section 5.3, we also envision a detailed comparative analysis of the approach presented in this thesis on one, and the coalition formation approaches familiar from the MAS literature, on the other hand. In particular, we would like to compare and contrast the purely peer-to-peer, *genuinely democratic* approaches to multi-agent coordination, where *all agents are made equal* (except possibly for the different capability vectors), with the asymmetric, less democratic and more leader-based coordination approaches (such as, e.g., those based on automated dynamic auctions). Intuitively, the genuinely leaderless mechanisms for coalition formation, such as our maximal clique based approach, are less prone to "bottlenecks" and single points of failure than the coordination strategies where some of the agents are given special roles. However, this intuition needs to be both further theoretically investigated and experimentally validated via appropriate comparative simulations and performance measurements.

REFERENCES

[1] N. M. Avouris, L. Gasser (editors). *"Distributed Artificial Intelligence: Theory and Praxis",* European Courses on Computer and Information Science, vol. 5, Kluwer Academic Publishers, 1992

[2] D. Bernstein, R. Givan, N. Immerman, S. Zilberstein. "The complexity of decentralized control of Markov decision processes", *Mathematics of Operations Research,* vol. 27 (4), pp. 819-840, 2002

[3] C. Boutilier. "Sequential optimality and coordination in multiagent systems", in Proceedings of The Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-02), pp. 478-485, held in Stockholm, Sweden, 1999

[4] C. Boutilier, T. Dean, S. Hanks. "Decision-theoretic planning: structural assumptions and computational leverage", *Journal of Artificial Intelligence Research,* vol. 11, pp. 1-94, 1999

[5] D. H. Cansever. "Incentive Control Strategies For Decision Problems With Parametric Uncertainties", Ph.D. thesis, University of Illinois Urbana-Champaign, 1985

[6] V. Chvatal. "A greedy heuristic for the set-covering problem", in *Mathematics of Operations Research,* vol. 4 (3), pp. 233-235, 1979

[7] I. Curiel. *"Cooperative Game Theory and Applications: Cooperative Games Arising from Combinatorial Optimization Problems",* Kluwer Academic print on demand, Boston, MA, 1997

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest. *"Introduction to Algorithms",* The MIT Press, Cambridge, MA, 1990

[9] S. E. Conry, K. Kuwabara, V. A. Lesser, R. A. Meyer. "Multistage negotiation for distributed constraint satisfaction", *IEEE Transactions on Systems, Man and Cybernetics,* Vol. 21, Num. 6, pp. 1462 - 1477, November 1991

[10] S. Franklin, A. Graesser. "Is it an Agent, or just a Program? A Taxonomy for Autonomous Agents", Proceedings of The Third International Workshop on Agent Theories, Architectures & Languages, Springer-Verlag, 1996

[11] M. R. Garey, D. S. Johnson. *"Computers and Intractability: a Guide to the Theory of NP-completeness",* W. H. Freedman & Co., New York, NY, 1979

[12] C. V. Goldman, S. Zilberstein. "Optimizing Information Exchange in Cooperative Multi-agent Systems", in Proceedings of The Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-03), held in Melbourne, Australia, July 14-18, 2003

[13] J. C. Harsanyi. "A simplified barganining model for n-person cooperative game", in *International Economic Review,* vol. 4, pp. 194-220, 1963

[14] J. C. Harsanyi. *"Rational Behavior and Bargaining Equilibrium in Games and Social Situations",* Cambridge University Press, 1977

[15] M. Jang, S. Reddy, P. Tosic, L. Chen, G. Agha. "An Actor-based Simulation for Studying UAV Coordination", in Proceedings of The Fifteenth European Symposium on Simulation (ESS-03), pp. 593-601, Delft, The Netherlands, October 26-29, 2003

[16] M. Jang, G. Agha. "On Efficient Communication and Service Agent Discovery in Multi-agent Systems", in Proceedings of The Third International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS-04), pp. 27-33, Edinburgh, Scotland, May 24-25, 2004

[17] H. Jung, M. Tambe, S. Kulkarni. "Argumentation as distributed constraint satisfaction: Applications and results", in Proceedings of The International Conference on Autonomous Agents, pp. 324-331, 2001

[18] V. Lesser, C. Ortiz, M. Tambe. "Distributed Sensor Nets: A Multiagent Perspective", Kluwer, 2003

[19] R. D. Luce, H. Raiffa. *"Games and Decisions",* John Willey and Sons, Inc., 1957

[20] N. Lynch. *"Distributed Algorithms",* Morgan Kaufmann Publisher, Wonderland, 1996

[21] P. J. Modi, H. Jung, W. Shen, M. Tambe, S. Kulkarni. "A Dynamic Distributed Constraint Satisfaction Approach to Resource Allocation", in Proceedings of The Seventh International Conference on Principles and Practice of Constraint Programming, 2001

[22] P. J. Modi, H. Jung, W. Shen. "Distributed Resource Allocation: Formalization, Complexity Results and Mappings to Distributed CSPs", technical report, November 2002

[23] P. J. Modi, W. Shen, M. Tambe, M. Yokoo. "An asynchronous complete method for distributed constraint optimization", in Proceedings of The Second International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-03), Melbourne, Australia, July 14-18, 2003

[24] J. von Neumann, O. Morgenstern. *"Theory of Games and Economic Behavior",* Princeton University Press, 1944

[25] C. Papadimitriou. *"Computational Complexity" ,* Addison-Wesley, Reading, Massachusetts, 1994

[26] S. Parsons, M. Wooldridge. "Game Theory and Decision Theory in Multi-Agent Systems", *International Journal of Autononomous Agents & Multi-Agent Systems,* vol. 5, Kluwer, 2000

[27] S. Parsons, P. Gmytrasiewicz, M. Wooldridge. *"Game Theory and Decision Theory in Agent-based Systems",* volume 5 of the *Multiagent Systems International Book Series*, Kluwer Academic Publishers, June 2002

[28] V. Parunak, A. Ward, M. Fleischer, J. Sauter, T. Chang. "Distributed component-centered design as agent-based distributed constraint optimization", in Proceedings of AAAI Workshop on Constraints and Agents, pp. 93-99, American Association for Artificial Intelligence, 1997

[29] M. Pechoucek, V. Marik, J. Barta. "A knowledge-based approach to coalition formation", *IEEE Intelligent Systems*, vol. 17 (3), pp. 17-25, 2002

[30] H. Raiffa. *"Decision Analysis: Introductory Lectures on Choices under Uncertainty",* Addison Wesley, Reading, Massachusetts, 1968

[31] A. S. Rao, M. P. Georgeff. "BDI Agents: From Theory to Practice", Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95), San Francisco, USA, 1995

[32] A. Rapoport. *"N-Person Game Theory",* The University of Michigan Press, Ann Arbor, Michigan, 1970

[33] S. J. Rosenschein, L. P. Kaelbling, "A Situated View of Presentation and Control", *Artificial Intelligence*, vol. 73, 1995

[34] J. Rosenschein, G. Zlotkin. *"Rules of Encounter: Designing Conventions for Automated Negotiations among Computers",* The MIT Press, Cambridge, Massachusetts, 1994

[35] S. Russell, P. Norvig. *"Artificial Intelligence: A Modern Approach",* 2nd ed., Prentice Hall Series in Artificial Intelligence, 2003

[36] T. Sandholm, K. Larson, M. Andersson, O. Shehory, F. Tohme. "Coalition structure generation with worst case guarantees", *Artificial Intelligence*, vol. 111, pp. 209-238, 1999

[37] T. Sandholm and V. Lesser. "Issues in automated negotiation and electronic commerce: Extending the contract net framework", in Proceedings of the First International Conference on Multiagent Systems (ICMAS-95), pp. 328-335, San Francisco, California, June 1995

[38] T. Sandholm, V. Lesser. "Coalitions among Computationally Bounded Agents", *Artificial Intelligence,* special issue on *"Principles of Multi-Agent Systems",* 1997

[39] T. Sandholm, K. Larson, M. Andersson, O. Shehory, F. Tohme. "Anytime Coalition Structure Generation with Worst Case Guarantees", Proceedings of the National Conference on Artificial Intelligence, pp 46-53, Madison, Wisconsin, USA, July 1998

[40] P. Scerri, P. J. Modi, W. Shen, M. Tambe. "Are multiagent algorithms relevant for robotics applications? A case study of distributed constraints algorithms", in ACM Symposium on Applied Computing, 2003

[41] S. Sen, P. S. Dutta. "Searching for optimal coalition structures", in Proceedings of the Fourth International Conference on Multiagent Systems (ICMAS), pp. 287-292, 2000

[42] O. Shehory, S. Kraus. "Coalition formation among autonomous agents: Strategies and complexity", in *From Reaction to Cognition,* Springer's *Lecture Notes in Artificial Intelligence*, vol. 957, pp. 57-72, 1995

[43] O. Shehory, S. Kraus. "Task allocation via coalition formation among autonomous agents", in Proceedings of The 14th International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Quebec, August 1995

[44] O. Shehory, S. Kraus. "Methods for Task Allocation via Agent Coalition Formation", *Artificial Intelligence*, vol. 101 (1-2), pp. 165-200, 1998

[45] O. Shehory, S. Kraus. "Feasible Formation of Coalitions Among Autonomous Agents in Non-superadditive Environments", *Computational Intelligence*, vol. 15 (3), pp. 218-251, 1999

[46] H. A. Simon. *"Models of Man",* J. Willey & Sons, New York City, NY, 1957

[47] R. G. Smith. "The contract net protocol: high-level communication and control in a distributed problem solver", *IEEE Transactions on Computers,* vol. 29 (12), pp. 1104-1113, December 1980

[48] G. Tel. *"Introduction to Distributed Algorithms",* 2nd ed., Cambridge University Press, 2000

[49] P. Tosic, M. Jang, S. Reddy, J. Chia, L. Chen, G. Agha. "Modeling a System of UAVs on a Mission", in Proceedings of The Seventh World Multiconference on Systemics, Cybernetics, and Informatics (SCI-03), invited session on multi-agent systems, pp. 508-514, Orlando, Florida, July 27-30, 2003

[50] P. Tosic, G. Agha. "Modeling Agents' Autonomous Decision Making in Multiagent, Multitask Environments", in Proceedings of The First European Workshop on Multi-Agent Systems (EUMAS-03), Oxford, England, December 18-19, 2003

[51] P. Tosic, G. Agha. "Maximal Clique Based Distributed Group Formation Algorithm for Autonomous Agent Coalitions", Workshop on Coalitions and Teams (W10), within The Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-04), Columbia University, New York City, NY, July 19-23, 2004

[52] P. Tosic, G. Agha. "Towards a Hierarchical Taxonomy of Autonomous Agents", in Proceedings of IEEE Conference on Systems, Man and Cybernetics (CD-Rom), held in The Hague, The Netherlands, on October 10-13, 2004

[53] P. Tosic, G. Agha. "A Fully Distributed Coalition Formation Algorithm for Collaborative Large-Scale Multi-Agent Systems", in Proceedings of The Second European Workshop on Multi-Agent Systems (EUMAS-04), held in Barcelona, Spain, December 18-19, 2004

[54] P. Tosic, G. Agha. "Maximal Clique Based Distributed Coalition Formation for Task Allocation in Large-Scale Multi-Agent Systems", Post-Proceedings of The First International Workshop on Massively Multi-Agent Systems (Kyoto, Japan, December 10-11, 2004), Springer's *Lecture Notes in Artificial Intelligence* series, vol. 3446, pp. 104-120, 2005

[55] For more on the *TRANSIMS* project at the Los Alamos National Laboratory, go to http://www-transims.tsasa.lanl.gov/ (The *'Documents'* link includes a number of papers and technical reports for the period 1995 - 2001)

[56] M. van de Vijsel. "Increasing Realism in Coalition Formation for Multi-Agent Systems", MS thesis, Department of Computer Science, The University of Manitoba, Winnipeg, Manitoba, August 2005

[57] W. Walsh, M. Wellman. "A market protocol for decentralized task allocation", in Proceedings of the Third International Conference on Multi-Agent Systems, 1998

[58] D. J. Watts. *"Small Worlds: The Dynamics of Networks Between Order and Randomness",* Princeton University Press, Princeton, New Jersey, 1999

[59] D. J. Watts, S. H. Strogatz. "Collective dynamics of 'small-world' networks", *Nature* 393, 1998

[60] G. Weiss (editor). *"Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence",* The MIT Press, Cambridge, Massachusetts, 1999

[61] E. Werner. "Toward a theory of communication and cooperation for multiagent planning", in Proceedings of The Second Conference on Theoretical Aspects of Reasoning about Knowledge, pp. 129-143, Pacific Grove, California, March 1988

[62] M. Wooldridge, N. Jennings. "Intelligent Agents: Theory and Practice", *Knowledge Engineering Review,* vol. 10 (2), 1995

[63] M. Wooldridge. *"An Introduction to Multiagent Systems",* John Wiley and Sons Ltd, February 2002

[64] M. Yokoo, K. Hirayama. "Algorithms for Distributed Constraint Satisfaction: A review", in *Autonomous Agents and Multi-Agent Systems,* vol. 3, no. 2, 2000

[65] M. Yokoo. *"Distributed Constraint Satisfaction: Foundation of Cooperation in Multi-agent Systems",* Springer, 2001

[66] G. Zlotkin, J.S. Rosenschein. "Coalition, cryptography and stability: Mechanisms for coalition formation in task oriented domains", in Proceedings of The Twelfth National Conference on Artificial Intelligence (AAAI-94), pp. 432-437, Seattle, Washington, August 1-4, 1994

APPENDIX

Since we assume that the agents are *homogeneous* (as per discussion in Chapter 2), all agents execute the same algorithm, which was described in section 4.2. In this *Appendix,* we provide the pseudo-code for MCDCF. The computational and communication cost analysis in section 5.1 is directly based on the version of the MCDCF algorithm given below. The present version of the MCDCF algorithm is essentially the same as the one found in [54]. Two earlier versions of MCDCF (including the pseudo-codes) can be found in references [51, 53].

Pseudo-Code for MCDCF Algorithm

Notation:

$i :=$ the i-th agent (node) in the system (say, $i = 1, ..., n$)

$V(i)$:= the i-th node's UID

$N(i)$:= the list of neighbors of node i

$L(i)$:= the extended neighborhood list (i.e., $L(i) = N(i) \bigcup \{i\}$)

$C(i, j) = L(i) \bigcap L(j)$

$C(i)$:= the coalition of choice of node i at the current stage (i.e., one of the $C(i, j)$'s)

$R(i) = (R_1(i), ..., R_s(i))$ is the *resource (or capability) vector* of agent i

(where there are $s \geq 1$ distinct, independent resources, for some $s = O(1)$)

$choice(i) :=$ the choice flag of node i

$dec(i) :=$ the decision flag of node i

Remark: For simplicity and clarity, we focus on distributed computation of (maximal) cliques, and on reaching consensus on what clique-like coalitions are to be formed. We therefore assume that there is an efficient, readily available subroutine for each agent to evaluate (or estimate) the

utility value of each potential coalition, based on each agent's resource vector $R(i)$ and the resource demands of those tasks that are known to agents. This subroutine is called independently by each agent within *Stage 2* below, and is not explicitly captured in the pseudo-code.

MAXIMAL CLIQUE-BASED DISTRIBUTED COALITION FORMATION ALGORITHM:

Stage 0:

DOALL $i = 1..n$

[$*$ each agent i executes locally, asynchronously and in parallel with all other agents $*$]

sort the list $L(i)$

send $[V(i), L(i), R(i), choice(i) = 3, dec(i) = 0]$ to each of your neighbors $j : j \in N(i)$

receive $[V(j), L(j), R(i), choice(j), dec(j)]$ from all $j : j \in N(i)$

END DOALL

Stage 1:

DOALL $i = 1..n$

FOR each $j \in N(i)$ DO

compute $C(i, j) \leftarrow L(i) \bigcap L(j)$

END FOR

END DOALL

$round \leftarrow 1$;

WHILE *(not all agents have joined a coalition)* DO

[$\star$ *Stages 2-5* are repeated until consensus on the coalition structure is reached $\star$]

Stage 2:

DOALL $i = 1..n$

IF $(round == 1)$ THEN

FOR all $j \in N(i)$ DO

node i computes *utility value* $val[C(i, j)]$ of each candidate coalition $C(i, j)$

sort $C(i, j)$ for $j \in C(i)$ with respect to $val[C(i, j)]$

END FOR

END IF

IF $(round > 1)$ THEN

FOR all $j \in N(i)$ DO [$\star$ check if $dec(j) = 1$ $\star$]

IF $(dec(j) == 1)$ THEN

delete $[V(j), C(j), R(j), choice(j), dec(j)]$;

delete j from $L(i)$ and $C(i, j'),\ \forall j' \in N(i) - \{j\}$

END FOR [$\star$ end of FOR loop $\star$]

FOR all $j \in N(i)$ DO [$\star$ FOR all remaining (undeleted) indices j $\star$]

update $C(i, j)$ and $val[C(i, j)]$

END FOR [$\star$ end of FOR loop $\star$]

END IF

END DOALL

Stage 3:

DOALL $i = 1..n$

pick $C(i, l)$ such that $val[C(i, l)] = max_{j \in N(i)}\ val[C(i, j)]$ (subject to $|C(i, j)| \leq K$)

$C(i) \leftarrow C(i, l)$;

IF *(there is more than one such choice of max. utility value)* THEN

set $choice(i) \leftarrow 2$;

ELSE *(if there are other choices $C(i, j')$ but only of strictly smaller utility value)*

set $choice(i) \leftarrow 1$;

ELSE *(if node i has no alternatives left for a satisfactory coalition)*

set $choice(i) \leftarrow 0$;

END IF

END DOALL

Stage 4:

DOALL $i = 1..n$

send $[V(i), C(i), choice(i), dec(i) = 0]$

receive $[V(j), C(j), choice(j), dec(j)]$ from (some of the neighbors) $j : j \in N(i)$

END DOALL

Stage 5:

DOALL $i = 1..n$

compare $C(i)$ with $C(j)$ received from one's neighbors $j \in N(i)$;

IF (*there exists a clique* $\{i, j_1, j_2, ...j_l\}$ *s. t.* $C(i) = C(j_1) = ... = C(j_l)$) THEN

set $dec(i) \leftarrow 1$ (an agreement has been reached);

broadcast coalition $G = \{i, j_1, j_2, ..., j_l\}$ and $dec(i) = 1$ to all neighbors $j \in N(i)$:

send $[V(i), C(i) = G, choice(i), dec(i) = 1]$

ELSE (*based on UID* i *and the priority as defined by the relation* $\prec$)

either DO NOTHING

or change your mind: $C(i) \leftarrow C^{new}(i, j)$

(a new coalition proposal from the list of still available candidate coalitions)

END IF

END DOALL

$round \leftarrow round + 1$

END DO $[* \; end \; of \; WHILE \; loop \; *]$

VITA

Predrag T. Tošić was born and raised in Belgrade, Serbia & Montenegro. He started his undergraduate university education at University of Maryland Baltimore County (UMBC) in 1992, where he subsequently completed B.S. (mathematics and physics) in May, 1994, and M.S. in applied mathematics in August, 1995. After completing M.S. in pure mathematics at University of Illinois at Urbana-Champaign (UIUC) two years later, and teaching and tutoring for a semester, he joined the computer science M.S./Ph.D. program at UIUC in January of 1998. Having spent some fifteen months away from Illinois and the world of academia (May 2000 - August 2001), Predrag returned to UIUC for the completion of his M.S. and Ph.D. degrees in computer science in the fall of 2001.

This M.S. thesis summarizes Predrag's work on decentralized coordination via coalition formation in multi-agent systems. This research, supervised by professor Gul Agha and supported by the DARPA TASK program, was conducted at UIUC during 2002 – 2004. In addition to the large-scale collaborative multi-agent systems that are the main subject of this thesis, and distributed artificial intelligence in general, Predrag's research interests also span formal methods, theoretical computer science, parallel and distributed computing, and complex dynamical systems. As of the spring of 2006, he has authored over twenty peer-reviewed research publications in those areas. He is expecting to defend his doctoral dissertation in the summer of 2006.