Diagnosing Coordination Faults in Multi-Agent Systems

Meir Kalech
Department of Computer Science

Ph.D. Thesis

Submitted to the Senate of Bar-Ilan University
Ramat Gan, Israel
January 2007

This work was carried out under the supervision of Dr. Gal A. Kaminka, Department of Computer Science, Bar-Ilan University.

Acknowledgments

Writing this acknowledgement, I realize that the list of people I should be thanking is quite long. Obviously, the first person I would like to express my gratitude to is to my advisor Dr. Gal A. Kaminka. I would like to gratefully and sincerely thank Gal for his guidance, understanding, patience, and most importantly, his friendship during my Ph.D. studies. It was an enjoyable and very helpful experience to work with Gal. At this point I would like to also thank Oshra, Gal's wife and very close friend, which enabled and encouraged Gal to guide me during many hours, days and nights.

Next, I would like to thank Prof. Sarit Kraus. Although Sarit was not directly involved in my thesis, her silent support (in matters concerning scholarships, publications and travels) took some of the pressure off my shoulders. Also, I would like to thank Prof. Moshe Koppel, Prof. Tommy Klein and Dr. Merav Hadad for their assistance in the first part of my Ph.D.. I would also like to especially thank Prof. Amnon Meisels and Roie Zivan for the many discussions about the first part of the dissertation. These discussions were very helpful in designing this part.

I would also like to thank all of the members of the Maverick research group, especially to Yehuda Elmaliah who helped me to operate the robots in the lab, to Efrat Manistersky for her assistance in the proofs, to Avi Rosenfeld and Sarah Simani for their assistance with the English and to Tom Shpigelman who was always happy to help me fix my computer.

I would like to acknowledge the Schupf Fund which supported me and enabled me to focus on the Ph.D. thesis. Also, I would like to thank Dafna, Hilla, Sylvie and the Administrative Stuff from the Computer Science Department. Indeed, their help in administrative matters, supplementary to the research itself, eased up the long period of my studies in Bar-Ilan.

Finally, and most importantly, I would like to thank my wife Ravit. Ravit is the first to encourage me to do everything that I like to do, and in this case she pushed me to do the Ph.D. Her support, encouragement, patience and her love were unlimited and helped me very much to accomplish my Ph.D.. At this point I would also like to thank my children, Oran, David and Barak. I know that my absence from home and my unstable moods were very difficult for them and for my wife. I appreciate their patience. I thank my parents, Eliko and Mira, that encouraged me to join the Ph.D. studies. Special thanks to my father that always took an interest in my thesis and contributed from his wide breadth of knowledge. I would also like to thank Ravit's mother, Shula. Her assistance and my mother's assistance in taking care of my children gave me the quiet to focus on my studies.

Contents

List of Figures

List of Algorithms

Abstract

With increasing deployment of multiple agents in complex, dynamic settings, there is an increasing need to respond to failures that occur in the agents coordination. In particular, there is need to detect and diagnose *coordination failures*—failures to maintain relationships between agents. We refer to this type of diagnosis as *social diagnosis*. Previous approaches to diagnosis in multi-agent settings have either ignored failures in coordination, or utilized heuristic approaches which do not scale up as the number of agents (and their interactions) increases.

This dissertation offers a comprehensive and principled approach to social diagnosis. We use a model-based diagnosis (*MBD*) approach. Here, a model of a diagnosed system is used to simulate the behavior of the system given the operational context (typically, the system's inputs), and to pinpoint possible failing components within the system. MBD has been difficult to apply to diagnosing coordination failures, because of the challenges in constructing a model of coordination, and the lack of appropriate (scalable) diagnosis algorithms.

In the first part of this dissertation we formally show how to construct a model of agent coordination, and use it to formally define the two key variant social diagnosis problems: the Consistency-based diagnosis problem, and the abductive diagnosis problem. We show that these are NP-Hard problems. We then build on known methods in constraint-satisfaction problems, to provide several algorithms for social diagnosis in centralized and distributed settings. The algorithms—whose analytical guarantees vary in terms of completeness and correctness—are evaluated empirically, in experiments carried out with teams of physical robots. We examined the computational requirements of the algorithms (i.e., their run-time and bandwidth usage), and the correctness of the diagnoses produced. We find that in general, a trade-off exists between computational costs and the correctness of the diagnosis.

In the second part of the dissertation, we focus on a particular type of coordination failures—*disagreements*—which are of particular interest when talking about teams. We examine the design space of disagreement diagnosis algorithms for a more complex class of situated agents (compared to the first part). We distinguish two phases of diagnosis: (i) the selection of the diagnosing agents; and (ii) the diagnosis of the global team state (by the selected agents). We provide alternative algorithms for these phases, and empirically evaluate their communications and run-time requirements. The results show that centralizing the disambiguation process is a key factor in improving communications, but is not a determining factor in run-time.

On the other hand, explicit reasoning about the different agents is a key factor in determining run-time.

Based on this conclusion we follow two principles in reducing the communications and the computations in large-scale teams. First, we modify the algorithms such that they communicate partial information that is most relevant to the diagnosis. Second, we enable diagnosis of only a limited number of representative agents instead of diagnosing all others. These principles yield a novel diagnosis method which significantly reduces the runtime, while keeping communications overhead to a minimum.

Chapter 1

Introduction

With increasing deployment of robotic and agent teams in complex, dynamic settings, there is an increasing need to also be able to respond to failures that occur in multi-agent teams [Jennings, 1995, Tambe, 1997, Parker, 1998, Kaminka and Tambe, 2000]. One type of failure of particular interest in multi-agent systems is a *coordination fault*, where agents come to disagree on salient aspects of their joint task.

There is thus a particular need to be able to detect and diagnose the causes for coordination faults that may occur, in order to facilitate recovery and reestablishment of collaboration, e.g., by negotiations [Kraus *et al.*, 1998]. This type of diagnosis is called *social diagnosis*, since it focuses on finding causes for failures to maintain social relationships [Kaminka and Tambe, 2000], i.e., coordination failures.

Unfortunately, social diagnosis has not been adequately addressed in previous work regarding diagnosis in multi-agent settings. Some previous investigations focus solely on detection, without diagnosis [Kaminka and Tambe, 2000, Poutakidis *et al.*, 2002]. Others take a heuristic approach which can fail without providing a correct diagnosis [Kaminka and Tambe, 1998]. Others still take a causal-model approach, in which possible faults are specified in advance [Klein and Dellarocas, 1999, Horling *et al.*, 1999, Dellarocas and Klein, 2000, Horling *et al.*, 2001]; this approach does not scale as the number of possible interactions increases. Thus social diagnosis remains an open challenge in several aspects.

First, a significant body of work exists on principled, model-based diagnosis (MBD) in artificial intelligence, but remains untapped in multi-agent systems. MBD tackles the problem of identifying faulty components in a system by observation [Reiter, 1987, de Kleer and Williams, 1987, Davis and Hamscher, 1988]. It relies on a model of the diagnosed system to detect failures and to pinpoint possible failing components within the system. While MBD has been applied on diagnosis in multi-agent and distributed systems (e.g., [Fröhlich *et al.*, 1997, Roos *et al.*, 2003a, Lamperti and Zanella, 2003]), it was always used to diagnose intra-agent failures, rather than coordination failures (which take place inter-agent). MBD has been difficult to apply on diagnosing coordination failures, because of the challenges in constructing a model of coordination, and the lack of appropriate (scalable) diagnosis algorithms.

In the first part of this dissertation we formally show how to construct a model of agent coordination, and use it to formally define the two key variant social diagnosis problems: Consistency-based diagnosis, and abductive diagnosis. We show that these are NP-Hard problems. We then build on known methods in constraint-satisfaction problems, to provide several algorithms for social diagnosis in centralized and distributed settings. The algorithms—whose analytical guarantees vary in terms of completeness and correctness—are evaluated empirically, in experiments carried out with teams of physical robots. We examined the computational requirements of the algorithms (i.e., their run-time and bandwidth usage), and the correctness of the diagnoses produced. We found that in general, a trade-off exists between computational costs and the correctness of the diagnosis.

In the second part of the dissertation, we focus on a particular type of coordination failure—*disagreements*—which is of particular interest in teams. We examine the design space of disagreement diagnosis algorithms for a more complex class of situated agents (compared to the first part). We distinguish two phases of diagnosis: (i) selection of the diagnosing agents; and (ii) diagnosis of the global team state (by the selected agents). We provide alternative algorithms for these phases, and empirically evaluate their communications and run-time requirements. The results show that centralizing the disambiguation process is a key factor in improving communications, but is not a determining factor in run-time. On the other hand, explicit reasoning about the different agents is a key factor in determining run-time.

Based on this conclusion we follow two principles in reducing the communications and the computations in large-scale teams. First, we modify the algorithms such that they communicate partial information that is most relevant to the diagnosis. Second, we enable diagnosis of only a limited number of representative agents instead of diagnosing all the others. These principles yield a novel diagnosis method which significantly reduces the runtime, while keeping communications overhead to a minimum.

1.1 Motivation

Coordination (e.g., on a joint plan or goal) is a key to establishment and maintenance of teamwork [Cohen and Levesque, 1991, Jennings, 1995, Grosz and Kraus, 1996, Tambe, 1997]. The *Joint Intentions* framework [Cohen and Levesque, 1991] focuses on agreement (mutual belief) in a team's joint goal. The *SharedPlans* framework [Grosz and Kraus, 1996] relies on an intentional attitude, in which an individual agent's intention is directed towards a group's joint action. This includes mutual belief and agreement among the teammates in a complete recipe including many actions. Similarly, the *Joint Responsibility* model [Jennings, 1995] establishes the team-members mutual belief in a specific recipe as a corner-stone for their collaboration.

There exist several architectures for building agents, using ideas from teamwork theories; coordination on specific features of the agents' internal states plays a critical role in all. GRATE* implements the joint responsibility model [Jennings, 1995] in industrial agent systems. STEAM [Tambe, 1997] and TEAMCORE [Pynadath *et al.*, 1999] use ideas from both Joint Intentions and SharedPlans, and add reactive team plans which are selected or deselected by a team or sub-team. BITE [Kaminka *et al.*, 2004] follows in this tradition, and additionally allows for a variety of coordination-synchronization protocols to be used interchangeably, in controlling physical robots.

In all of these architectures, despite all efforts, application in complex, dynamic settings, can sometimes lead to *coordination failures* among team-members, e.g., due to sensing failures, or different interpretations of sensor readings [Kaminka and Tambe, 1998, Dellarocas and Klein, 2000]. The function of a diagnosis process is to go from the *detection* of the coordination fault (where an alarm is raised when a coordination fault occurs), to the fault *identification*, where the causes for the coordination fault are discovered. Such failures could be a result of differences in sensor readings or interpretation, in sensor malfunctions, or communication difficulties.

Diagnosis is an essential step beyond the detection of the failures. Mere detection of a failure does not necessarily lead to its resolution. First, because the agents that caused a fault are not necessarily those that detected it, and may thus be unaware of it. Therefore, they may not be able to replan around it. Second, even if somehow an undiagnosed (though detected) fault manages to temporarily overcome the failure—it may still continue to occur in various forms, if its causes are not resolved, e.g., via negotiations [Kraus *et al.*, 1998]. In dynamic domains, such as a RoboCup soccer game, it may indeed be more effective (for a short while) to simply replan rather than engage in diagnosis (if the cost of replanning is cheaper than diagnosis, and if it is possible without knowing the causes for the coordination fault). However, even in such settings, it often makes sense to use the diagnosis post-hoc to discover the reasons for any failures; for instance, in conducting a post-game analysis.

1.2 Contributions

While the problem of detection has been addressed in the literature, e.g., [Kaminka and Tambe, 1998, Poutakidis *et al.*, 2002, Klein and Dellarocas, 1999], diagnosis of coordination faults remains an open issue. The contribution of this dissertation is by formalizing social diagnosis and by providing centralized and distributed algorithms to compute diagnosis in small and large scale teams.

In this thesis we focus on diagnosis algorithms that produce a set of hypothesized faulty agents. We do not address probabilities on the type of the faults of the agents, nor on the type or amount of the faulty agents. For instance, we do not prefer a diagnosis which hypothesizes a single faulty agent over a diagnosis that hypothesizes multiple faulty agents. In addition, we do not address problems like how to repair the faults automatically or how to recover the system. This pioneer thesis lays the foundations to the diagnosis problem in teams of multi-agent systems.

Social diagnosis is an open challenge in several aspects. First, this problem has not been formalized in terms of MBD. Second, most of the work done on this subject presents centralized algorithms; however, multi-agent systems are distributed systems therefore distributed diagnosis is more appropriate. Third, unlike centralized systems, in multi-agent systems questions like which agent makes the diagnosis, and how it disambiguates the agent's state, have not been addressed. A design space for social diagnosis algorithms that tackles these questions is necessary. Our last challenge is examining social diagnosis in large-scale teams. Algorithms for small teams, may not be appropriate for teams with a large number of agents since they can be computationally expensive in practice, in terms of communications and run-time. In the next sections we will describe each one of these challenges in detail.

1.2.1 Formalizing Social Diagnosis

MBD is increasingly being applied in distributed and multi-agent systems (e.g., [Fröhlich *et al.*, 1997, Roos *et al.*, 2003a, Lamperti and Zanella, 2003]). However, while successfully addressing key challenges, MBD has been difficult to apply on diagnosing coordination failures [Micalizio *et al.*, 2004]. This is because many such failures take place at the boundaries between the agent and their environment, including other agents. For instance, in a team, an agent may send a message that another agent, due to a broken radio, did not receive. As a result, the two agents come to disagree on an action to be taken. Lacking an omniscient diagnoser that knows of the sending of the message, the receiver has no way to detect and diagnose its fault, since the context—the message that can be fed into a model of the radio of both agents—is unobservable to the diagnoser.

Surprisingly, it is still often possible to detect and diagnose coordination failures, given the actions of agents, and the coordination constraints that should ideally hold between them. In the example above, knowing that the two agents should be in agreement as to their actions, and seeing that their actions are not in agreement, is sufficient to (1) show that a coordination failure has occurred; and (2) to propose several possible diagnoses for it (e.g., the first agent did not send a message, the second agent did not receive it, etc.).

This thesis takes a first step towards addressing the open challenge of formalizing diagnosis of coordination (*inter-agent*) failures in terms of model-based techniques. We model the coordination between agents as a graph of concurrence and mutual exclusion constraints on agents' actions. The diagnosis process begins with an observation of the agents' actions and inferring, by comparing the minimal number of agents that deviate from the expected coordination (i.e., a minimal set of emphabnormal agents) to the coordination model. We argue that this minimality criteria is inherent to diagnosis of multi-agent systems, and that it must be treated separately from classic criteria, such as minimality in number of failing components within the system.

The formalization presented in Chapter 4, allows definition of both consistency-based and abductive diagnosis problems, and points at several approaches to their solution. While the formalization does not commit to centralized or distributed diagnosis settings, the initial methods we provide are centralized. For consistency-based diagnosis, we show that we can map the minimal vertex cover to the problem of computing the coordination diagnosis. For abductive diagnosis, we take an approach based on satisfiability. Both of these problems are thus NP-Hard.

1.2.2 Distributed Social Diagnosis

Previous work in diagnosis of coordination failures has focused on centralized methods for such diagnosis [Lamperti and Zanella, 2003, Micalizio *et al.*, 2004, Ardissono *et al.*, 2005]. Unfortunately, centralized methods suffer from key limitations: First, they can be computationally expensive in practice, in terms of communications and run-time. Second, they rely on a single diagnoser, and thus risk a single point of failure. Moreover, this assumes no communication limitations, e.g., range. Finally, they do not necessarily exploit the different knowledge of different agents; e.g., an intended receiver faces difficulty detecting that a message to it was lost, where the sender may do it more readily. Indeed, distributed methods that have been proposed, e.g., [Roos *et al.*, 2003b] do not address coordination failures. In Chapter 5 we present distributed algorithms to compute abductive social diagnosis based on the formalism presented in Chapter 4. We show that modeling the coordination as a constraint graph brings to bear solution methods from distributed constraint satisfaction (DisCSP) literature, as solutions to the constraint graph form the basis for diagnoses.

We present five distributed model-based diagnosis algorithms to compute the diagnosis, based on DisCSP algorithms. While the reasoning behind all this is the same as outlined above, the algorithms differ from each other with respect to their expected run-time (based on DisCSP literature) and their completeness of the diagnoses (based on whether they find all or a single DisCSP solution). Two of the algorithms (based on *synchronous backtracking*) are expensive, but compute a complete set of minimal diagnoses . One algorithm, (*asynchronous backtracking*) is expected to be computationally cheaper, and guarantees computing a single diagnosis (though not necessarily minimal). The last two algorithms (*distributed stochastic search* and *distributed breakout search*) are local search algorithms that are not guaranteed to find a diagnosis, but are known to be highly effective (and cheapest of the above) in solving DisCSPs in practice [Yokoo and Hirayama, 2000, Zhang *et al.*, 2005].

We evaluated the use of these algorithms in comprehensive experiments with a team of physical and simulated Sony AIBO robots, experiencing systematic coordination failures. We examined the computational requirements of the algorithms (i.e., their run-time and bandwidth usage), and the correctness of the diagnoses produced. We find that in general, synchronous backtracking methods that compute the entire space of minimal diagnoses are naturally more expensive than others, though they produced better diagnosis results. However, a surprising result is that the local search algorithms (which typically outperforms asynchronous backtracking methods in DisCSPs) show only mediocre results, both in terms of the quality of the diagnosis, as well as in terms of computational requirements.

1.2.3 A Design Space for Social Diagnosis

Naive implementations of social diagnosis processes can require significant computation and communications, which prohibits them from being effective as the number of agents is scaled up, or the number of failures to diagnose increases. Previous work did not rigorously address this concern [Dellarocas and Klein, 2000, Horling *et al.*, 1999, Fröhlich *et al.*, 1997, Roos *et al.*, 2001, Roos *et al.*, 2003b, Roos *et al.*, 2004]. In Chapter 7, we seek to examine in depth the communication and computation requirements of social diagnosis.

To explore these phases concretely, we focus on teams of situated (behavior-based) agents [Firby, 1987, Newell, 1990, Mataric, 1998, Tambe, 1998]. Also, we focus on one kind of coordination faults—disagreements between teammates. The control process of such agents and faults is relatively simple to model, and we can therefore focus on the core communications and computational requirements of the diagnosis.

We distinguish two phases of social diagnosis: (i) the selection of the diagnosing agents; and (ii) the diagnosis of the team state (by the selected agents). We provide alternative algorithms for these phases, and combine them in different ways, to present six diagnosis methods, corresponding to different design decisions. We then examine the runtime and communication complexity and empirically evaluate these parameters in diagnosing thousands of systematically-generated failure cases, occurring in a team of behavior-based agents in two different complex domains.

We draw general lessons about the design of social diagnosis algorithms from the empirical results. Specifically, the results show that centralizing the disambiguation process is a key factor in dramatically improving communications efficiency, but is not a determining factor in runtime efficiency. On the other hand, explicit reasoning about other agents is a key factor in determining runtime: Agents that reason explicitly about others incur significant computational costs, though they are sometimes able to reduce the amount of communications. These results contrast with previous work in disagreement detection, in which distributed algorithms reduce communications (and to some extent, runtime) by reasoning about other agents [Kaminka and Tambe, 1998].

1.2.4 Social Diagnosis in Large-Scale Teams

In large scale teams the problem of high communication and computation overhead becomes much more serious, since most of the social diagnosis methods are exponential. In addition, in large multi-agent systems it is necessary to reduce the communication due to security, bandwidth limitations, and reliability concerns.

Unfortunately, previous works of diagnosing multi-agent systems do not address large-scale teams, in which both communications and runtime must be tightly managed. Some rely on fault models and exceptions (e.g., [Horling *et al.*, 2001, Micalizio *et al.*, 2004]), which explode combinatorially as the number of agent relations grow. Previous work on large-scale systems did not address social diagnosis, instead it focused on fault detection [Kaminka and Bowling, 2002], or coordination [Durfee, 2001, Scerri *et al.*, 2005a, Scerri *et al.*, 2005b].

We seek to enable social diagnosis in large-scale teams of behavior-based agents. Since we want to examine the communication and computation, we focus on teams of situated (behavior-based) agents and on disagreement faults between teammates. We first develop techniques which use communications earlier in the diagnosis process (compared to the algorithms presents in Chapter 7), in an attempt to stave off both the runtime associated with the generation of the diagnostic hypotheses, as well as later communications. These techniques include: (i) using initial queries to alleviate diagnostic reasoning (BEHAVIOR QUERYING); (ii) using communications in light-weight behavior recognition to focus on relevant beliefs that may be in conflict (SHARED BELIEFS).

These "communicate early" techniques enable a third method (GROUPING) in which the diagnosed agents are divided into groups based on their selected behavior and their role, such that all members of a group are in agreement, and at least one disagreement fault exists between any two groups. Then, only representative agents of each group are diagnosed, and the results are used for others in their group. By using grouping, we limit the required communication and computation which is done only among the representative agents, and thus make the approach applicable to large teams.

We empirically examine these techniques in two domains through thousands of tests, measuring the number of messages, and reasoning runtime. We find that

BEHAVIOR QUERYING reduces both runtime and communications. However, the SHARED BELIEFS technique does not scale well. Moreover, when combined, these techniques do not reduce communications nor runtime. Surprisingly, however, the GROUPING method (which is enabled by this disappointing combination), results in a diagnosis process which is highly scalable in both communication and computation.

1.3 Publications

Subsets of the results that appear in this dissertation were published in the proceedings of the following refereed journals, conferences and workshops:

Formal Model of Social Diagnosis:

- [Kalech and Kaminka, 2005b] Kalech, M. Kaminka, A. G., Towards Model-Based Diagnosis of Coordination Failures, *the Twenty National Conference on Artificial Intelligence (AAAI-05)*, 2005. An earlier version appeared in the Sixteenth International Workshop on Principles of Diagnosis (DX-05), 2005

Distributed Social Diagnosis:

- [Kalech and Kaminka, 2006a] Kalech, M. Kaminka, A. G. Meisels, A. Elmaliah, Y., Diagnosis of Multi-Robot Coordination Failures Using Distributed CSP Algorithms, *The Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, 2006. An earlier version appeared in The 3rd Monet Workshop on Model-Based Systems, The 17th European Conference on Artificial Intelligence (ECAI-06), 2006

Design Space for Social Diagnosis:

- [Kalech and Kaminka, 2003] Kalech, M. Kaminka, A. G. On the Design of Social Diagnosis Algorithms for multi-agent Teams, *the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, Acapulco 2003

- [Kalech and Kaminka, 2006b] Kalech, M. Kaminka, On the Design of Coordination Diagnosis Algorithms for Teams of Situated Agents, accepted to *Artificial Intelligence Journal* (in revision), 2006

Social Diagnosis for large-scale Teams:

- [Kalech and Kaminka, 2005b] Kalech, M. Kaminka, A. G., Diagnosing a Team of Agents: Scaling-Up, *the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-05)*, 2005 An earlier version appeared in the Fifteenth International Workshop on Principles of Diagnosis (DX-04), 2004

- [Kalech and Kaminka, 2006c] Kalech, M. Kaminka, A. G., Coordination Diagnosis Algorithms for Teams of Situated Agents: Scaling-Up, *Autonomous Agents and multi-agent Systems (JAAMAS)* (submitted), 2006

1.4 Dissertation Organization: An Overview

This dissertation is constructed of ten chapters, organized in two main parts. This chapter constitutes the introduction to this thesis, while the next chapter surveys the related work.

The first part I, introduces model-based diagnosis (Chapter 3) which is used as the basis for this thesis. Then in Chapter 4 we present the formalism of social diagnosis in terms of model-based diagnosis. Social diagnosis is formalized both in consistency-based approach as well as an abductive approach. Based on this formalism, in Chapter 5 we continue to develop a distributed approach for the social diagnosis using distributed CSP algorithms. Chapter 6 brings this part to the end by evaluating the distributed algorithms in terms of communication, run-time and correctness.

The second part II focuses on disagreement faults in teams of situated agents. Chapter 7 draws lessons about a design space for social diagnosis algorithms, for such teams. Then, Chapter 8 copes with teams with a large number of agents. For such teams the algorithms which are presented in Chapter 7 could not be scaled. Chapter 9 evaluates the algorithms presented in Chapters 7 and 8. Chapter 10 summarizes this work and presents new directions for future work.

Chapter 2

Related Work

The work presented in this thesis touches on different areas of related research. These are discussed in separate sections below. Section 2.1 addresses previous investigations in diagnosis of distributed systems and multi agent systems. Sections 2.2 and 2.3 focus on related work in distributed diagnosis and diagnosis of large-scale systems correspondingly. Section 2.4 presents several heuristic approaches to diagnosis and Section 2.5 summarizes.

2.1 Diagnosis of Distributed and Multi-Agent Systems

While diagnosis of a single-agent system is relatively well understood and known to be computationally difficult [Hamscher *et al.*, 1992], diagnosis of distributed systems and multi-agent systems remain an open area of research. Two approaches in this area have been explored in the literature: a fault-based approach that models the predicted faults of the system, and a model-based approach that models the system's normal operation. Most of the work in these approaches does not explore the diagnosis of coordination failures.

2.1.1 Fault-Based Approach

Dellarocas and Klein [Klein and Dellarocas, 1999, Dellarocas and Klein, 2000] report on a system of domain-independent exceptions handling services. A first component contains a knowledge base of generic exceptions. A second component contains a decision tree of diagnoses; the diagnosing process is done by traversing down the tree by asking queries about the relevant problem. A third component is responsible for seeking a solution to the exception, based on a resolution knowledge base. This approach transfers the failure-handling responsibility from the agent to an external system, to alleviate the load on each agent designer (which would now be freed of the responsibility of implementing an exception-handling system in each agent). However, in contrast to our work, communication and runtime concerns are not addressed. In their system *sentinel agents* monitor the agents in the multi-agent system and proactively query agents about their status. They do not mention the monitoring method and when a querying is necessary, but both of those actions have a large impact on communication and computation complexity.

Similarly, Horling et al. [Horling *et al.*, 1999] uses a fault-model of failures and diagnoses to detect and respond to multi-agent failures. In this model a set of predefined diagnoses are stored in acyclic graph's nodes. When a fault is detected a suitable node is triggered and according to the fault characters the node activates other nodes along the graph. The advantage of Horling's fault-model system over Dellarocas and Klein's system is the use of a learning algorithm that can be employed to maintain structure as time passes. As with Dellarocas and Klein, Horling's work does not explicitly address communication complexity.

Based on the discrete-event system approach [Sampath *et al.*, 1995, Sampath *et al.*, 1996], Pencolé et al. [Pencolé *et al.*, 2002] and Lamperti and Zanella [Lamperti and Zanella, 2003] that uses a fault-model approach, where the distributed system is modeled as a discrete event system, and the faults are modeled in advance. The diagnoser infers unobservable faulty events by computing possible paths in the discrete event system that match observable events. A common theme in all of these is that they require pre-enumeration of faulty interactions among system entities. However, in multi-agent systems, these are not necessarily known in advance since they depend on the specific run-time conditions of the environment, and the actions taken by the agents.

2.1.2 Consistency-Based Approach

Fröhlich et al. [Fröhlich *et al.*, 1997] and Letia and Netin [Letia and Netin, 1999, Letia *et al.*, 2000] suggest dividing a spatially distributed system into regions, each under the responsibility of a diagnosing agent. If the fault depends on two regions the agents that are responsible for those regions cooperate in making the diagnosis. This method is inappropriate for dynamic team settings, where agents cannot pre-select their communication partners. Similarly, Roos et al. [Roos *et al.*, 2002, Roos *et al.*, 2003b, Roos *et al.*, 2004] analyze a model-based diagnosis method for spatially distributed knowledge. But, their method assumes that there are no conflicts between the knowledge of the different agents, i.e., that no coordination failure occurs.

Roos et al. [Roos *et al.*, 2001] address semantically distributed systems, where the knowledge is distributed among the agents. Each agent is an expert in a certain problem domain. In this system, if each agent makes diagnosis separately the diagnosis will be incomplete since the dependencies between the agents are not diagnosed. Roos et al. suggest to maintain the dependencies between the agents. Each agent will diagnose its own domain and the related dependencies of its domain. The communication links are fixed, such that each failure is diagnosed strictly by the agents that are associated with its communication link. Biteus et al. [Biteus *et al.*, 2006] expand this approach to compute even minimal cardinality diagnosis.

Roos et al., Fröhlich at al. and Biteus at al. assume that each diagnoser agent knows the context of its sub-system and so it may make the diagnosis. However, the interactions among system entities, in multi-agent teams, are not known in advance since they depend on the specific conditions of the environment in runtime and the appropriate actions assigned by the agents [Micalizio *et al.*, 2004]. We can solve this problem by keeping all the possible interactions between the agents, however, as Roos et al. point out this may cause a large communication complexity, especially in a large system, since the number of candidate diagnoses is exponential (in the number of dependencies).

2.1.3 Modeling as Constraint Satisfaction Problem

Only a few researchers use CSP methods to practically diagnose a system. Wotawa [Wotawa, 2004] makes use of the corresponding representation of the environmental models as constraint satisfaction problems. He shows how this representation can be used directly to derive explanations and diagnoses. For this goal, he models the system using the cause-effect model, such that different solutions for the CSP are actually different explanations for the system, and the diagnoses are derived from them.

Sachenbacher and Williams [Sachenbacher and Williams, 2004] extend this model to cope with constraint optimization problems over lattices, and with semiring-CSPs. Here again, a satisfaction of constraints signifies an explanation to a fault.

In contrast to both works, we use constraints to model the ideal coordination relationships. Thus the diagnosis algorithm goal is to diagnose the violated constraints.

2.2 Distributed Diagnosis

Micalizio et al. [Micalizio *et al.*, 2004] combine a discrete-event system approach and a causal model approach to detect and diagnose failures in multi-robot system. Once a failed action is detected in the discrete-event system, the diagnoser continues to diagnose the reason for the failure by finding the causes for the failure in a database of diagnosis rules. Although their model is of a distributed multi-robot system, a centralized diagnoser makes the diagnosis. In contrast to this, we present a diagnosis computation in a distributed fashion.

Ardissono et al. [Ardissono *et al.*, 2005] divide the system into sub-systems where every agent is responsible for its own sub-system. Instead of letting the agents compute the global diagnosis by exchanging information, the agents send only necessary information to a central diagnostic service by request. A central diagnoser may be able to solve coordination failures (although they did not mention it in their paper). In contrast, some of the methods we report on here are distributed, and thus avoid the shortcomings of centralized methods.

Provan [Provan, 2002] proposes a diagnosis for distributed systems where every sub-system computes a local diagnosis. Then by using communication the sub-systems compute a global diagnosis. This approach is based on a structure-based diagnosis framework of Darwiche [Darwiche, 1998], where each sub-system is modeled as a component in a decomposition graph and the edges between the components represent the relations between the components. One drawback of this approach is that we must be able to model the distributed system in a decomposition hierarchy graph. This is impossible in multi-agent system since the interactions among system entities are not known in advance since they depend on the specific conditions of the environment in runtime and the appropriate actions assigned by the agents.

Recently Daigle at al. [Daigle *et al.*, 2006] presented a distributed diagnosis approach of coupled mobile robots. They modeled the robots with a qualitative temporal causal graph, where each robot is followed by a local diagnoser. Once the diagnoser detects a residual between the model and the actual values, it builds a set of local measurements to isolate the fault. By cooperation the diagnosers compute the minimal set of joint measurements. In this work the authors did not address minimal communication and computation in finding the joint measurements. Also, they did not mention a team of more than two agents, where the challenge of minimizing communication and computation becomes more relevant.

2.3 Large-Scale Teams

Some works address diagnosing a team of agents, but none of them consider the problem of large-scale teams. For instance, Fröhlich et al. [Fröhlich *et al.*, 1997] and Roos et al. [Roos *et al.*, 2002, Roos *et al.*, 2003b, Roos *et al.*, 2004] (described in section 2.1) assume the communication links are fixed, such that each failure is diagnosed strictly by the agents that are associated with its communication link. In contrast to this assumption; however, in teams the interactions among system entities are not necessarily known in advance since they depend on the specific conditions of the environment at runtime, and the appropriate actions assigned by the agents [Micalizio *et al.*, 2004]. It is impossible to address this by keeping all the possible interactions between the agents since it may increase communication complexity, especially in large systems, since the number of candidate diagnoses is exponential (in the number of dependencies).

A number of previous works address scalability in multi-agent systems, but do not consider diagnosis. The most related area of work deals with failure detection, rather than diagnosis. Kaminka and Bowling [Kaminka and Bowling, 2002] and later Kaminka [Kaminka and Frenkel, 2005, Kaminka, 2006] addresses large-scale teams, and their detection capabilities can complement ours by triggering the diagnosis methods we present once a failure has been detected. They show that only specific key agents in a team must be monitored to detect failures, similarly to our use of representative agents for diagnosis (in the GROUPING method in section 8.3).

Scerri et al. [Scerri *et al.*, 2005a, Scerri *et al.*, 2005b] address tasks of team coordination among the members of large teams. Specifically, they developed algorithms meeting the requirements of large teams for planning, sharing information and task allocation—but not diagnosis. They achieve the scalability by organizing all members into an associated network, which is similarly the using of grouping in social diagnosis. The associated network is performed at the initialization and remains static during the execution.

Durfee [Durfee, 2001] discusses heuristic methods for reducing the knowledge that agents use in coordination. The methods are based on hierarchies and abstractions which depend on the task environments and collective behavior of the team. As the former work to this work also addresses large-scale teams but does not consider the diagnosis problem.

2.4 Diagnosis by Heuristical Approaches

A closely-related work to ours is reported by Kaminka and Tambe [Kaminka and Tambe, 1998, Kaminka and Tambe, 2000]. This previous investigation provides guarantees on detection of disagreements, but only presents a heuristic approach to diagnosis, which indeed does not always succeed. The algorithms we present here succeed in the same examples where the previous heuristic approach fails.

Brodie et al. [Brodie *et al.*, 2001, Brodie *et al.*, 2002] tries to minimize communication costs in computer networks. They use probes which are sent through the network in order to query remote nodes. By combining the results of different probes, failing nodes can be identified and isolated. Thus Brodie et al.'s work essentially determines the liveliness status of agents, while our work focuses on fine-grain diagnosis of causes for disagreements, in terms of contradictory beliefs held by different agents.

Parker [Parker, 1998] reports on a behavior-based architecture which is very robust and is able to recover from failures by having robots take over tasks from failing teammates. This is done using continuous communications, but without an explicit diagnosis process such as those described in this dissertation.

2.5 Summary

To summarize, while in distributed systems every sub-system is controlled separately considering the **pre-defined** dependencies with the other sub-systems, in multi-agent systems, the dependencies are not necessarily known in advance since they depend on the specific conditions of the environment in runtime. In addition, the agent-based approach of diagnosing distributed systems, assumes that the diagnosing agents are external to the system, and so they know the input and output of the system—the basis for making a diagnosis. However, in multi-agent system the agents are part of the system, and so the input and output are subjective by the point of view of each diagnosing agent. Therefore, the diagnoser must take it into account while computing the diagnosis.

While most of the literature of diagnosing distributed systems and multi-agent systems does not address the problem of coordination failures, in this dissertation we address the following challenges, which have not been addressed in previous work:

1. Allowing model-based diagnosis of coordination failures.
2. Centralized and distributed diagnosis methods.
3. The exploration of a design space for coordination diagnosis algorithms.
4. Coordination failure diagnosis in teams with many agents.

Part I

Model-Based Social Diagnosis

Coordination among teammates (e.g., on a joint plan or goal) is a key factor/component to establishment and maintenance of teamwork [Cohen and Levesque, 1991, Jennings, 1995, Grosz and Kraus, 1996, Tambe, 1997]. For instance, in RoboCup the players must coordinate the attack and the defense ([Matsubara *et al.*, 1998]). In disaster rescue, where teams composed of agents assisted response vehicles, robots and people may enable more rapid crisis response ([Tambe *et al.*, 2005]). Moreover, in an attack squadron the scouts must be in coordination with the attackers (e.g. in the ModSAF domain [Kaminka and Tambe, 1998]). Unfortunately, coordination may fail due to sensing failures, different interpretations of sensor readings or intermittent communication failures, etc. In this part we address failures in coordination and propose diagnosis methods to detect and isolate such failures.

The idea of diagnosis is to go from fault *detection* (where an alarm is raised when a fault occurs), to fault *identification*, where the causes for the fault are discovered. In diagnosis of coordination failures we go a step beyond the detection of the fault [Kaminka and Tambe, 2000] to isolate the faulty agents and specifically identify the faulty sensors that caused the failure. Once the failures are known, then the agents can be negotiated and argued about, to resolve the coordination failures e.g., [Kraus *et al.*, 1998]. Diagnosis is a necessary process since detecting a failure does not always lead to a solution since it is possible that the failed agents do not know that they caused the failure. Also, in case that the failure is in a sensor, we may want to fix it before continuing the original plan in order to prevent the same failure in the future.

To illustrate, assume a group of robotic space explorers, whose goal is to slowly creep on a newly-discovered alien. To capture the alien, they must approach it from all sides in alternating steps: A bit from the left, then from the right, then again from the left, etc (Figure 2.1). To do this in a coordinated manner, the robots divide into sub-teams of three that spread around the alien, each with a sub-team leader (C_i) and two followers (D_i), that move in formation using cameras to maintain distances and angles. A mission commander (B) directs the sub-team leaders, alternating commands for them to go and stop, as needed.

The robots must coordinate all throughout their mission. The sub-team leaders are coordinated with each other via the mission commanders' commands; and each sub-team's leader is coordinated with its followers using vision. Once a coordination failure is detected, the mission must be suspended, in order to diagnose the failing

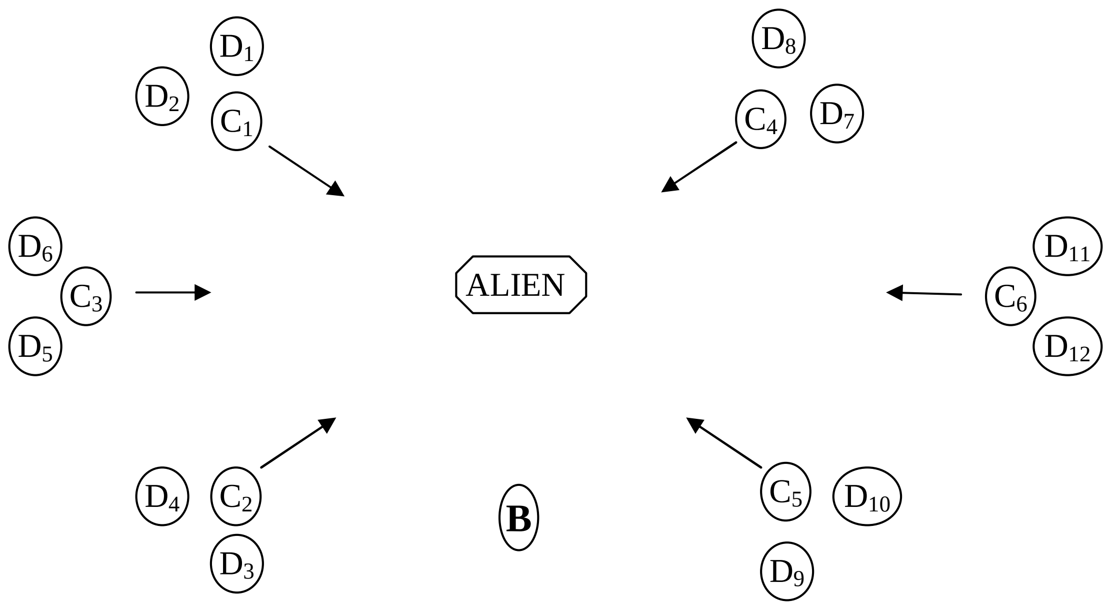


Figure 2.1: Example: Robotic space explorers tackling an alien.

robots and then reestablish collaboration.

A coordination failure could happen due to intermittent communication failures (between the mission commander and sub-team leaders) or due to a vision failure (a sub-team leader and its followers). Multiple failures could happen at the same time, for instance, assume the commander sent a message to the leader of the left sub-team to go and concurrently to the leader of the right sub-team to stop. Both of them did not receive the message due to communication failure. Concurrently the followers of the leader of the left sub-team stopped since, due to a failure in their vision, they thought that their leader stopped. The goal of the diagnosis process is to find the space of the possibilities over which the agents failed.

This part formalizes the coordination diagnosis process as a model-based diagnosis method. It is organized as follows: Chapter 3 is an introduction to model-based diagnosis. In this chapter we introduce the notions of consistency-based diagnosis as well as abductive diagnosis. Chapter 4 formalizes the coordination diagnosis in terms of consistency-based and abductive diagnoses. Chapter 5 focuses on abductive diagnosis by presenting distributed methods to compute the diagnosis based on distributed constraint satisfaction problem algorithms. Chapter 6 presents the experiments for the coordination diagnosis algorithms and provides a discussion to summarize the advantages and disadvantages of the different methods.

Chapter 3

Introduction to Model-Based Diagnosis

Model-based diagnosis (MBD) is an area of Artificial Intelligence which tackles the problem of identifying faulty components in a system by observation. As the basis of this approach, we distinguish between the actual system and the model of the system. The model describes either how the system is supposed to behave (what is the correct behavior), or the relationship between faults and symptoms (what is the faulty behavior), or possibly both. The diagnoser observes the actual behavior of the system and predicts its behavior by the model. Discrepancies between the observation and the prediction—symptoms—are used as the input for diagnosis algorithms which produce a set of possible faults that can have explain such symptoms.

Many of the early works [de Kleer and Sussman, 1980, Davis, 1984] start from the assumption that the model of the system contain information about the correct behavior solely. The task of diagnosis then is to identify which components of the system are faulty, in the sense that their normal behavior is inconsistent with the observations. This approach, known as consistency-based diagnosis, was formalized by Reiter [Reiter, 1987].

Other researchers [Peng and Reggia, 1990] adopt a different approach which involve describing objective knowledge about the causal relationships between faults and symptoms [Bylander, 1990]. They thus exploited the causal model of a system, containing explicit information about which faults can occur. This approach is known as abductive diagnosis [Cox and Pietrzykowski, 1987]. In the following sections we will outline the these approaches, which are the basis of the work we present in this thesis; for an in depth account of these and other works, see

[Hamscher *et al.*, 1992].

3.1 Consistency-Based Diagnosis

In consistency-based approach we model the correct behavior of the systems, then find the discrepancies given a context. Let us formalize the notion of consistency-based diagnosis using Reiter's formalism [Reiter, 1987]. In order to define the diagnostic problem we introduce some notions. SD is a description of the system behavior in firstorder logic. It can be divided in two parts: the components description, which states the behavior of the individual components (usually expressing some relations between their input and output variables), and the system structure, which states how the components are connected together (usually equating an output of a component with an input of another one). $COMPS$ is a set of components; in particular each component is associated with a constant c (the component name), where $ab(c)$ is an unary predicate which is true when component c is faulty. Thus the description of a component's behavior is always conditioned by the premise that the component must not be behaving abnormally. If a component c is behaving abnormally (that is, if $ab(c)$ holds) then we can say nothing about how it behaves. OBS is the set of observations that have been made on the system, represented by first-order sentences.

A diagnosis is necessary, when the observations are not consistent with the assumption that the whole system is behaving correctly.

Definition 3.1.1. *Diagnosis Problem.* Given $\{SD, COMPS, OBS\}$ the *diagnosis problem (DP)* arises when

$$SD \cup \{\neg AB(COMPS_i) | COMPS_i \in COMPS\} \cup OBS \vdash \bot$$

At this point it comes natural to define a diagnosis as follows:

Definition 3.1.2. A *consistency-based diagnosis* is a minimal set $\Delta \subseteq COMPS$ such that:

$$SD \bigcup \{AB(COMPS_i) | COMPS_i \in \Delta\} \bigcup \{\neg AB(COMPS_i) | COMPS_i \in COMPS - \Delta\} \bigcup OBS \nvdash \bot$$

Diagnosis Δ is a set of components, which when considered abnormal, cause the system description with the other components and the observation to be consistent. If asserting abnormality for some components restores consistency, then we can conclude that those components were responsible for the original inconsistency. We are interested in *minimal diagnosis*, meaning, no proper subset of it is a diagnosis.

3.2 Abductive Diagnosis

Diagnosis can be characterized as a process of generating explanations for a set of observations in a given context. In consistency-based a diagnosis explains an observation if it does not contradict it. On the other hand, in abductive diagnosis the explanation notion is stronger. A diagnosis explains an observation if it directly supports it [Console and Torasso, 1991].

Reasoning by abduction is the process of inferring the causes from their effects. The abduction rule is used in common sense reasoning, nevertheless it is not logically correct, since it is true that if B holds then it may have been caused by A, but it is also true that B may hold independently from A. Abductive reasoning is typical in medical diagnosis, where one tries to infer the causes of the symptoms from the symptoms themselves, and from "rules" stating the cause effect relationships between diseases and their manifestations.

In model-based diagnosis such kind of knowledge can be represented in a causal model of the system to diagnose. In first-order logic a causal model is described by implicational rules. Some predicates represent faults that can occur in the system and are thus declared to be abducible. They are the causes by which we want to explain the symptoms [Luca console, 1990, Console *et al.*, 1991].

Based on the formalism presented in consistency based approach, in abductive diagnosis normal behaviors as well as the faulty behaviors must be represented by the description of the system SD. The diagnosis must explain the specific faulty behaviors. An abductive diagnosis is a minimal set of assignments of the behavior modes to the components (normal and faulty), that is both consistent with the system description and the observation.

Formally, in addition to the based condition of consistency based diagnosis ($SD \bigcup \Delta \bigcup OBS \nvDash \bot$), we must add one more condition that the the diagnosis must entail the observation [Console *et al.*, 1989]:

Definition 3.2.1. An *abductive diagnosis* is a minimal set Δ such that:

$$SD \bigcup \Delta \bigcup OBS \nvDash \bot$$

$$SD \bigcup \Delta \models OBS$$

This diagnosis is stronger than the consistency-based diagnosis, and supplies more information on the faulty modes of the system. The price is that the fault modes must be defined apriori. This requirement is not satisfied in many systems and so abductive diagnosis in such systems cannot be complete (i.e. will not generate all explanations).

Chapter 4

Social Diagnosis Formalism

In the previous chapter we presented two approaches for model-based diagnosis the consistency-based approach and abductive approach. Implementing these approaches in social diagnosis is not an easy task, since while in a single system the context and the observation are known to the diagnoser, in multi-agent system this assumption is not always correct. This is because many failures take place at the boundaries between the agent and their environment, including other agents. In addition, many diagnosis methods for distributed systems [Pencolé *et al.*, 2002, Lamperti and Zanella, 2003] and multi-agent systems [Fröhlich *et al.*, 1997, Roos and Witteveen, 2005], distinguish between the starting points of the inputs (context) and the ending points of the outputs (observations). Then the diagnosis problem is to find a set of abnormal components that explain the observations given the context. However, in social diagnosis the observations depend on the the expected coordination between the agents and not on a given context.

Surprisingly, it is still often possible to detect and diagnose coordination failures, given the actions of agents (observations), and the coordination constraints that should ideally hold between them. In this sense, the diagnosis problem is to find a minimal set of abnormal agents which explain the observations given the coordination constraints between them.

We adopt a model-based diagnosis approach to diagnose the agents and the coordination failures. In model-based diagnosis of a single agent, the diagnoser uses a model of the agent to generate expectations which are compared to the observations, in order to form diagnoses [Davis and Hamscher, 1988, Reiter, 1987, de Kleer and Williams, 1987]. In model-based multi-agents diagnosis, the diagnoser models the coordination between the agents. The goal of the diagnosis is to diagnose the failures in the coordination by detecting deviation of the observation from the model's predictions.

The chapter is organized as follows: Section 4.1 formalizes the model of a single agent and the model of a team coordination. Section 4.2 presents the coordination as constraints between the agents. Based on this formalism we define the coordination diagnosis problem. Section 4.3 defines the diagnosis by consistency based approach and Section 4.4 defines it by abductive based approach.

4.1 Coordinated multi-agent Systems

In order to base our model on MBD, we will present a model of a single agent and a model of the coordination between the agents.

4.1.1 The Agent Model

An agent is an entity that perceives its environment through sensors and takes actions upon its environment using actuators. Obviously, there are many different possible models that can be used to describe agents. Our focus is on the coordination of multiple agents through their actuators and their sensors, and thus we will use a simplified model, of completely reactive agents, composed only of sensor and actuator components. The connections between the sensors and actuators are described logically.

Definition 4.1.1. An *agent* is a pair $\langle CMP, CON\rangle$ of components CMP, and connections CON. CMP is a pair $\langle SEN, ACT\rangle$ where SEN is a set of boolean variables representing the sensors and ACT is a set of boolean variables representing the actions. CON is a set of logical consequence statements, where the literals of SEN are on the left side of consequences, and the literals of ACT are on the right side.

At any time, the agent may sense through a number of sensors, but may only select one action. Thus multiple literals in SEN may be true, but at any time exactly one literal of ACT must be true. To enforce this, we apply a *completeness* formula (i.e. $ACT_1 \vee \ldots \vee ACT_{|ACT|}$) and a set of *mutual-exclusion* formulas $\forall i, j \neg(ACT_i \wedge ACT_j)$.

Example 4.1.1. *Following the example presented in the introduction to this part, suppose we model a sub-team leader robot who approaches to the alien. The robot has* *two sensor components, one is a radio sensor with two message values* $\{go, found\}$ *and the other is a camera sensor which indicates if the wounded is found. The actions of the robot* $\{GO, WAIT\}$ *are selected based on the sensor readings: Once the robot receives a go message from the commander it selects the action* GO*. It will switch to the action* $WAIT$ *upon capture the alien (via its camera), or upon receiving a message that it was captured (by someone else).*

We represent this agent as follows:

$$\begin{aligned} SEN = &\quad \{SEN_{radio\_go}, SEN_{radio\_found}, SEN_{camera\_found}\} \\ ACT = &\quad \{GO, WAIT\} \\ CON = &\quad \{SEN_{radio\_go} \wedge \neg SEN_{camera\_found} \Rightarrow GO, \\ &\quad SEN_{radio\_found} \vee SEN_{camera\_found} \Rightarrow WAIT\} \end{aligned}$$

In addition we should verify that only one action is selected by the agent, using the following completeness and mutual-exclusion axioms:

$$WAIT \vee GO$$
$$\neg(WAIT \wedge GO)$$

4.1.2 A Model of Coordination

The multi-agent systems of interest to us are composed of several agents, which (by design) are to satisfy certain coordination constraints. We call this type of system a *team*, to distinguish it from general multi-agent systems in which it is possible that no coordination constraints exist.

Definition 4.1.2. A *team* T is a set of agents. $T = \{A_1 ... A_n\}$ where A_i is an agent. Given a team T, AS represents the set of the action literals of the agents. Formally, let ACT_i be the set of actions of agent A_i then $AS = \bigcup_{i=1}^{n} ACT_i$, where AS_{ij} represents the j'th boolean action variable of agent A_i. As a shorthand, we use AS_i to denote the boolean action literal of agent A_i whose value is true. We call AS_i the *active selection* of agent A_i.

The actions of agents in a team are coordinated. We utilize two coordination primitives—*concurrence* and *mutual exclusion*—to define the coordination constraints. Concurrence states that two specific actions must be taken jointly, at the same time. Mutual exclusion states the opposite, i.e., that two specific actions may not be taken at the same time.

Definition 4.1.3. A *concurrence coordination (CCRN)* constraint between two actions of different agents mandates that the two actions must be true concurrently. Logically, we represent this constraint in a DNF (disjunctive normal form). For two actions AS_{ix} and AS_{jy} (action x of agent A_i and action y of agent A_j) as follows:

$$CCRN(AS_{ix}, AS_{jy}) \Rightarrow (AS_{ix} \wedge AS_{jy}) \vee (\neg AS_{ix} \wedge \neg AS_{jy})$$

Definition 4.1.4. A *mutual exclusion coordination MUEX* constraint between two actions of different agents mandates that they cannot be true concurrently. Logically, for two actions AS_{ix} and AS_{jy} (action x of agent A_i and action y of agent A_j) as follows

$$\begin{aligned} MUEX(AS_{ix}, AS_{jy}) \;\; \Rightarrow \;\; & (AS_{ix} \wedge \neg AS_{jy}) \vee \\ & (\neg AS_{ix} \wedge AS_{jy}) \vee \\ & (\neg AS_{ix} \wedge \neg AS_{jy}) \end{aligned}$$

Once we defined the coordination types, we can model the coordination between the agents formally with a set of coordination constraints, defining a graph:

Definition 4.1.5. A *coordination graph* for a team T is an undirected graph $CG = \{V, E\}$, where the vertices set V represents the boolean variables of the actions of the agents, and the set of edges E is the set of coordination constraints between the actions. We use CG_m to refer to the m'th constraint within E. $CG(AS_{ix}, AS_{jy})$ denotes the constraint relating AS_{ix} and AS_{jy}. CG_m is considered true if the constraint holds and false otherwise.

Example 4.1.2. *Figure 4.1 presents a coordination graph. The concurrence constraints are represented by solid lines, and the mutual exclusion constraints are represented by dashed lines. Following the example presented in the introduction to this part, assume a team of four agents $\{B, C_1, D_1, D_2\}$. B is the commander robot, C_1 is the leader of the sub-team $\{C_1, D_1, D_2\}$ (as described in Example 4.1.1) and D_1 and D_2 are the followers.*

The commander robot has two command actions $\{GO, STOP\}$ to direct the sub-team leader. There are concurrence coordination constraints between the commands of the commander and the actions taken by the sub-team leader. The followers have *vision sensors which enable them to follow the leader by two actions {FOLLOW, FREE}. Once the sub-team leader goes they should follow it. We take care of the following coordination by concurrence coordination constraint between the commands of the commander and the actions of the sub-team leader. In addition a mutual exclusion coordination constraint is defined to prevent the followers from continuing to freely move after the sub-team leader stops. In this example, we could easily change this mutual exclusion coordination with a concurrence coordination by replacing the* $\{FREE\}$ *action to* $\{STOP\}$ *action. Then we could define the coordination between the* $\{STOP\}$ *action of the whole team as a concurrence coordination. However, in general, where the followers have more than only two actions, it may be not possible to reduce mutual-exclusion to a series of concurrence between pairs of other actions.*

$$
\begin{aligned}
V = \quad & \{AS_{B_{STOP}}, AS_{B_{GO}}, AS_{C1_{STOP}}, AS_{C1_{GO}}, \\
& AS_{D1_{FOLLOW}}, AS_{D1_{FREE}}, AS_{D2_{FOLLOW}}, AS_{D2_{FREE}}\} \\
E = \quad & \{CCRN(AS_{B_{GO}}, AS_{C1_{GO}}), \\
& CCRN(AS_{B_{STOP}}, AS_{C1_{STOP}}), \\
& CCRN(AS_{C1_{GO}}, AS_{D1_{FOLLOW}}), \\
& CCRN(AS_{C1_{GO}}, AS_{D2_{FOLLOW}}), \\
& MUEX(AS_{C1_{STOP}}, AS_{D1_{FREE}}), \\
& MUEX(AS_{C1_{STOP}}, AS_{D2_{FREE}}),
\end{aligned}
$$

Given a coordination graph CG and a team T, we can define a multi-agent system description as a set of implications from the normality of the agents to the satisfaction of the coordination constraints. This is the final piece in formalizing a normally-functioning multi-agent system.

Definition 4.1.6. A *multi agent system description (MASD)* is a set of implications from the normality of agents in a team T to CG. The meaning of the predicate $AB(\cdot)$ is that the corresponding agent is considered abnormal (failing).

$$
\begin{aligned}
MASD = \quad & \{\neg AB(A_i) \wedge \neg AB(A_j) \Rightarrow CG(AS_{ix}, AS_{jy}) \\
& |CG(AS_{ix}, AS_{jy}) \in CG \wedge A_i, A_j \in T\}
\end{aligned}
$$

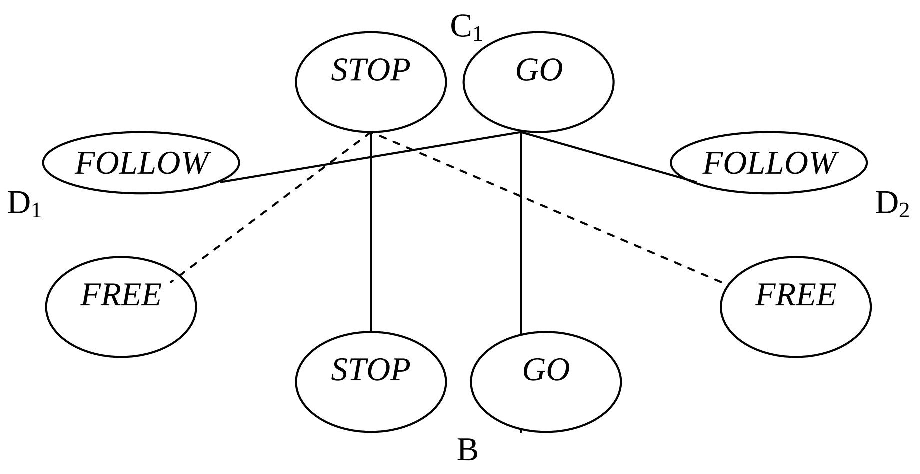


Figure 4.1: The coordination graph for team $\{B, C_1, D_1, D_2\}$.

This definition enforces the dependency between the perfection, or in terms of model-based diagnosis, the normality of the agents and the correctness of the coordination among the team. For instance, if the commander and the leader of the sub-team are undamaged, then the coordination between them is expected to be correct. In other words, the concurrence constraints between their actions are correct.

4.2 Diagnosis of Coordination Faults

A fault in the coordination of a multi-agent system may be the result of a fault in one of the sensors or other agent components [1] Given a $MASD$ and a set of normality assumptions, it is possible to infer that a fault exists (and to generate hypotheses as to its identity), by checking whether the observed actions of the agents satisfy the $MASD$.

Let us formalize the coordination diagnosis in terms of model based diagnosis:

Definition 4.2.1. *Coordination Diagnosis Problem.* Given $\{T, MASD, AS\}$ where T is a team of agents $\{A_1...A_n\}$, $MASD$ is a multi agent system description defined over T (Definition 4.1.6), and AS is the set of the actions of the agents (Definition 4.1.2), then the *coordination diagnosis problem (CDP)* arises when

[1]It may also be the result of a fault in the environment, e.g., when a message is lost in transit. This is treated as a fault in the receiver.

$$MASD \cup \{\neg AB(A_i) | A_i \in T\} \cup AS \vdash \perp$$

We use the following example to illustrate.

Example 4.2.1. *Suppose we are given the following $MASD$, T, and AS (only the true literals in AS are shown):*

$$\begin{aligned}
T = & \quad \{A_1, A_2, A_3, A_4, A_5, A_6\} \\
MASD = & \quad \{\neg AB(A_1) \wedge \neg AB(A_4) \Rightarrow MUEX(AS_{11}, AS_{41}), \\
& \quad \neg AB(A_1) \wedge \neg AB(A_2) \Rightarrow CCRN(AS_{12}, AS_{21}), \\
& \quad \neg AB(A_1) \wedge \neg AB(A_6) \Rightarrow CCRN(AS_{12}, AS_{61}), \\
& \quad \neg AB(A_2) \wedge \neg AB(A_3) \Rightarrow CCRN(AS_{22}, AS_{31}), \\
& \quad \neg AB(A_2) \wedge \neg AB(A_5) \Rightarrow CCRN(AS_{22}, AS_{51}), \\
& \quad \neg AB(A_2) \wedge \neg AB(A_6) \Rightarrow CCRN(AS_{21}, AS_{61}), \\
& \quad \neg AB(A_3) \wedge \neg AB(A_4) \Rightarrow MUEX(AS_{32}, AS_{42}), \\
& \quad \neg AB(A_3) \wedge \neg AB(A_5) \Rightarrow CCRN(AS_{31}, AS_{51})\} \\
AS = & \quad \{AS_{11}, AS_{21}, AS_{31}, AS_{41}, AS_{51}, AS_{61}\}
\end{aligned}$$

Figure 4.2 shows the coordination graph for this CDP (gray vertexes represent the active selection of the agents). Assuming all the agents are not abnormal, the actions of the agents are not consistent with the coordination graph. For instance, the actions $AS_{11} = true$ and $AS_{41} = true$ causes an inconsistency in CG_1, as it produces a false value of $MUEX(AS_{11}, AS_{41})$, though, $MUEX(AS_{11}, AS_{41})$ should be true, given the normality assumptions $\neg AB(A_1), \neg AB(A_4)$. On the other hand, if the actions AS_{12}, AS_{21}, AS_{32}, AS_{41}, AS_{52}, AS_{61} were true (implying that the other actions were false), they would have been consistent with the coordination graph.

Given a CDP, the goal of the coordination diagnosis process is to determine a minimal set of abnormal agents whose selection and subsequent setting of the $AB(\cdot)$ clause would eliminate the inconsistency (consistency-based diagnosis, Section 4.3), or explain it (abductive diagnosis, Section 4.4). A coordination diagnosis (a set of abnormal agents) is minimal, iff no proper subset of it is a coordination diagnosis.

Once the set of such abnormal agents is found, the diagnoser infers the abnormal components (in our case, sensors) within the abnormal agents. This is done using straightforward back-chaining through the set CON (definition 4.1.1) of logical consequence statements connecting sensors to actions.

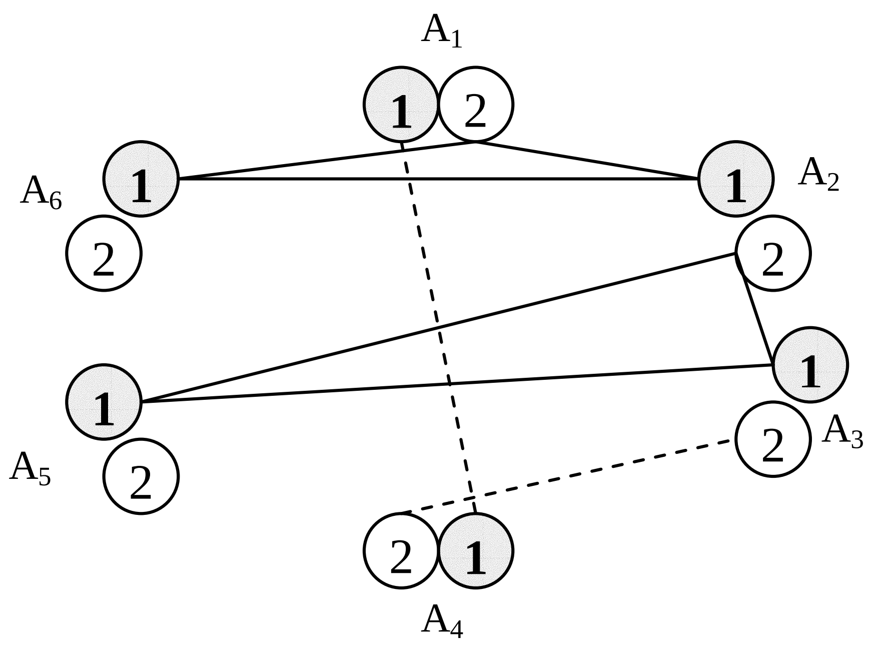


Figure 4.2: The coordination graph and active selection (gray vertexes) of the team $T = \{A_1, A_2, A_3, A_4, A_5, A_6\}$.

4.3 Consistency-Based Coordination Diagnosis

We begin by defining consistency-based coordination diagnosis.

Definition 4.3.1. A *consistency-based global coordination diagnosis (CGCD)* is a minimal set $\Delta \subseteq T$ such that:

$$MASD \bigcup \{AB(A_i) | A_i \in \Delta\} \bigcup \{\neg AB(A_i) | A_i \in T - \Delta\} \bigcup AS \nvdash \bot$$

The first step in this process to determine which agents are in conflict:

Definition 4.3.2. Two agents a and b are called *conflict pair* $\langle a, b \rangle$, if there exist a constraint CG_i that relates a and b and whose value is false.

$$\forall a, b \in T, \exists i, j, k \ s.t. \neg CG_i(AS_{aj}, AS_{bk}) \Rightarrow \langle a, b \rangle$$

Definition 4.3.3. A *local conflict set* is a set of all conflict pairs in the system, and is denoted by LC.

Example 4.3.1. *LC in the graph of example 4.2.1 is:* $LC = \{\langle A_1, A_4\rangle, \langle A_1, A_2\rangle, \langle A_1, A_6\rangle, \langle A_2, A_3\rangle, \langle A_2, A_5\rangle\}$*, since the coordination constraints between the actions of these agent pairs were violated.*

The local conflict set forms the basis for the $CGCD$, because for each conflict pair, at least one of the agents is abnormal. However, $CGCD$ is not a simple combination of all agents in the LC pairs, as arbitrary selection of agents may lead to diagnosis sets that are themselves inconsistent. For instance, treating each pair in the computed LC in Example 4.3.1 by itself, produces the following subset of possible diagnoses:

$$\langle A_1, A_2\rangle \Rightarrow \{AB(A_1), \neg AB(A_2)\}$$
$$\langle A_1, A_2\rangle \Rightarrow \{\neg AB(A_1), AB(A_2)\}$$
$$\langle A_1, A_4\rangle \Rightarrow \{AB(A_1), \neg AB(A_4)\}$$
$$\langle A_1, A_4\rangle \Rightarrow \{\neg AB(A_1), AB(A_4)\}$$

It is easy to see that combining these diagnoses may produce inconsistency. For instance, combining the first and last implications would produce the set $\{AB(A_1), \neg AB(A_2), \neg AB(A_1), AB(A_4)\}$, which contains both $AB(A_2)$ and $\neg AB(A_2)$.

Therefore, we cannot diagnose every conflict pair by itself and then combine the results. Rather, we should compute the diagnoses sets Δ considering the dependencies between the conflict pairs. To do this, we should look for the abnormal agent(s) in every conflict pair.

We achieve this goal by generating a hitting-set of agents, selecting at least one agent as abnormal from every conflict pair, such that the resulting agents cover between them all conflict pairs. We want to maintain a minimal number of such agents. This is somewhat similar to Reiter's HS-Tree [Reiter, 1987], or de Kleer and Williams' technique [de Kleer and Williams, 1987]. It is also related to minimal model techniques used in non-monotonic reasoning [Olivetti, 1992, Niemelä, 1996].

We achieve this goal by transforming the conflict set into a graph, and finding the vertex cover for this graph. Let us define a conflict graph:

Definition 4.3.4. A *conflict graph* $CONG = \{V', E'\}$ is a graph generated by the conflict pairs where E' is a set of the conflict pairs and V' is a set of the agents involved in the conflict pairs.

In order to compute the diagnosis we run an algorithm to find a minimal vertex cover—a set of vertices that involve all edges. A vertex cover set is guaranteed to be a diagnosis since all the edges, namely the conflict pairs, are covered by this set, namely by a set of abnormal agents. We are looking for all the possible minimal vertex cover sets, since the diagnosis contains all the possibilities of abnormal agents. Minimal vertex covers guarantee minimal diagnosis, since a vertex cover is minimal only if no proper subset of it is a vertex cover.

Algorithm `CONSISTENCY_BASED_COORDINATION_DIAGNOSIS` (Algorithm 1) summarizes the process of computing the coordination diagnosis in a consistency based approach. In line 1 we initialize Δ — a set of all diagnosis sets. Then in lines 2–4 we go over the constraints (edges) in the coordination graph (definition 4.1.5) and check for each one of them if it produces a conflict (definition 4.3.2). In case of a conflict, the conflict pair is added to a local conflict set (definition 4.3.3). Once all the conflict pairs where found, we build in line 5 a conflict graph (definition 4.3.4) and then compute the vertex cover sets in all sizes (lines 6–7). A vertex cover set is added to Δ as a new diagnosis if it is not a superset of an existing diagnosis. If the vertex cover set is a subset of an existing diagnosis then it replaces the diagnosis since it is minimal (lines 8–14).

Algorithm 1 CONSISTENCY_BASED_COORDINATION_DIAGNOSIS
(**input**: coordination graph CG
output: diagnoses set Δ)

```
1:  $\Delta \leftarrow \emptyset$
2:  for all $CG_i \in CG$ do
3:    if $\exists j,k\ s.t. \neg CG_i(AS_{aj}, AS_{bk}) \Rightarrow \langle a,b \rangle$ then
4:      $LC \leftarrow LC \cup \{\langle a,b \rangle\}$
5:  build $CONG$ from $LC$
6:  for all $m$ where $m = \{1...|V'| \in CONG\}$ do
7:    for all vertex cover $vc$ in size $m$ do
8:      if $vc \subset \Delta_x \in \Delta$ then
9:        remove $\Delta_x$ from $\Delta$
10:       $\Delta \leftarrow \Delta \cup \{vc\}$
11:     if $vc \supseteq \Delta_x \in \Delta$ then
12:       continue
13:     else
14:       $\Delta \leftarrow \Delta \cup \{vc\}$
15: return $\Delta$
```

Example 4.3.2. *Figure 4.3 presents the graph of the conflict pairs that were computed in example 4.3.1. The vertex cover set of size one is empty, for size two it is $VC_1 = \{A_1, A_2\}$, and there are two sets of size three: $VC_2 = \{A_1, A_3, A_5\}$ and $VC_3 = \{A_2, A_4, A_6\}$ (there are more vertex cover sets which are superset of VC_1), it is unnecessary to continue to check the vertex cover in size four and more since every such vertex cover will be a superset of the formers. By building the vertex cover sets we obtain the global coordination diagnosis, $\Delta_1 = \{A_1, A_2\}$, $\Delta_2 = \{A_1, A_3, A_5\}$, $\Delta_3 = \{A_2, A_4, A_6\}\}$.*

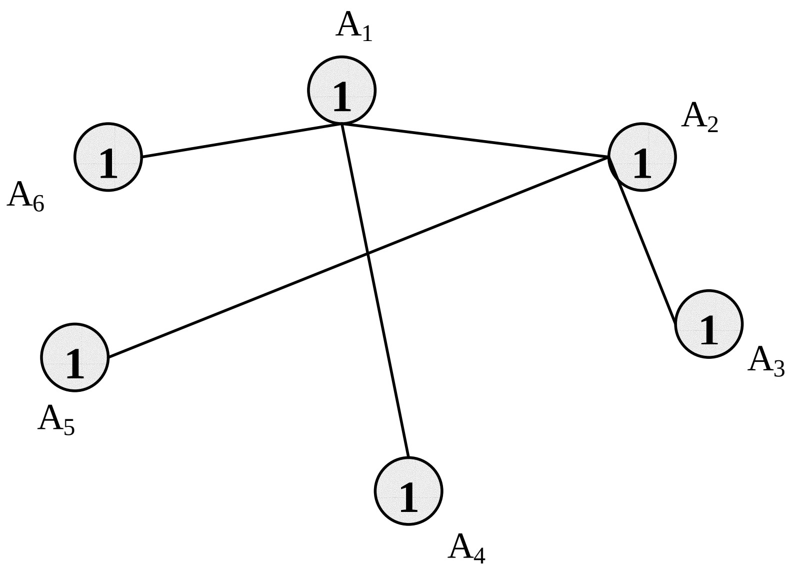


Figure 4.3: A graph of the conflict pairs in example 4.3.1.

Let us prove the correctness of computing the diagnosis by vertex cover.

Theorem 1: the set of vertex cover over conflict graph $CONG$ (definition 4.3.4) is a complete set of consistency-based global coordination diagnoses $CGCD$ (definition 4.3.1).

Proof: By contradiction. Assume for contradiction that a consistency-based global coordination diagnosis Δ_x is not a vertex cover over $CONG$. According to definition 4.3.1, Δ_x contains only abnormal agents therefore $\forall A_i \in \Delta_x \Rightarrow A_i \in CONG$. According to definition 4.3.1 the abnormal agents in Δ_x explain all the conflicts, i.e., they are involved in all the edges in $CONG$ and so they must be vertex cover, contrary to our assumption. $\square$

Determining a minimal vertex cover is known to be NP-Complete. The problem of determining the set of all minimal vertex covers is NP-Hard [Skiena, 1990]. A simple $O(2^{|V'|})$ exact algorithm for its solution is to find all the possible vertex covers in size one, then continue to find the possible vertex cover in size two, under the condition that it is not a superset of a previous vertex cover, and so on up to the max size of the graph. The complexity of computing the $CGCD$ is thus the same as in single-agent diagnosis methods, e.g., [de Kleer and Williams, 1987]. We now prove that consistency based global coordination diagnosis is NP-hard.

Theorem 2: $CGCD$ (consistency based global coordination diagnosis) is NP-Hard.

We can prove it by a reduction from "Independent set" which is a known NP-hard problem. First we re-cast the definition of $CGCD$ (4.3.1) as a decision problem[2].

Definition 4.3.5. $CGCD$: Given a set of variables $\{X_1, ..., X_n\}$, and a set of domain $\{D_1, ..., D_m\}$ for each variable with initial value $\{D_{w_1}, ..., D_{w_n}\}$ corresponding to the variables (parallel to AS in definition 4.1.2), and given a set of constraints between the variables in the forms:

1. $X_i = D_{y_i} \Rightarrow X_j = D_{z_j}$ (parallel to $CCRN$ in definition 4.1.3)
2. $X_i = D_{y_i} \Rightarrow X_j \neq D_{z_j}$ (parallel to $MUEX$ in definition 4.1.4)

Given $k_1 > 0$. Do there exist at most k_1 variables such that ignoring these k_1 variables and their constraints will satisfy the constraints that are associates with the rest of the variables $(n - k_1)$?

As in the original definition of $CGCD$ (definition 4.3.1) here we look for a diagnosis set of k_1 variables (that represent the abnormal agents) we can ignore so the rest of the normal agents are consistent with the actions (AS) and the multi-agent system description ($MASD$).

Definition 4.3.6. Independent Set Problem **ISP**: Given a graph $G = (V, E)$ and $k_2 > 0$. Do there exist a set of at least k_2 vertices such that no pair of vertices defines an edge of E?

We will show a reduction from ISP to $CGCD$, $ISP \prec CGCD$:

Construction: Given a graph $G = (V, E)$ in ISP, we build $|V|$ variables $(n = |V|)$ $\{X_1, ..., X_n\}$ with domain $\{0, 1\}$ in $CGCD$, we initialize all the variables with the value 1. Also, for every edge between V_i and V_j in ISP we build a constraint $X_i = 1 \Rightarrow X_j \neq 1$ in $CGCD$. $k_1 \leq n - k_2$.

[2]Joint credit for this proof to Efrat Manistersky.

Example 4.3.3. *Given a graph $G = (V, E)$ and $k_2 = 3$.:*

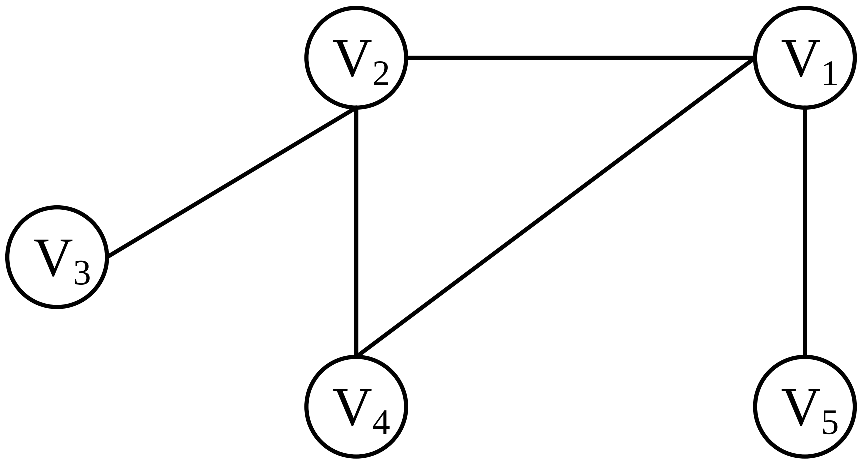


Figure 4.4: Construction.

The construction is:

$$\begin{aligned}
&\{X_1 = 1, X_2 = 1, X_3 = 1, X_4, = 1, X_5 = 1\}\\
&X_1 = 1 \Rightarrow X_2 \neq 1\\
&X_1 = 1 \Rightarrow X_4 \neq 1\\
&X_1 = 1 \Rightarrow X_5 \neq 1\\
&X_2 = 1 \Rightarrow X_1 \neq 1\\
&X_2 = 1 \Rightarrow X_3 \neq 1\\
&X_2 = 1 \Rightarrow X_4 \neq 1\\
&X_3 = 1 \Rightarrow X_2 \neq 1\\
&X_4 = 1 \Rightarrow X_1 \neq 1\\
&X_4 = 1 \Rightarrow X_2 \neq 1\\
&X_5 = 1 \Rightarrow X_1 \neq 1\\
&p = 2
\end{aligned}$$

We should prove that there is an independent set $\geq k_2$ in $G \Leftrightarrow$ there is a set of variables $\leq k_1$ in $\{X_1, ..., X_n\}$, such that ignoring their values and the constraints associated with them will satisfy the rest of the constraints.

Proof:

1. $\Rightarrow$

   Assume an independent set of size $\geq k_2$ in G. This implies that there is a set of $\geq k_2$ variables that there are no constraints between them (and so they satisfy the constraints). Consequently, the number of variables that must be ignored in order to satisfy the constraints that are not associated with them, is $\leq n - k_2$.

2. $\Leftarrow$

   Assume a set of variables $\leq k_1$ in $\{X_1, ..., X_n\}$, such that ignoring them and the constraints associated with them, will satisfy the rest of the constraints. This implies that there are at least $n - k_1$ variables with value 1, which do not violate the constraints. By construction, there are at least $n - k_1$ vertexes with no edge connect them. Consequently, there is an independent set of size $\geq n - k_1$.

$\square$

A disadvantage of the consistency-based approach is that it may produce diagnoses that are unsound, in the sense that while they eliminate the inconsistency, they do not explain it. Intuitively, such diagnoses correspond to eliminating the abnormal agents from consideration, rather than suggesting that they change their actions. For such diagnoses, there may be no actions that the abnormal agents could take that would be consistent with the $MASD$.

For instance, in Example 4.3.2 the diagnosis set $\{A_1, A_2\}$ represents a minimal set of abnormal agents, but changing their actions ($A_{11} = false$, $A_{12} = true$, $A_{21} = false, A_{22} = true$) will leave the system inconsistent, with $CCRN(AS_{12}, AS_{21}) = false$. On the other hand, changing the actions of the agents in the other diagnoses ($\{A_1, A_3, A_5\}$, $\{A_2, A_4, A_6\}$) will eliminate the inconsistency.

4.4 Abductive Coordination Diagnosis

The implication is that stronger conditions on the solution sets may be needed. Such conditions correspond to abductive diagnosis, in which changing the actions of the abnormal agents entails the coordination graph:

Definition 4.4.1. An *abductive global coordination diagnosis (AGCD)* is a minimal set $\Delta \subseteq T$ such that:

$$MASD \bigcup \{AB(A_i) | A_i \in \Delta\} \bigcup \{\neg AB(A_i) | A_i \in T - \Delta\} \bigcup AS \nvdash \bot$$

and,

$$\{AB(A_i) | A_i \in \Delta\} \bigcup \{\neg AB(A_i) | A_i \in T - \Delta\} \bigcup AS \Rightarrow CG$$

where, we make the active selection of agent A_i (definition 4.1.2), AS_i, false, and force A_i to choose a different action,

$$AB(A_i) \Rightarrow \neg AS_i \wedge (AS_{i1} \vee \ldots \vee AS_{i|ACT|})$$

The first condition in definition 4.4.1 is exactly as in definition 4.3.1 (i.e., $CGCD$) to satisfy the consistency requirement. The second condition requires that for any abnormal agents found, it will be possible to change their active selection, in order to entail the coordination graph and thus satisfy the coordination constraints. Note that the entailment here is of the coordination graph, not the full $MASD$.

The unsound diagnosis set $\{A_1, A_2\}$, given by the consistency-based approach, will not pass this second condition, since the alternative actions of agent A_1 and of agent A_2 do not entail the coordination graph as shown in Figure 4.5 (the new violated constraints are marked with X).

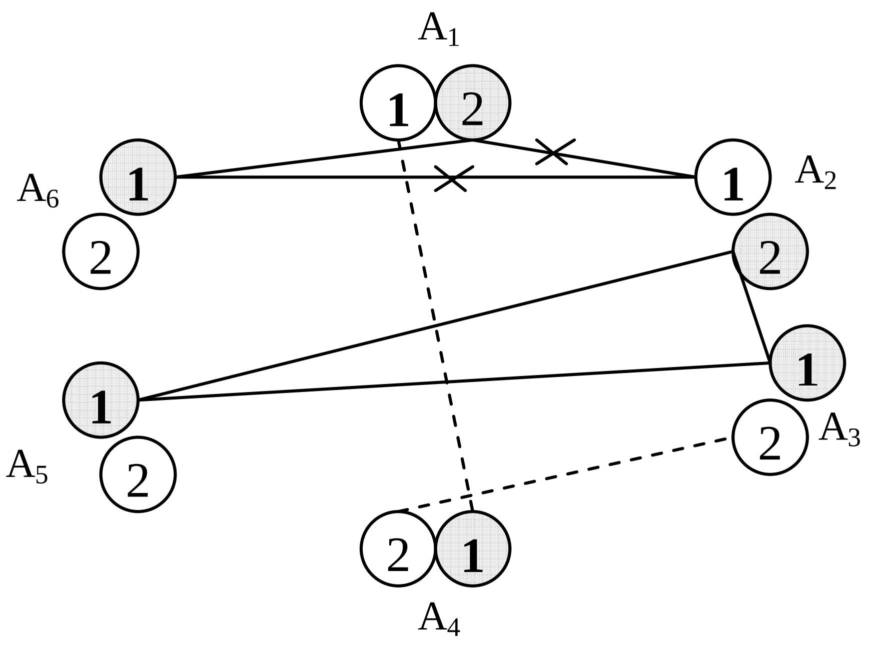


Figure 4.5: The coordination graph and active selection (gray circles) of the team $T = \{A_1, A_2, A_3, A_4, A_5, A_6\}$ after changed their values.

In order to satisfy definition 4.4.1, the diagnosis process needs to go beyond pinpointing suspect agents, to verifying that by changing their actions, coordination will be restored. Thus in contrast with consistency-based approach, we do not utilize conflict pairs to compute the diagnoses, but instead examine all action literals assignments that entail the coordination graph, i.e., all actions which will satisfy the coordination constraints. Formally:

Definition 4.4.2. *CG solution (S)*: given a coordination graph *CG* (definition 4.1.5), *CG solution (S)* is a set of assignments to the agents' actions (AS) which satisfy the coordination constraints in *CG*.

Once the entire CG solutions were found, the process compares the existing truth values to the CG solutions, and computes a *minimal* set of changes.

Algorithm `ABDUCTIVE_BASED_COORDINATION_DIAGNOSIS` (Algorithm 2) summarizes the process of computing the coordination diagnosis in an abductive based approach. In lines 1–2 we initialize Δ (a set of all diagnosis sets) and S (a set of all CG solution sets). Then in 3 we compute the CG solutions and initialize S with these sets, where S_i is the i'th CG solution set. This task is of course very hard, but will be discussed later. From line 4 we go over the CG solutions and check for each one of them, in line 6, if the active selection of agent A_i (as AS_i was defined in definition 4.1.2) is consistent with the solution. If it is not consistent, the agent which selects this inconsistent value (A_i) is added to a new diagnosis set Δ_{new}. Once the active selections of all the agents were checked, Δ_{new} is added to Δ as a new diagnosis if it is not a superset of an existing diagnosis. If Δ_{new} is a subset of an existing diagnosis then it replaces the diagnosis since it is minimal (lines 9–15).

Example 4.4.1. *Let us compute the AGCD of the coordination graph in Example 4.2.1. Table 4.1 presents the CG solutions for the actions of agents $A_1 \dots A_6$. There are only two such solutions. In order to find the minimal AGCD, we should compare the actions of the agents with the solutions and point out the agents that deviate. Consider the actions in Example 4.2.1 (where AS_{11}, AS_{21}, AS_{31}, AS_{41}, AS_{51}, AS_{61} are true, and the other action literals are false). Then, in the first solution S_1 $AS_{11} = false$, but actually the action taken by A_1 is $AS_{11} = true$. We thus mark action AS_{11} as faulty. The second value of S_1 is $AS_{12} = true$, but actually $AS_{12} = false$, so we again mark this as faulty, and so on for each one of the actions. For the first solution in the solutions table we got the following faulty actions: $AS_{11}, AS_{12}, AS_{31}, AS_{32}, AS_{51}, AS_{52}$. From this list, we can determine the abnormal agents by finding the agents whose actions are faulty. We thus conclude that a minimal AGCD is $\Delta_1 = \{A_1, A_3, A_5\}$ for S_1. From the second solution S_2, we similarly find $\Delta_2 = \{A_2, A_4, A_6\}$. Setting these agents to abnormal, and thus forcing them to select different actions, would satisfy the coordination constraints.*

Algorithm 2 ABDUCTIVE_BASED_COORDINATION_DIAGNOSIS
(**input**: coordination graph CG
output: diagnoses set Δ)

```
1: $\Delta \leftarrow \emptyset$
2: $S \leftarrow \emptyset$
3: initialize $S$ with all CG solutions
4: for all $S_i \in S$ do
5:    $\Delta_{new} \leftarrow \emptyset$
6:    for all $AS_j$, $j \in \{1 \dots n\}$ do
7:       if $\{AS_j\} \bigcup S_i \vdash \perp$ then
8:          $\Delta_{new} \leftarrow \Delta_{new} \cup \{A_j\}$
9:    if $\Delta_{new} \subset \Delta_x \in \Delta$ then
10:      remove $\Delta_x$ from $\Delta$
11:      $\Delta \leftarrow \Delta \cup \Delta_{new}$
12:   if $\Delta_{new} \supseteq \Delta_x \in \Delta$ then
13:      continue
14:   else
15:      $\Delta \leftarrow \Delta \cup \Delta_{new}$
16: return $\Delta$
```

#	A_1		A_2		A_3		A_4		A_5		A_6	
	1	**2**	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**
S_1	0	1	1	0	0	1	1	0	0	1	1	0
S_2	1	0	0	1	1	0	0	1	1	0	0	1

Table 4.1: **Coordination-satisfying actions in Example 6.**

Obviously, we should consider only the minimal *AGCD*. We fulfill this requirement by comparing every new hypothesized coordination diagnosis to the former coordination diagnoses, and checking whether it is a subset, a superset, or different than the former diagnoses.

Example 4.4.2. *Assume $T = \{A_1, A_2, A_3\}$. Every agent has two actions $D = \{1, 2\}$. The constraints between their actions are very simple, A_1 and A_2 should perform the same action concurrently ($CCRN(AS_{11}, AS_{21})$, $CCRN(AS_{12}, AS_{22})$) and A_3 is free to select either action 1 or action 2 does not matter which action was selected by A_1 and A_2. Table 4.2 presents the CG solutions for their actions. Consider the actions taken by the agents are: $AS_{12} = true$, $AS_{21} = true$, $AS_{32} = true$ (the other actions are false). Comparing the actual actions to the CG solutions* *in Table 4.2 entails the following diagnoses (corresponding to the order of the CG solutions):*

$\Delta 1 = \{A_1\}$

$\Delta 2 = \{A_1, A_3\}$

$\Delta 3 = \{A_2, A_3\}$

$\Delta 4 = \{A_2\}$

However, $\Delta_1 \subseteq \Delta_2$ *and* $\Delta_4 \subseteq \Delta_3$*, therefore the minimal diagnosis sets are only* Δ_1 *and* Δ_4*.*

#	A_1		A_2		A_3	
	1	**2**	**1**	**2**	**1**	**2**
S_1	1	0	1	0	0	1
S_2	1	0	1	0	1	0
S_3	0	1	0	1	1	0
S_4	0	1	0	1	0	1

Table 4.2: **Coordination-satisfying actions in Example 7.**

Let us analyze the complexity of abductive based global coordination diagnosis.

Theorem 3: *AGCD* (abductive based global coordination diagnosis) is NP-hard.

We can prove it by using again the reduction from "Independent set" as we used to prove that *CGCD* is NP-hard (Theorem 2). First we re-cast the definition of *AGCD* (4.4.1) as a decision problem[3].

Definition 4.4.3. *AGCD*: Given a set of variables $\{X_1, ..., X_n\}$, and a set of domain $\{D_1, ..., D_m\}$ for each variable with initial value $\{D_{w1}, ..., D_{w_n}\}$ correspondingly to the variables (parallel to *AS* in definition 4.1.2), and given a set of constraints between the variables in the forms:

1. $X_i = D_{y_i} \Rightarrow X_j = D_{z_j}$ (parallel to *CCRN* in definition 4.1.3)
2. $X_i = D_{y_i} \Rightarrow X_j \neq D_{z_j}$ (parallel to *MUEX* in definition 4.1.4)

Given $k_1 > 0$. Do there exist at most k_1 variables such that *changing* their value will satisfy the set of the constraints?

[3]Joint credit for this proof to Efrat Manistersky.

As in the original definition of $AGCD$ (definition 4.4.1) here we look for a diagnosis set of k_1 variables that represent the abnormal agents, such that changing their value will cause to satisfy the constraints.

The decision problem is presented almost in the same way as in Theorem 2, but while in $CGCD$ we require the *ignoring* of k_1 variables and the constraints associated with them, here, in $AGCD$, we require the *changing* of the value of k_1 variables. The rest of the reduction is the same as the reduction we presented in Theorem 2.

Thus the $AGCD$ problem is essentially that of finding all CG solution sets, an NP-Hard problem. A detailed discussion of satisfiability, and the rich literature offering efficient exact and approximate solution methods is well beyond the scope of this dissertation. However, we point at two diagnosis-specific mechanisms that can potentially be used to alleviate computational load in our case:

1. Ordered binary decision diagram (OBDD) [Bryant, 1992] can be used to efficiently reason about diagnosis-satisfying assignments [Torasso and Torta, 2003]. By restricting the representation, boolean manipulation becomes much simpler computationally. We can compactly represent the coordination graph using OBDDs (an off-line construction process), and then truth assignments can be computed in linear time in many cases.

2. Assumption-based truth maintenance systems (ATMS) [de Kleer, 1986] can be used to build the satisfying assignments incrementally. They exploit the fact that it is unnecessary to check all the assignments since the legal assignments depend each on the other. For instance, assume a concurrence coordination between a and b and between b and c:

   $$\begin{array}{rl} & ((a \wedge b) \vee (\neg a \wedge \neg b)) \\ \bigwedge & ((b \wedge c) \vee (\neg b \wedge \neg c)) \end{array}$$

   Instead of computing the full truth table of a, b and c, (2^3), we can use an ATMS, which given these justifications will provide only two assignments: $(a = true, b = true, c = true)$ or $(a = false, b = false, c = false)$.

The abductive approach has some advantages over consistency-based approach:

1. It computes a sound and complete diagnosis, in contrast to consistency-based approach which computes only a complete diagnosis.

2. It finds also a solution to the fault by proposing a satisfaction to the coordination constraints.

3. It also enables to find the consistency assignments of the coordination graph in off-line, then in on-line the diagnosis is given in a linear time, while the process of the consistency-based approach must computed online.

Despite these advantages, consistency-based approach has an important benefit considering the complexity. In the worst case the complexity of the abductive approach is $O(k^n)$ while k represents the size of the variables domain and n the size of the team, while the complexity of the first approach in the worst case is $O(2^n)$. In addition, the runtime of computing the vertex cover set is reduced as the number of conflicts is reduced. Finally, every vertex cover that is found reduces the size of the graph for the continuing computing of other vertex cover set in larger size. For instance, if a was found as a vertex cover in a graph $V = \{a, b, c, d\}$, then when we look for vertex cover sets in size two, we will compute it only in the subgraph $V = \{b, c, d\}$.

Chapter 5

Distributed Social Diagnosis

In the previous chapter we focused on centralized methods for diagnosis of coordination failures. Unfortunately, centralized methods suffer from key limitations: First, they can be computationally expensive in practice, in terms of communications and run-time. Second, they rely on a single diagnoser, and thus risk a single point of failure. Moreover, this assumes no communication limitations, e.g., range. Finally, they do not not necessarily exploit the different knowledge of different agents; e.g., an intended receiver faces difficulty detecting that a message to it was lost, where the sender may do it more readily.

In the following chapter we focus on abductive coordination diagnosis utilizing distributed model-based diagnosis algorithms to compute the diagnosis, based on distributed CSP algorithms. While the reasoning behind all is the same as outlined above, the algorithms differ from each other with respect to their expected run-time (based on DisCSP literature) and their completeness of the diagnoses.

We present a distributed approach, where the agents find the CG solutions (see definition 4.4.2) and compute the abductive coordination diagnosis by exchanging information with each other. As in the centralized approach, computing the diagnosis is done by finding the CG solutions, and contrasting these with actual values. As far as we know, there is no existing algorithm which finds a minimal CG solutions (in terms of minimal diagnosis), where no proper subset of the changed values could also satisfy the constraints. Thus the minimality goal is preserved only for some algorithms (see below).

In the next two sections we propose five distributed algorithms to find CG solutions and compute the diagnosis. All the algorithms use communication, therefore they work only in nonpermanent communication breakdowns. In permanent communication breakdowns neither distributed nor centralized approach will work. In

Section 5.1 we present two algorithms for computing a *complete* set of minimal diagnoses, and in Section 5.2 we present three algorithms for computing an *incomplete* diagnosis which is not guaranteed to be minimal. As we shall see, these can offer an attractive alternative, despite their lack of guarantees.

5.1 Algorithms for Complete Minimal Diagnoses

In order to compute a complete set of minimal diagnoses, the agents must compute the whole CG solution space of the system. We use a synchronous backtracking algorithm (SBT) to compute the CG solutions [Yokoo *et al.*, 1998]. This algorithm is based on a distributed depth-first search. The agents are arranged in a static order. Every agent sends its possible values to its next agent. The receiving agent checks the compatibility of the former assignments with every value of its domain, separately. It returns backward a *nogood* message upon inconsistency, or the partial assignments to the next agent, upon consistency.

Example 5.1.1. *Let us demonstrate it by simplifying the example given in the introduction to this part. Focusing on a case with a single team, $T = \{B, C_1, D_1, D_2\}$, where D_1, D_2 are followers, C_1 is a sub-team leader, and B is a mission commander, we define the domain of the agents to be the actions go (g) or stop (s), $d = \{g, s\}$. The coordination constraints between the agents are:*

CCRN: $\langle C_1 = g, D_1 = g \rangle, \langle C_1 = g, D_2 = g \rangle$

MUEX: $\langle B = s, C_1 = g \rangle, \langle B = g, C_1 = s \rangle$

The appropriate coordination graph is presented in Figure (Figure 5.1).

Assume the agents are arranged in an alphabetic order. B sends the possible values of its domain to C_1. C_1 checks for each one of them whether it is consistent with its possible values. For the value $B = g$ only $C_1 = g$ is consistent and for the value $B = s$ only $C_1 = s$ is consistent. C_1 continues to forward the partial consistent values to D_1, which adds its consistent values and produces the following partial consistent values: $S_1 = \{g, g, g\}$ and $S_2 = \{s, s, s\}$ (corresponding to the order of the agents (B, C_1, D_1)). D_2 receives S_1 and S_2 from D_1 and accomplishes the computing of the CG solutions $S_1 = (g, g, g, g)$ and $S_2 = (s, s, s, s)$ (corresponding to the order of the agents (B, C_1, D_1, D_2)).

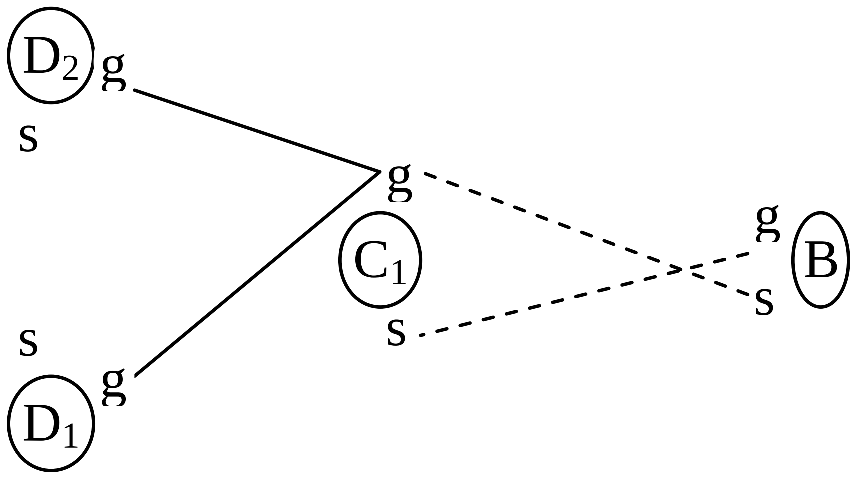


Figure 5.1: Coordination graph for mission commander B, sub-team leader C_1, and the two followers (version 1).

In a system where the constraints between the agents are static, i.e. they do not change dynamically, the agents could compute all the CG solutions in advance (offline). During run-time, every agent keeps a copy of all CG solutions, using them to compute the diagnosis. We denote this method *SBT_OFF*. On the other hand, in systems where the constraints can change dynamically, the agents must compute the CG solutions, as well as the diagnosis, online. We denote this *SBT_ON*.

During diagnosis, every agent reports to the other agents the indexes of the CG solution database in which that agent found an inconsistency. Every agent collects this information from the others and computes the diagnoses by dividing the agents according to the reported indexes. So as to produce minimal diagnoses, if a diagnosis set is a superset of another diagnosis, it is dropped.

Algorithm `DISTRIBUTED_COORDINATION_DIAGNOSIS` (Algorithm 3) summarizes the process of computing the coordination diagnosis in a distributed approach. The algorithm gets the index of the agent who runs the algorithm (*current*). In lines 1–2 we initialize Δ—a set of all diagnosis sets and S—a set of all CG solution sets. Then in line 3 we compute the CG solutions and initialize S with these sets, where S_i denotes the i'th CG solution set. In line 4 we initialize $INDX$, a set of the indexes of the CG solution database in which agents found an inconsistency. $INDX_i \in INDX$ denotes the indexes of agent A_i. In lines 5–7 the agent that runs this algorithm ($A_{current}$) builds its $INDX_{current}$ by checking the consistency of each CG solution with its current value ($AS_{current}$). If an inconsistency is found, the index of the CG solution is added to $INDX_{current}$. Then it sends the set of the indexes of the inconsistency to the other agents and receives from them the same information (lines 8–10). By receiving this set, the current agent goes over the indexes in which the sender agent (A_i) found an inconsistency, and add it as abnormal agent to the diagnosis set Δ_j with the same index j (lines 11–12). In this way we verify that Δ_j contains all the agents that found inconsistency with the j'th CG solution. The current agent makes the same sort to Δ_j also for its own indexes $INDX_{current}$ (lines 13–14). Finally, in lines 15–20 the agent verifies minimal diagnosis by dropping superset diagnoses.

Algorithm 3 DISTRIBUTED_COORDINATION_DIAGNOSIS

```
(input: integer current
output: diagnoses set $\Delta$)
1: $\Delta \leftarrow \emptyset$
2: $S \leftarrow \emptyset$
3: initialize $S$ with all CG solutions
4: $INDX \leftarrow \emptyset$
5: for all $S_i \in S$ do
6:    if $\{AS_{current}\} \bigcup S_i \vdash \perp$ then
7:       $INDX_{current} \leftarrow INDX_{current} \cup \{i\}$
8: for all $A_i \in T$, $i \in \{1 \dots n\}$ $(i \neq current)$ do
9:    send $INDX_{current}$ to $A_i$
10:   receive $INDX_i$ from $A_i$
11:   for all $j$, $j \in INDX_i$ do
12:      $\Delta_j \leftarrow \Delta_j \cup \{A_i\}$
13: for all $j$, $j \in INDX_{current}$ do
14:   $\Delta_j \leftarrow \Delta_j \cup \{A_{current}\}$
15: for all $\Delta_i \in \Delta$, $i \in \{1 \dots |\Delta| - 1\}$ do
16:   for all $\Delta_j \in \Delta$, $j \in \{i \dots |\Delta|\}$ do
17:      if $\Delta_i \subset \Delta_j$ then
18:         remove $\Delta_j$ from $\Delta$
19:      if $\Delta_i \supseteq \Delta_j$ then
20:         remove $\Delta_i$ from $\Delta$
21: return $\Delta$
```

Example 5.1.2. *As computed in example 5.1.1 the CG solutions for the coordination graph are: $S_1 = (g, g, g, g)$ and $S_2 = (s, s, s, s)$ (corresponding to the order of the agents (B, C_1, D_1, D_2)). Assume that follower D_1 failed due to a failure in its vision, which caused it to select the action stop (s). The agents exchange the CG solution* *indexes in which they found an inconsistency. B sends index 2 since its current value is g which is not equal to its expected value in CG solutions S_2. In the same manner, C_1 sends index 2, D_1 sends index 1 and D_2 sends index 2. Once an agent accepts this information from all the others, it divides them according to the indexes, to form two diagnoses: $\Delta_1 = \{D_1\}$ and $\Delta_2 = \{B, C_1, D_2\}$.*

In the same manner, we address also of cases where the minimal diagnosis is a subset of another diagnosis.

Example 5.1.3. *Assume the coordination constraints between the agents in the team presented in example 5.1.1 are as the following:*

CCRN: $\langle C_1 = g, D_1 = g\rangle, \langle C_1 = g, D_2 = g\rangle$

MUEX: $\langle B = s, C_1 = g\rangle$

The appropriate coordination graph is presented in Figure (Figure 5.2).

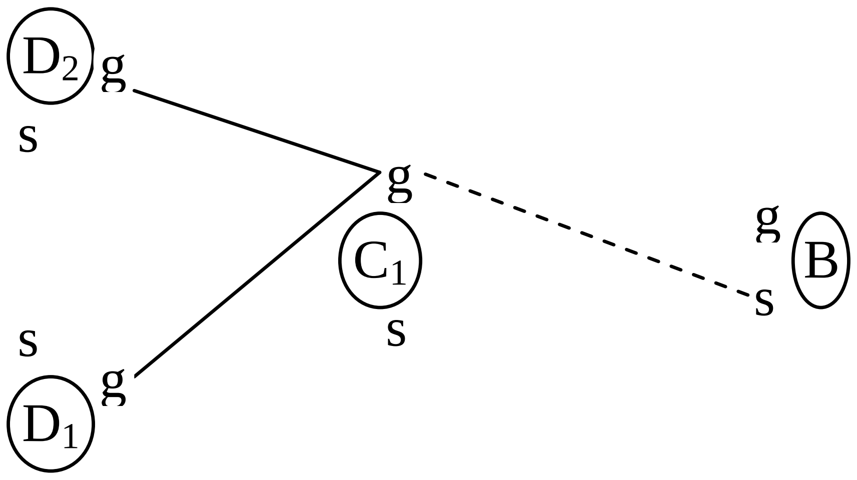


Figure 5.2: Coordination graph for mission commander B, sub-team leader C_1, and the two followers (version 2).

CG solution $S_3 = (g, s, s, s)$ also satisfy the team variables. Then the indexes sent by the agents are as the following:

$B : 2$

$C_1 : 2, 3$

$D_1 : 1$

Diagnosis	communication	computation
SBT_ON	$O((k^n) + (n^2m))$	$O((k^n) + (nm + m^2))$
SBT_OFF	$O(n^2m)$	$O(nm + m^2)$

Table 5.1: The worst case complexity of communication and computation for SBT_ON and SBT_OFF.

$D_2 : 2, 3$

Dividing the agents according to the indexes produces the following diagnoses: $\Delta_1 = \{D_1\}$*,* $\Delta_2 = \{B, C_1, D_2\}$ *and* $\Delta_3 = \{C_1, D_2\}$*. However,* $\Delta_3 \subseteq \Delta_2$ *and so* Δ_2 *is dropped. Then the final minimal diagnosis sets are:* $\Delta_1 = \{D_1\}$ *and* $\Delta_3 = \{C_1, D_2\}$*.*

The first stage of building the CG solution database, involves an exponential number of messages and its computation is also exponential in the number of agents. Assume n is the number of agents and k represents the size of the variables domain. The first agent communicates its variables to the second agent, which also communicates to the next agent the possible combinations of its variables with the variables of the former agent—k^2. The i'th agent communicates k^i combinations of the former agents with its own variables. As a result, the worst case complexity of the communication is the sum of all possible combinations of the n agents or $O(k^n)$. SBT is a synchronous process since agents do not make computations in parallel, therefore, the worst case complexity of the computation is exactly as in centralized approach $O(k^n)$.

However, the second stage of the diagnosis process itself entails only the exchanging of the indexes of the CG solutions in which the agents found an inconsistency. Assume the size of the CG solution database is m, then the communication complexity is only $O(n^2m)$. The rest of the complexity of computation is linear in the number of agents since every agent goes over the indexes of the other agents in order to sort them to diagnosis sets—$O(nm)$. However, in order to guarantee minimal diagnosis the agent goes over the diagnosis sets polynomially and so the final complexity in the worst case is $O(nm + m^2)$.

Table 5.1 summarizes the online communication and communication worst case complexities of SBT_ON and SBT_OFF. In systems where the constraints are static (SBT_OFF), these costs are most delegated to offline processes. However, where constraints change dynamically, the agents must compute all the CG solutions dynamically, and these computational costs are incurred during runtime.

Indeed, distributed CSP literature recognizes the computational costs of SBT because it runs synchronously, and offers cheaper alternatives [Yokoo and Hirayama, 2000]. In addition, SBT has drawback in some domains, e.g. wireless communication which has limited range, because there are technical limitations which allow certain local communication but not global communications [Davin and Modi, 2005].

5.2 Non-Minimal Diagnosis

One alternative taken by many distributed CSP algorithms is to settle for computing only one solution to a given CSP. However, for diagnosis, this means that the results are not guaranteed to be minimal. Moreover, since only one of possibly many diagnoses would be produced, the result may not even be correct. But, since the system is distributed, we can use distributed algorithms to compute the CG solution, and thus reduce the runtime and communication load. As in SBT, once a CG solution is found, the agents compute the diagnosis by comparing their current values to the expected values in the CG solution. The deviant agents are suspected as the abnormal agents, and will be added to the diagnosis set. We examine three distributed CSP algorithms: Asynchronous backtracking, distributed stochastic search and distributed breakout algorithm.

5.2.1 Asynchronous Backtracking (ABT)

The asynchronous backtracking algorithm is a distributed, *asynchronous* version of a backtracking algorithm (SBT). In ABT, the priority order of agents' variables is fixed, and each agent communicates its value assignment to neighboring agents with lower priority via *ok*? messages. Each agent maintains an *agentview*, the current value assignment of higher priority neighboring agents. An agent changes its assignment if its current value assignment is not consistent with the assignments in the *agentview*. If there exists no value that is consistent with the *agentview*, the agent generates a new constraint (called a *nogood*), and communicates the *nogood* to a lower priority agent in the *agentview*, thus the higher priority agent changes its value considering the accepted *nogood* (for detailed description see [Yokoo and Hirayama, 2000]).

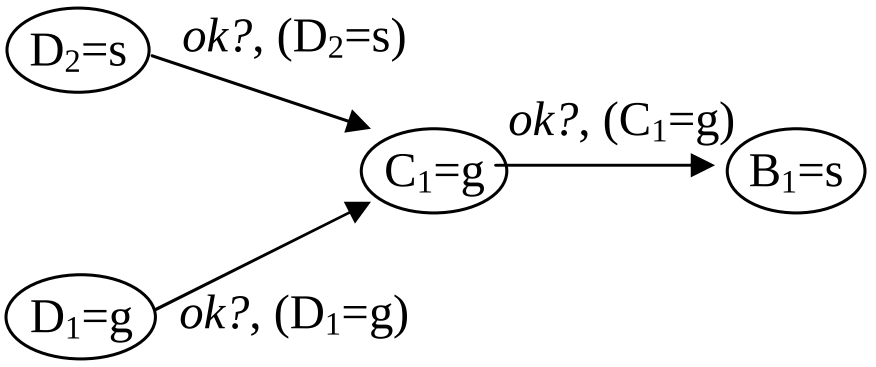
Example 5.2.1. *Assume a team with coordination constraints as presented in example 5.1.1. Assume also two failures: B stops while the sub-team leader C_1 continues to go, concurrently one of the followers (D_1) follows the leader while the other (D_2)*

stops ($\{B = s, C_1 = g, D_1 = g, D_2 = s\}$). The priority between the agents is arranged in alphabetic order. C_1 sends ok? message with its current value g to its lower priority neighbor B. D_1 and D_2 send ok? with their values (g, s, correspondingly) to C_1 (Figure 5.3(a), the arrows represent the communication links). The agentview of B is $\{C_1 = g\}$ and so it changes its value to g to satisfy the constraints with C_1. C_1 could not find consistent value with its agentview $\{D_1 = g, D_2 = s\}$, and so it sends nogood to the lower priority agent in its agentview—D_1 (bold line in Figure 5.3(b)). D_1 adds this nogood constraint $\langle D_1 = g, D_2 = s\rangle$ to its constraints database and requests from D_2 to add a constraint link between them (dashed line in Figure 5.3(c)). Now, D_1 changes its value to g in order to be consistent with its new agentview $D_2 = g$ (Figure 5.3(c)), and sends ok? with its new value to C_1, and a solution is found ($\{B = g, C_1 = g, D_1 = g, D_2 = g\}$).

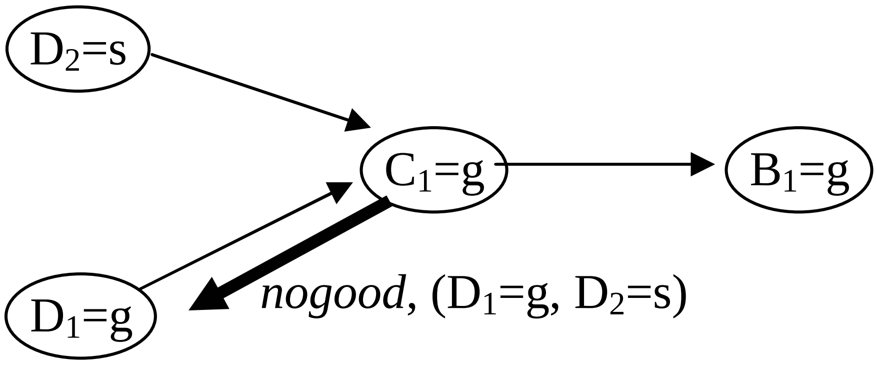
ABT is complete in terms of CSP. It always finds a solution if one exists, and terminates if no solution exists, so we are guaranteed to find one diagnosis. However, still it has three drawbacks, first, we cannot be sure in advance which agents will communicate with each other, since an agent that detects a *nogood* constraint with non–neighboring agent adds communication channel to it (as added between D_2 and D_1 in Example 5.2.1). Second, in SBT at the end of the diagnosis process each one of the agents has the entire minimal diagnosis sets, but here, in ABT, each agent may have only a portion of the diagnosis, related only to its *agentview*. Third, once a CG solution is found, the agents do not know that the search was completed. The next algorithms copes with some of these drawbacks.

5.2.2 Distributed Stochastic Search Algorithm (DSA).

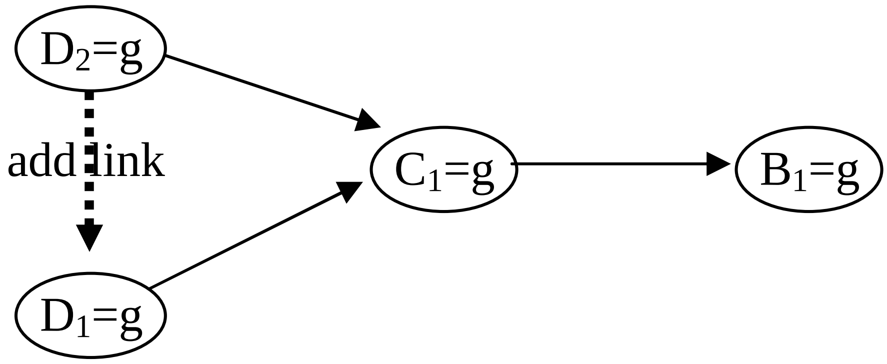
In contrast to ABT, DSA is synchronous in that all processes proceed in synchronized steps. The agents go through a sequence of steps until a termination threshold is met (for example, limited number of cycles). In each step, an agent sends its current variable value to its neighboring agents, and concurrently receives the values from the neighbors. It then decides stochastically, whether to keep its current value or change to a new one. This is done based on a pre-defined strategy that depends on the possibility to reduce violated constraints. The most critical step of DSA is for an agent to decide the next value, based on its current state and its perceived states of the neighboring agents. The decision strategy we utilized is the following: If the agent cannot find a new value to improve its current state (reduces violations), it will not change its current value; if there exists such a value that improves



(a) Sending *ok?* messages.



(b) C_1 sends *nogood* to D_1.



(c) D_2 agrees to the request of D_1 and adds a constraint link to D_1.

Figure 5.3: ABT process in team $T = \{B, C_1, D_1, D_2\}$.

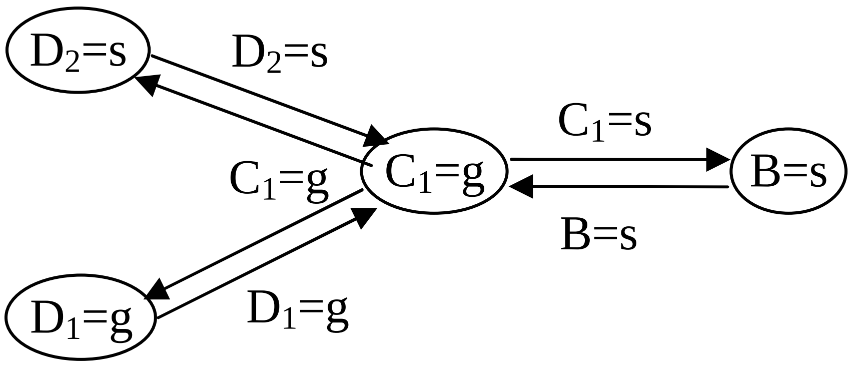
its state, the agent may change to the new value with probability p, or keep the current value unchanged with probability $1 - p$. This continues until a termination threshold is reached (i.e., a certain number of cycles) (for detailed description see [Zhang *et al.*, 2005]).

Example 5.2.2. *Assume a team with coordination constraints and failures as described in example 5.2.1. The values of the agents' variables are* $\{B = s, C_1 = g, D_1 = g, D_2 = s\}$. *Each agent sends its value to its neighbors (Figure 5.4(a)),* *then the agents compute the number of violated constraints with their neighbors. B has one violation with C_1 and it can improve it by changing its value to g. C_1 has two violations (with B and D_2) and it could reduce it to only one violation with D_1 by changing its value to s. D_1 has no violations and D_2 has one violation with C_1 that could be solved by changing its value to g. Assume the probability to change value is $p = 70\%$, and in the first cycle every agent, except D_1 (which could not improve its current state), decides to change its value. Then their new values are $\{B = g, C_1 = s, D_1 = g, D_2 = g\}$. The agents communicate again their new value to their neighboring agents (Figure 5.4(b)); now all the agents can reduce their violations. Assume, in the second cycle only B and C_1 change their values. Then after receiving the new values ($\{B = s, C_1 = g, D_1 = g, D_2 = g\}$), D_1 and D_2 have no violation, c_1 has one violation with B but no violation with D_1 and D_2, so C_1, D_1 and D_2 should not change their values since they can not improve their current state. However, B can improve it by changing its value to g, indeed it changes it and a CG solution is found. The agents communicate their new values in the last time since the termination condition of number of cycles is defined in advance to be 3 (Figure 5.4(c)).*

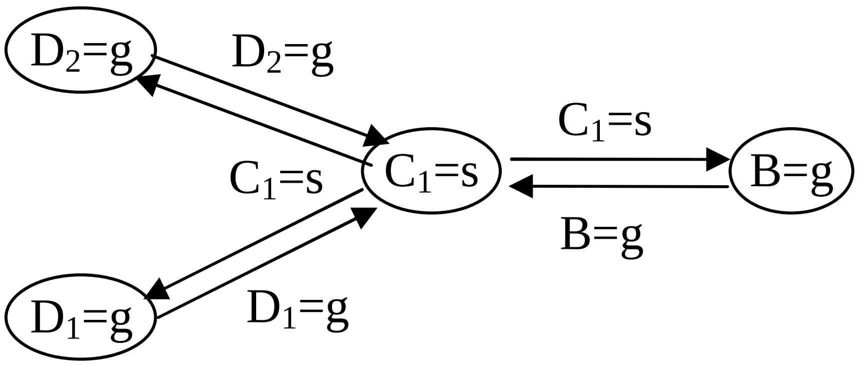
One drawback of this algorithm is that it is incomplete, namely, it could find a solution before the threshold is met, although the agents will continue to communicate until the threshold will be met. To further complicate the issue, the threshold may be met before a solution is found although one exists. For example, in Example 5.2.2 if the threshold was set to 2 then no solution would be found. However, it copes with some disadvantages of ABT. First we know in advance the communication channels of every agent (neighboring agents) and they are not changed dynamically. Second, if an agent is diagnosed as abnormal, this diagnosis is known to the abnormal agent and its neighboring agents. Third, the termination threshold is known to all the agents.

5.2.3 Distributed Breakout Algorithm (DBA)

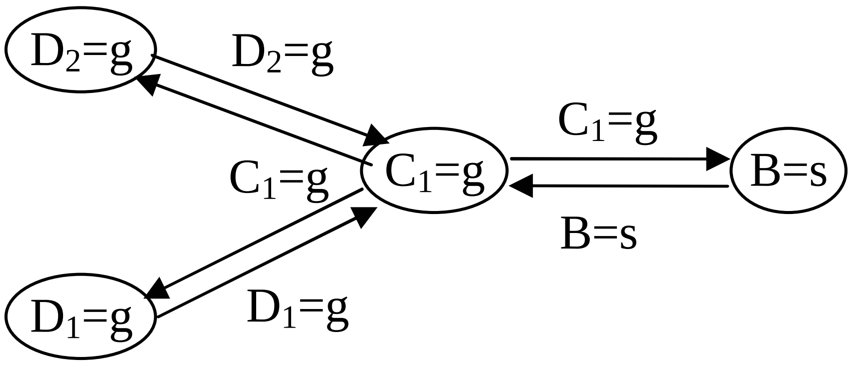
The breakout algorithm is also a local search method equipped with an innovative scheme of escaping local minima for CSP. As in DSA, (i) an agent communicates only with its neighbors, (ii) each step is synchronized, and (iii) the agents go through a sequence of steps until a termination threshold is met. However, local minima are addressed non-stochastically.



(a) Cycle 1: agents send values to their neighboring agents.



(b) Cycle 2: agents send values to their neighboring agents.



(c) Cycle 3: agents send values to their neighboring agents.

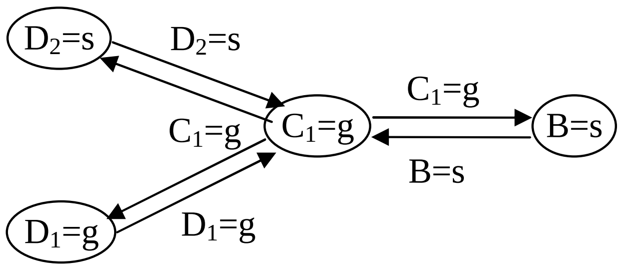
Figure 5.4: DSA process in team $T = \{B, C_1, D_1, D_2\}$.

Every agent first assigns a weight of one to all the constraints with its neighbors. Then, at each step, every agent exchanges its current variable value with its neighbors, and then computes the improvement (possible weight reduction) based on the recorded values of its neighbors. To avoid simultaneous variable changes among neighboring agents, before changing the value, the agents exchange the expected improvement. After collecting the expected improvement of the neighbors only the agent having the maximal improvement has the right to alter its current value. If some agents have the same improvement value, the one with the lowest identifier will change its value. An agent continues the process of value and improvement exchanging, until no weight-reducing variable can be found. At that point, it reaches a local minimum if a constraint violation still exists. Then, the agent tries to escape from the local minimum by increasing the weights of all violated constraints by one and proceeds as before. This weight change forces some of the agents to alter their values to satisfy the violated constraints (for detailed description see [Hirayama and Yokoo, 2005]).

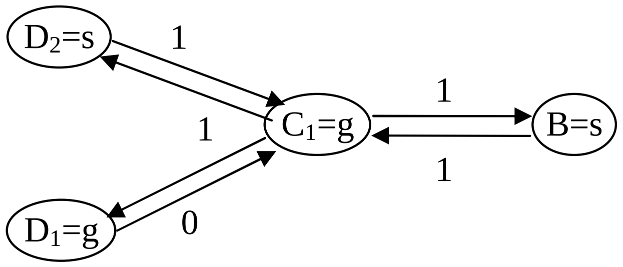
Example 5.2.3. *Assume a team with coordination constraints and failures as described in example 11. The values of the agents' variables are* $\{B = s, C_1 = g, D_1 = g, D_2 = s\}$. *Each agent sends its value to its neighbors, then the agents compute the number of violated constraints with their neighbors and the expected improvement by changing its value (Figure 5.5(a)). B has one violation with* C_1 *and it can improve it by changing its value to g.* C_1 *has two violations (with B and* D_2*) and it could improve it by one by changing its value to s.* D_1 *has no violations and* D_2 *has one violation with* C_1 *that could be solved by changing its value to g (Figure 5.5(b)). After exchanging the improvement values, B,* c_1 *and* D_2 *observe that they can improve their value by one. Consequently, only B changes its value since it has the lowest identifier (Figure 5.5(c)). The same process is done again. Agents exchange their values, only* D_2 *could improve its current state by changing its value to g (*c_1 *has one violation but its other two links are satisfied). The agents exchange this information (Figure 5.5(d)) and only* D_2 *changes its value and a CG solution is found. In Figure 5.5(e) the agents continue to exchange information without changing their values (since a solution was found and no improvement exist) until the termination condition is met.*

DBA is also incomplete, so we are not guaranteed to find a diagnosis. But, since it is a local search method we are expected to reduce the communication and computation significantly, compared to SBT. In addition, it has the same advantages as DSA:

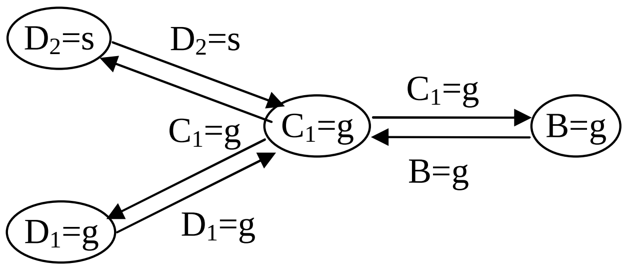
1. An agent communicates only with its neighboring agents.
2. If an agent is diagnosed as abnormal, this diagnosis is known to the abnormal agent and its neighboring agents.
3. The termination threshold is known to all the agents.



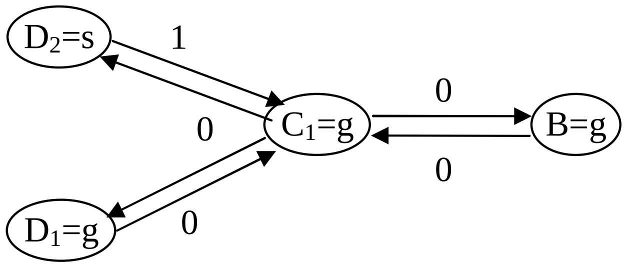
(a) Cycle 1: agents send values to their neighboring agents.



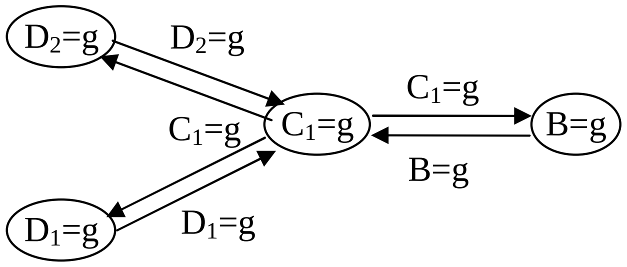
(b) Cycle 2: agents exchange their improvement by changing the current value.



(c) Cycle 3: agents send values to their neighboring agents.



(d) Cycle 4: agents exchange their improvement by changing the current value.



(e) Cycle 5: agents send values to their neighboring agents.

Figure 5.5: DBA process in team $T = \{B, C_1, D_1, D_2\}$.

Chapter 6

Coordination Diagnosis: Experiments and Discussion

This chapter evaluates the distributed diagnosis algorithms we presented, in terms of computation and communication. In addition, we examine, for every algorithm, the trade-off between its computational costs and its ability to produce correct diagnosis.

Table 6.1 summarizes the attributes of the different algorithms, in terms of the minimality of the diagnosis and completeness. SBT_ON and SBT_OFF look for the whole CG solution space and so they guarantee complete diagnosis and minimality, in contrast to the algorithms that find only one CG solution (AST,DSA,DBA). However, ABT like the complete algorithms, guarantees a complete CG solution, in contrast to the local search algorithms (DSA and DBA) that do not necessarily find even one CG solution.

We created laboratory versions of the space exploration example described previously. We evaluated every algorithm in different size groups: four robots, seven robots and ten robots. In the experiments for four robots, the group consisted of

Algorithm	**number of diagnoses**	**complete CG solution**	**complete diagnosis**	**minimal diagnosis**
SBT_ON	all	yes	yes	yes
SBT_OFF	all	yes	yes	yes
ABT	1	yes	no	no
DSA	1	no	no	no
DBA	1	no	no	no

Table 6.1: The minimality and completeness of the diagnosis algorithms.

a mission commander and a sub-team consisting of one sub-team leader and two followers. The group of seven robots consisted of a mission commander and two sub-teams, and the group of ten robots consisted of a mission commander and three sub-teams.

In order to evaluate the algorithms on a representative and diverse set of problems, a wide set of combination of potential failures was selected. First, we generated all single-faults possible (1–7 in the list below). Note that we assume all followers/leaders are the same, so it does not matter which follower/leader has failed. Then we created double-fault combinations (8–12), and a quadruple failure (13). For the experiments with a single team, only the experiments marked with a star are possible:

1. A follower thinks that the leader stops, although the leader continues to go (*).
2. A follower thinks that the leader started to go although it actually did not (*).
3. A sub-team leader thinks that it got a message from the mission commander to stop, although the message was not sent (*).
4. A sub-team leader thinks that it got a message from the mission commander to go, but the message was not sent (*).
5. The mission commander sent a message to the sub-team leaders, but only some of them received it.
6. A follower stops because of an individual technical problem (nothing to do with coordination) (*).
7. A leader stops because of an individual technical problem (nothing to do with coordination) (*).
8. Failure 1 above, in two different followers (*).
9. Failure 2 above, in two different followers (*).
10. Failures 3 and 4 above (one in each sub-team).
11. Failure 5 above for two sub-team leaders.
12. Failure 6 above for two different followers (*).

13. Failure 9 above (twice, for two different followers), and failure 10 above (twice, for two different sub-team leaders).

Failures 6 and 7 reflect a local fault but not a coordination fault, since the action values of the robots in the group remain the same. In particular, although the robot stopped, it did not select the "stop" action; it believes that its current action should be "go", but actually it stopped due to technical problem. For these failures, we expect the diagnosis process to find that the agents' values satisfy the constraints and therefore the agents will continue to diagnose the fault locally. This process is beyond the scope of this thesis.

To evaluate the performance of the algorithms from a computational perspective, two independent measures of performance were used. We measured communication load in terms of the total number of messages sent [Lynch, 1996]. We also measured runtime in terms of non-concurrent constraint checks (cycles) [Meisels *et al.*, 2002, Zivan and Meisels, 2006]. Every agent holds a counter of computation steps. Every message carries the value of the sending agents counter. When an agent receives a message it updates its counter to the largest value between its own counter and the counter value carried by the message. By reporting the cost of the search as the largest counter held by some agent at the end of the search, we achieve a measure of concurrent search effort.

Each of the test-case failures is different, and for all algorithms other than DSA, a single run is sufficient to determine the results, since no randomization takes place, and no noise is involved in the observations or deterministic decisions of the algorithms. However, for DSA (which is a stochastic algorithm), results may change between runs, even starting with the same initial conditions. For DSA, we therefore run every experiment 30 times and takes the average. The termination threshold for DSA and DBA was set to the number of robots in the team (below we will present results using a lower—fixed—termination threshold).

Experiments with four robots were carried out on Sony Aibo robots (Figure 6.1). These experiments were then repeated using the Player/Stage software package [Gerkey *et al.*, 2003] simulator, a popular and practical development tool for robotics (Figure 6.2). We verified that the results of the physical and simulated robots (group of four) were identical, and then continued the experiments in larger groups in simulation. Experiments using the DSA were all carried out using the simulator (because of the need for a significant number of repeated trials).

In Table 6.2 we present the results of a single test (test #7) for a team of seven

Algorithm	**Diagnoses**	**Messages**	**Runtime**	**% Failed robots**
SBT_OFF	{4,7,8,9},{0,5,6}	42	30	0
SBT_ON	{4,7,8,9},{0,5,6}	256	700	0
DBA	{0,5,6}	108	113	0
ABT	{0,5,6}	18	7	0
DSA	{0,4}	108	114	28.6

Table 6.2: results of diagnosing a specific failure case (test #7). The ground truth diagnosis in this case is $\{0, 5, 6\}$.

agents. The first column reports the method used. The second column shows the diagnosis results. The third column presents the number of messages sent. The next column summarize runtime in terms of non-concurrent constraint checks, and the last column reports the percentage of robots that failed to find a solution to the DisCSP. The ground truth diagnosis in this case is $\{0, 5, 6\}$ (the numbers identify the robots), i.e., the robots $\{0, 5, 6\}$ caused to the failure.

For instance, the diagnosis sets computed by SBT methods are $\{4, 7, 8, 9\}$ and $\{0, 5, 6\}$. DBA and ABT compute only the second diagnosis set since they compute only a single diagnosis and DSA failed to compute a diagnosis at all (it is incomplete). The total number of messages sent by agents in DBA is 108 and the number of non-concurrent constraint checks is 113. DSA reports on 28.6% of the agents that failed to compute a solution, the other methods succeed to compute a diagnosis (although DBA is also incomplete in this case the agents found a solution).

The results of the communication load and the runtime are presented in Figure 6.4 and Figure 6.3, respectively. The x axis shows the diagnosis algorithm and the y axis presents the total number of messages sent (Figure 6.4) and the runtime (Figure 6.3). For each algorithm, three bars are shown, one for each of the group sizes. Each bar represents the average results across the different failures.

As expected, computing all the CG solutions online (SBT_ON) is expensive in terms of both communication as well as computation. Obviously, computing the CG solutions offline (SBT_OFF) and then finding the diagnosis online, improves the efficiency significantly. SBT_OFF is even better than the local search algorithms, DSA and DBA, although it computes a complete set of diagnoses and not a single one. The reason for this is that in SBT_OFF the agents communicate only the indexes of the inconsistent CG solutions, but the solutions are computed offline.

DSA and DBA present equal values in the number of messages (Figure 6.4),

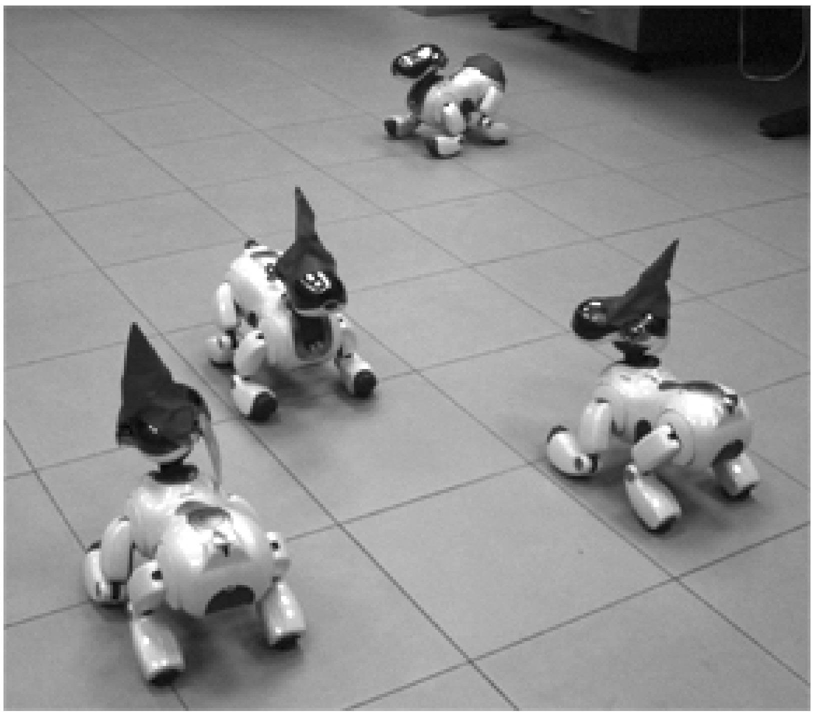


Figure 6.1: Sony Aibo robots capturing a mock alien.

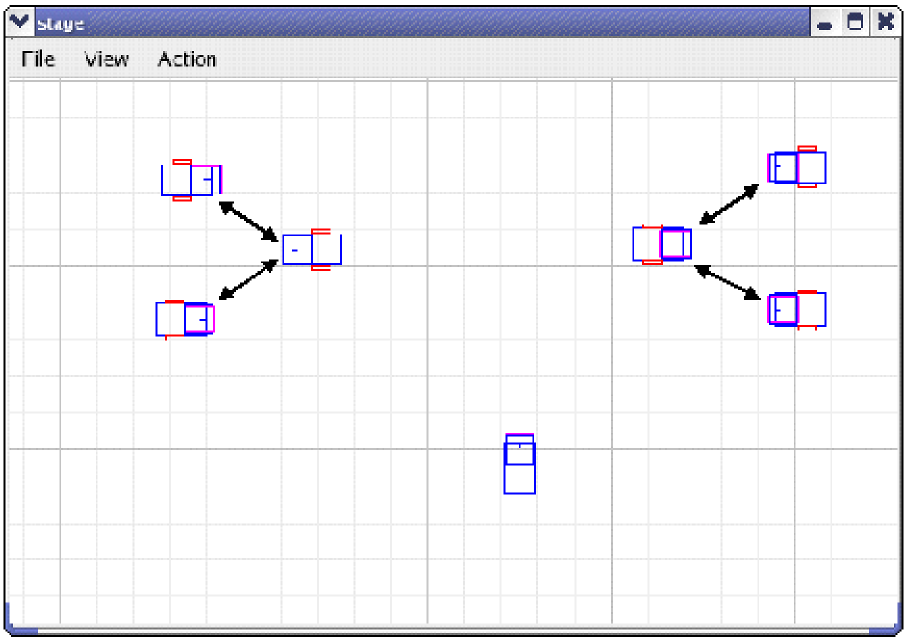


Figure 6.2: Screen shot of Stage simulator in action.

since in both algorithms every cycle the agents communicate their values to their neighboring agents and they have the same termination threshold on the number of cycles. However, considering the time cycles DBA uses less constraint checks than DSA since in DSA the gents change their values stochastically. It is possible that an agent does not change its value although it should have changed it. Because of this, its neighbors must again check the constraints between them.

Surprisingly, ABT outperforms DSA and DBA. These results are surprising in light of previous research that showed that the local search algorithms are more efficient than ABT [Zhang *et al.*, 2005]. This has to do with the likely state of a multi-agent system after a coordination failure. In a team that was in coordination and then failed, the selected actions of most agents are likely going to be close to the CG solution. This enables ABT to find a CG solution in only a few steps. On the other hand, in DSA and DBA the search may proceed towards a different part of the space; also, the termination threshold may cause DSA and DBA to continue running needlessly (see below for experiments with a reduced threshold).

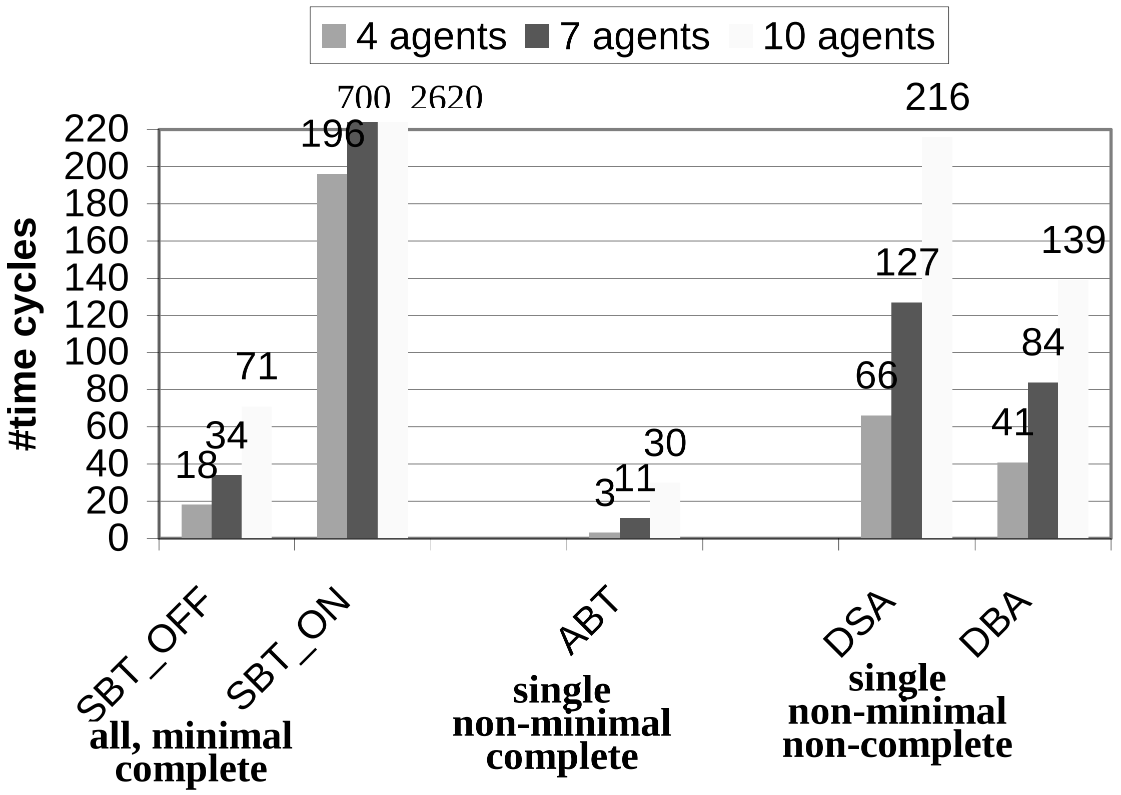


Figure 6.3: Average number of cycles in different diagnosis methods.

In order to further evaluate the diagnosis algorithms we examine also the correctness of the diagnoses they produce. SBT_ON and SBT_OFF produce a complete set of minimal diagnoses. However, the other algorithms produce only a single diagnosis. This diagnosis is not guaranteed to be minimal and thus to correctly explain the fault(s). In this sense, ABT is better than DSA and DBA, since it is complete

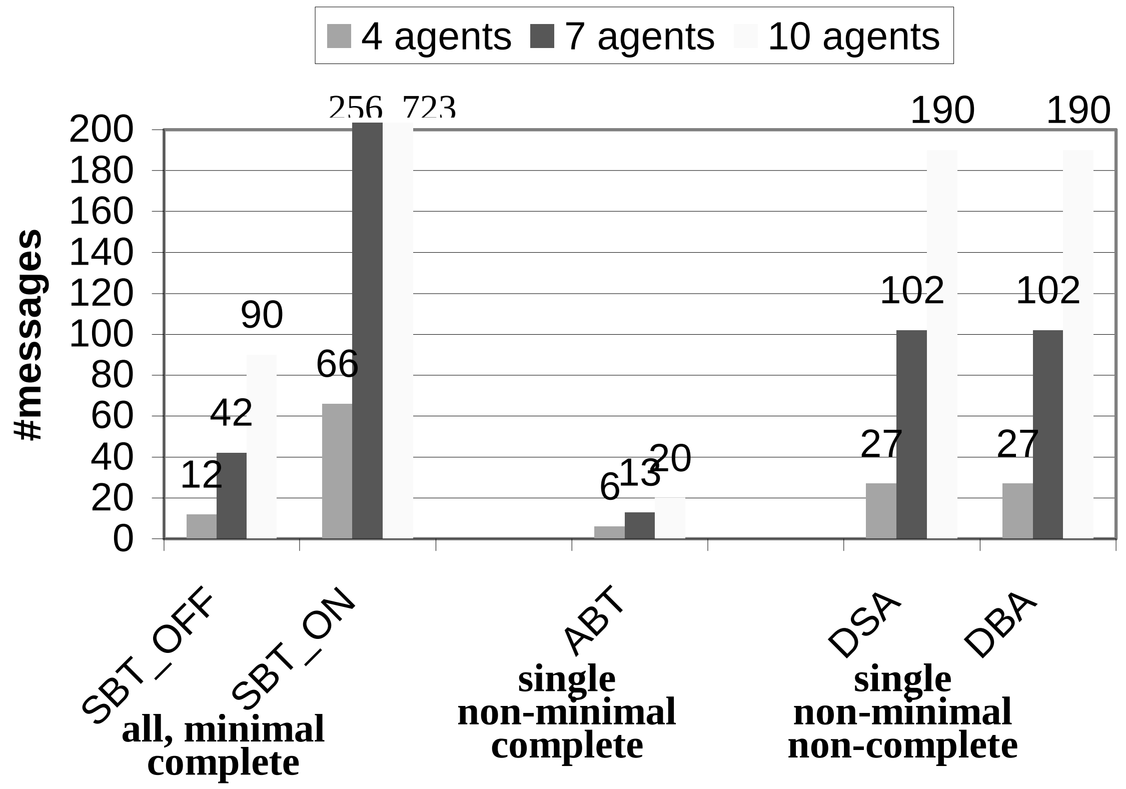


Figure 6.4: Average number of messages in different diagnosis methods.

and so guarantees to find a diagnosis if one exists (although its minimality is not guaranteed).

We examine three factors in diagnosis correctness (Table 6.3): (i) the percentage of robots that failed to find a solution to the DisCSP, even if some of their peers did (here the diagnosis did not completely fail); (ii) the percentage of experiments in which the group failed to compute a diagnosis; and (iii) the percentage of experiments in which the computed diagnosis did not match the correct explanation of the failure(s). Obviously, ABT always succeeds to compute a diagnosis, because it is complete, and therefore the number of failed robots and failures in computing the diagnosis is zero. DSA and DBA are based on local search and are incomplete; some robots failed to compute a diagnosis in 8% of cases in DSA and in 5% of the cases in DBA, and all failed to compute even a single diagnosis in 33% in DSA and in 20% in DBA. Obviously DBA presents better results than DSA since in DSA the agents change their values stochastically in contrast to DBA where we are guaranteed that only one agent changes its value in a conflict between two agents. The three algorithms generate diagnoses that do not match the correct explanation (ABT: 28%, DSA: 46%, DBA: 29%), since they compute only a single diagnosis and

Diagnosis	**% failed robots**	**% diagnosis failures**	**% incorrect diagnosis**
ABT	0	0	28
DSA	8	33	46
DBA	5	20	29

Table 6.3: Diagnosis failures and correctness measures, for ABT, DSA and DBA.

	4 agents	**7 agents**	**10 agents**
# messages	25	30	46
runtime	23	23	23

Table 6.4: DSA and DBA with a threshold of 2 cycles: Number of messages and runtime in cycles. Each data point in DSA is an average of 30 trials.

not a complete set of all the diagnoses.

The results of DSA and DBA are affected by the termination threshold, which determines how long the search runs. To evaluate the effect of this factor, we reran the above experiments for DSA and DBA with a threshold of two cycles instead of a threshold depends on the number of robots in the team. Table 6.4 summarizes the results of the number of messages and runtime (DSA and DBA shows the same results). Comparing these results to the results presented in Figures 6.4 and 6.3, shows a significant improvement especially in terms of constraint checks because the agents operated constraint checks only twice. However, as shown in Table 6.5 compared to running with non-fixed threshold (Table 6.3), diagnosis quality has deteriorated further.

Diagnosis	**%failed robots**	**%diagnosis failures**	**%incorrect diagnosis**
DSA	23	49	56
DBA	12	34	41

Table 6.5: Threshold of 2 Cycles: Average percentage of the recorded failures.

One lesson—expected to some degree—is that there exists trade-off between the effectiveness of the algorithms in terms of communication and computation and the correctness of the diagnosis that the algorithms produce. Algorithms that produce only a single diagnosis cannot always provide the correct diagnosis (ABT: in 28% of experiments, DSA: 46% and DBA: 29%). The correct diagnosis is not predictable and the only way to create it is by generating the whole diagnosis, as in SBT_OFF and SBT_ON. However, as shown in Figure 6.4 and Figure 6.3, SBT_ON is very expensive in terms of communication and computation, and SBT_OFF is applicable only in systems where the constraints are static and defined in advance.

However, there are two surprises. First, ABT outperforms DSA and DBA in terms of constraint checks and communications, in contrast to results in distributed CSP. We believe that this is a general result in the use of ABT for coordination diagnosis, because when there are only few failures at a time, ABT determines in a few steps a close solution to the CSP (and based on it, a diagnosis), compared to the local search behavior of DSA and DBA. Second, ABT outperforms DSA and DBA in terms of the diagnosis results: ABT provides a guarantee to find a diagnosis (DSA and DBA do not), and empirically returns the correct diagnosis much more often than the local search algorithms.

Part II

Design Space for Social Diagnosis Algorithms

In the previous part we formalized social diagnosis for simple reactive and focused on the difference between centralized and distributed approaches. In this part we draw lessons about a design space for social diagnosis algorithms specialized for more complete agents. We examine diagnosis algorithms in small teams and in large-scale teams.

We focus on disagreement faults in teams of situated-agents (behavior-based agents). Because of their frequent use in practice [Tambe, 1997, Matsubara *et al.*, 1998, Pynadath *et al.*, 1999]. Behavior-based agents are a good platform to diagnose disagreement faults in depth, so that we will be able to diagnose the specific disagreements in the behaviors and beliefs of the agents.

To illustrate the problem of diagnosis of disagreements in team of situated-agents, we use an example, originally reported in [Kaminka and Tambe, 2000]. Here, a team of synthetic pilot agents is flying through a virtual battlefield. When reaching a waypoint, they must divide into scouts (who move forward) and attackers (who wait behind). Kaminka and Tambe report on a failure case where the attackers detect the way-point and land, while a scout fails to detect way-point.

A diagnosis system, running on at least one of the agents, must now isolate possible explanations for the disagreement, of which the real cause (agents differ in their belief that the way-point was detected) is but one. Unfortunately, since each agent is potentially unsure of what its team-members are doing, the number of possible explanations can be quite large, which entails (i) significant computation requirements (to compute the hypotheses), and (ii) significant use of communications (to communicate the hypotheses). For instance, for a hypothetical team of eight attackers and eight scouts experiencing a similar failure, we found that 585 messages would have been sent in such a diagnosis process. Furthermore, having each agent simply report its internal beliefs to the others, is also infeasible; the agents cannot simply dump the entire contents of their memory—all their beliefs—to their teammates. Indeed, it has been long recognized that multi-agent systems techniques requiring high bandwidth are not likely to scale in the number of agents [Jennings, 1995, Scerri *et al.*, 2005c].

To address these challenges, in Chapter 7 we present a design space for coordination diagnosis algorithms, distinguishing several phases in the diagnosis process, and providing alternative algorithms for each phase. In Chapter 8 we present novel techniques that enable scalability in three ways. First, we use communications early in the diagnosis process, to stave off unneeded reasoning, which ultimately leads to unneeded communications. Second, we use light-weight (and inaccurate) behavior recognition to focus the diagnostic reasoning on beliefs of agents that might be in conflict. Finally, we propose diagnosing only a limited number of representative agents (instead of all the agents). Chapter 9 presents an experimental evaluation of the basic algorithms (presented in Chapter 7) as well as their scalable versions (presented in Chapter 8) large-scale teams in terms of computation and communication, in different domains, in thousands of trials.

Chapter 7

Social Diagnosis Algorithms for Teams of Situated Agents

In this chapter we seek to draw lessons about the design space of social diagnosis algorithms. We distinguish two phases of social diagnosis: (i) selecting who will carry out the diagnosis; (ii) having the selected agent(s) generate and disambiguate diagnosis hypotheses, where a diagnosis hypothesis is a set of conflicting beliefs, and the agents that disagree about them (i.e., that hold these beliefs). These phases can be distinguished for any social diagnosis process.

To explore these phases concretely, we focus on disagreement faults in teams of situated (behavior-based) agents [Firby, 1987, Newell, 1990, Mataric, 1998, Tambe, 1998]. The control process of such agents is relatively simple to model, and we can therefore focus on the core communications and computational requirements of the diagnosis of disagreement faults. We provide alternative algorithms for these phases, and combine them in different ways, to present six diagnosis methods, corresponding to different design decisions.

This chapter is organized as follows: Section 7.1 presents the architecture of behavior-based agents. In particular, we formalize the notions of belief, behavior, behavior hierarchy, team etc. Based on these definitions we continue to formalize the diagnosis problem by defining disagreement, conflicts and diagnosis. Section 7.2 presents the disambiguating diagnosis hypotheses phase and Section 7.3 presents the diagnosing agent selection phase.

7.1 Building Blocks for Diagnosis of Situated Agents

Behavior-based agents dynamically switch between alternative behaviors (control modules, see Definition 7.1.1 below). Their selection of a controller is done as a result of examining their own internal beliefs, which are influenced by the external world. In such teams we expect to have faults due to the differences between the beliefs of the agents, e.g. because of their different sensing of the external world [Kaminka and Tambe, 1998, Kaminka and Tambe, 2000].

Definition 7.1.1. A *behavior* is a tuple $BHV = \langle VAL, PRE, TER, ACT \rangle$, where VAL is the identifier of the behavior, PRE and TER are sets of logic propositions respectively representing the pre-conditions (which, when satisfied, allow the behavior to be selected), and termination conditions (which terminate its selection if the conditions are satisfied), correspondingly. ACT stands for the actions associated with the behavior, which are executed (possibly in sequence, or repeatedly) once the behavior selected.

We model an agent as having a decomposition hierarchy of behavior nodes organized in an acyclic graph:

Definition 7.1.2. A *behavior hierarchy* is a directed acyclic graph of behaviors $BH = (V, E)$, where V represents the behavior nodes and E represents decomposition relations between the behaviors. An edge $\langle b_1, b_2 \rangle \in E$ denotes that b_2 is a possible decomposition of b_1. We then refer to b_2 as a child of b_1.

At any given time, the agent is controlled by a top-to-bottom path through the hierarchy, root-to-leaf:

Definition 7.1.3. A *behavior path* is a path of behaviors through the hierarchy, root-to-leaf, organized in a set $BP = \{b_1 ... b_h\}$, where b_i represents behavior b in depth (level) i in the hierarchy. Only one behavior in each level of the hierarchy can be part of a behavior path.

Example 7.1.1. *This example is taken from ModSAF, an application involving a virtual battlefield environment with synthetic helicopter pilots. In the example, a team of synthetic pilot agents is divided into two: scouts and attackers. In the beginning all teammates fly in formation looking for a specific way-point (a given position), where the scouts move forward towards the enemy, while the attackers* *land and wait for a signal. Once a signal is sent to the attackers, they take off and fly in formation toward the scouts who await the attackers.*

Figures 7.1 and 7.2 describe portions of the behavior hierarchies of the attacker and scout, respectively. A selection of the path of behaviors {Execute Mission,Wait Point,Fly Route} *is possible by a scout only if the pre-condition of the behaviors in this path:* battle-point scouted=false *is satisfied, and the behavior path will be deselected when the termination-condition* battle-point scouted=true *will be satisfied. Once selecting this behavior path by the scout it will execute the action speed=200. In this example we do not assume a pre-defined sequence of the behavior paths, i.e. any behavior path can potentially be selected at any point. The selection of other behavior path does not depend on the previous behavior path but on the satisfaction of the termination condition of the previous behavior path and the pre-conditions of the new behavior path. In section 7.2.2 we will address pre-defined precedence of behavior paths.*

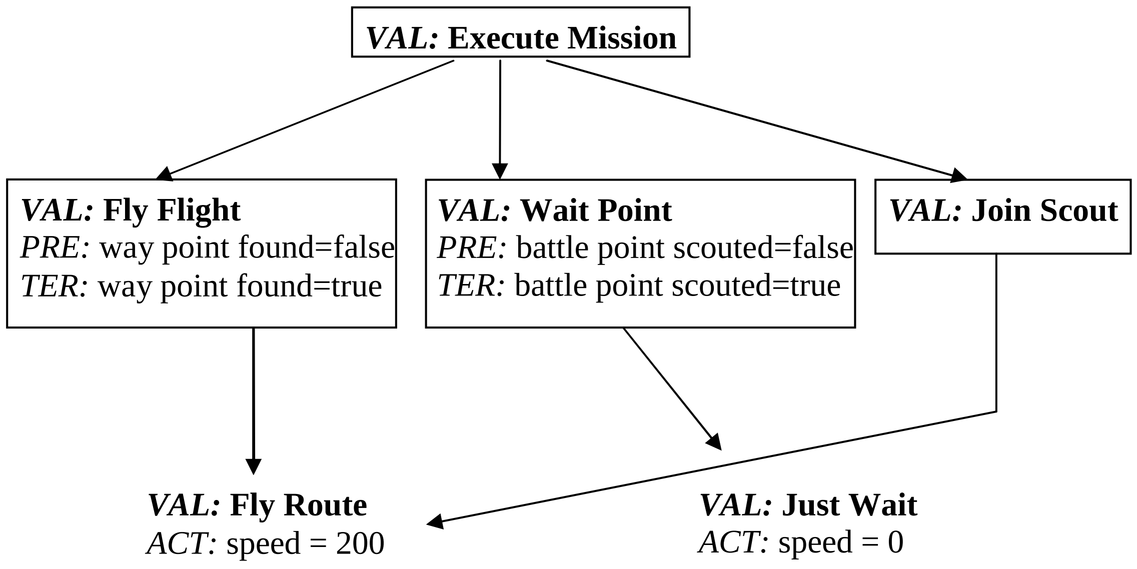
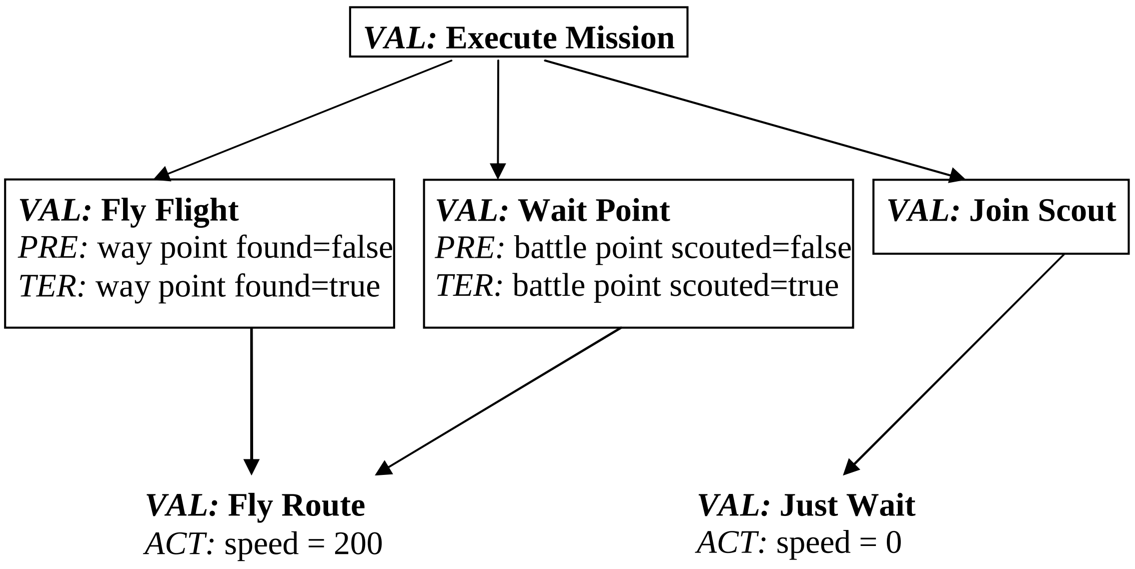


Figure 7.1: attacker's behaviors hierarchy tree (portion).

An agent uses a copy of the behavior hierarchy to track its current selections. Using its sensors it determines its beliefs and selects the behavior path which its pre-conditions are satisfied by its beliefs.

Definition 7.1.4. The *current state* of an agent is a pair $\langle BP, BL\rangle$ where BP represents its selected behavior path and BL the set of its beliefs. A belief is a pair $\langle p, v\rangle$, where p is a proposition and $v \in \{true, false\}$ is its truth value.



VAL: **Execute Mission**
VAL: **Fly Flight**
PRE: way point found=false
TER: way point found=true
VAL: **Wait Point**
PRE: battle point scouted=false
TER: battle point scouted=true
VAL: **Join Scout**
VAL: **Fly Route**
ACT: speed = 200
VAL: **Just Wait**
ACT: speed = 0

Figure 7.2: scout's behavior hierarchy tree (portion).

Example 7.1.2. *In Figure 7.1 the behavior path of an attacker that is waiting after detecting the way point is:* BP={Execute Mission, Wait Point, Just Wait} *and it executes a landing action (speed=0). The beliefs set that lead the attacker to select this behavior path is* BL={battle point scouted=false}.

Definition 7.1.5. A *team* $T = \{a_1...a_n\}$ is a set of n agents, and B is a set of agents' beliefs $B = \{b_1, ..., b_n\}$, where b_i is a set of q beliefs of agent a_i: $b_i = \{b_{i_1}, ..., b_{i_q}\}$.

We follow the convention of agent teamwork architectures, where agents coordinate through the joint selection and deselection of team behaviors, by using communications or other means of synchronization [Jennings, 1995, Tambe, 1997, Kaminka and Frenkel, 2005]. In other words, while each agent executes its own behavior hierarchy, selection of team behaviors within the hierarchy is synchronized. Team behaviors, typically at higher-levels of the hierarchy, serve to synchronize high-level tasks, while at lower-levels of the hierarchy agents select individual (and often different) behaviors which control their execution of their own individual role. Team behaviors are represented by boxes in Figures 7.1 and 7.2.

Definition 7.1.6. A *team behavior* is a behavior which is to be selected and deselected jointly for all the team: $\forall i, j \in T$, $Id_{i_x} = Id_{j_x}$, where T is a team, and Id_{i_x} is the identifier of team behavior node x of behavior hierarchy of agent a_i.

Disagreement between team-members is manifested by selection of different team behaviors, by different agents, at the same time, i.e. by synchronization failures [Kaminka and Tambe, 2000]:

Definition 7.1.7. A *disagreement* exists when the following condition holds: $\exists i, j \in T$, such that $tb \in BP_i \wedge tb \notin BP_j$, where T is a team, tb is a team behavior, and BP_i represents the behavior path of agent a_i.

Example 7.1.3. *Suppose a team of one scout and three attackers* $T = \{S, A_1, A_2, A_3\}$ *(see Figures 7.1 and 7.2 for their behavior hierarchies). A disagreement (coordination fault) occurs if attacker* A_1 *selects to wait* BP={Execute Mission, Wait Point, Just Wait}*, while the scout* S *selects to fly in formation* BP={Execute Mission, Fly Flight, Fly Route}*. The fault occurs due to the selection of* Fly flight *by the scout, in contrast with the selection of the behavior* Wait Point *by attacker* A_1*.*

Disagreements can be detected by socially-attentive monitoring [Kaminka and Tambe, 2000]. In this process all the agents monitor certain key agents using a behavior recognition algorithm. Once a monitor agent cannot find a matching between its own behavior and the behavior of the monitored key agent, it concludes that there is a fault. Since team behaviors are to be jointly selected (as discussed above), such a disagreement can be traced to a difference in the satisfaction of the relevant pre-conditions and termination conditions, e.g., agent A believes P, while agent B believes $\neg P$, causing them to select different behaviors. In the diagnosis process we investigate these conflicting beliefs:

Definition 7.1.8. *Conflicting beliefs* are a pair of two equal belief propositions, of different agents, which have contradictory values. $\langle b_{i_x}, b_{j_y} \rangle$ where $(i \neq j) \wedge (p \in b_{i_x} = p \in b_{j_y}) \wedge (v \in b_{i_x} \neq v \in b_{j_y})$.

It is these conflicting beliefs which the diagnosis process seeks to discover:

Definition 7.1.9. A *diagnosis* for a disagreement is a set of conflicting beliefs $D = \{d_1 ... d_m\}$ that accounts for the disagreement.

Example 7.1.4. *In Example 7.1.3, the belief of scout* S *is the pre-conditions of its behavior (*BP={Execute Mission, Fly Flight, Fly Route}*), e.g.* $\langle$way point found, $false\rangle$*. The beliefs of attacker* A_1 *are the termination conditions of its previous behavior (*BP={Execute Mission, Fly Flight, Fly Route}*) and the pre-conditions of its current behavior (*BP={Execute Mission, Wait Point, Just Wait}*), e.g.* $\langle$way point found*, true*⟩ *and* ⟨battle point scouted*, $false$*⟩*. A diagnosis may be that S believes that the waypoint was not yet found, while A_1 believes that it was, i.e.,* D={way point $found_S$=false*,* way point $found_{A_1}$=true}*.*

7.2 Disambiguating Diagnosis Hypotheses

The design space of diagnosis algorithm is composed of two dimensions: First, the selection of one or more team-members to carry out the diagnosis (in the centralized case, only one, and in the distributed case, all or many); and second, the process by which the selected diagnosing agents disambigurate diagnosis hypotheses to arrive at the correct diagnosis. The algorithms used for selecting the diagnosing agents may depend on the diagnosis process selected in the second phase (disambiguation), and so for clarity of presentation, this section will first discuss the second phase;

The next section (Section 7.3) discusses alternatives for agent selection.

Let us assume for now that one or more agents have been selected to carry out the diagnosis process. The agents must now identify the beliefs of their peers and then find the disagreements. we present two options: (i) the agents report their status to the diagnosing agents (Section 7.2.1); (ii) the diagnosing agents actively query agents as to the state of their beliefs 7.2.2. Obviously, these methods do not exhaust the range of options for diagnoser selection and diagnosis methods. For instance, there are some methods to utilize queries for the diagnosis process. However, we chose these methods since they highlight the extremes of the design space.

7.2.1 Reporting

Perhaps the simplest algorithm for detecting the beliefs of team-members to have all team-members send their relevant beliefs to the diagnosing agent (the diagnosing agent can inform the team-members of the detection of a disagreement to trigger this communication). In order to prevent flooding the diagnosing agent with irrelevant information, each team-member sends only beliefs that are potentially relevant to the diagnosis, i.e., only the beliefs that are associated with its currently selected behavior path.

Upon receiving the relevant beliefs from all agents, the generation of the diagnosis proceeds simply by comparing all beliefs of team-members to find conflicting beliefs (e.g., agent A believes P, while agent B believes $\neg P$). Since the beliefs of the other agents are known with certainty (based on the communications), the resulting diagnosis must be the correct one. However, having all agents send their beliefs may severely impact network load.

The procedure `FIND_CONTRADICTION` (Algorithm 4), gets a set of agents' beliefs B (Definition 7.1.5) and returns a diagnosis D (Definition 7.1.9).

Algorithm 4 FIND_CONTRADICTION
(**input**: set of agents' beliefs B
output: diagnoses set Δ)

```
1: $D \leftarrow \emptyset$
2: for all $b_i \in B$ do
3:     for all $b_j \in B$ where $i < j$ do
4:         for all $b_{i_x} \in b_i$ do
5:             for all $b_{j_y} \in b_j$ do
6:                 compare between the beliefs $b_{i_x}$ and $b_{j_y}$
7:                 if $\langle b_{i_x}, b_{j_y} \rangle$ are conflicting beliefs (Definition 7.1.8) then
8:                     $D = D \cup \langle b_{i_x}, b_{j_y} \rangle$
9: return D
```

In the first line we initialize Δ—a set of diagnosis. In lines 2–3 the diagnosing agent goes over the belief sets of every pair of agents, in order to compare between their beliefs (b_i is the belief set of agent a_i and b_j is the belief set of agent a_j). Then in lines 4–5 the diagnosing agent goes over the beliefs in the set of agent a_i, comparing it to the belief set of agent a_j. In lines 6–8, the diagnosing agent compares between every pair of beliefs. If they have the same proposition but different truth values, it add these conflicting beliefs to the diagnosis set D, associated with their agents.

Let us analyze the runtime and communications complexity of Algorithm 4. Let n denotes the number of agents, m denotes the number of behaviors in a worst-case behavior hierarchy. Let b denotes the maximum number of beliefs per behavior, including both the pre-conditions and termination conditions. The agent is controlled by a top-to-bottom path through the hierarchy, root-to-leaf, where its maximum length is the height of a worst-case (degenerate) tree $O(m)$, and the total number of its beliefs (through the path) is therefore $O(mb)$. The agents send their beliefs to the diagnosing agent. Under the assumption that each belief message is identical in size; the total number of messages in the worst case is equal to the total number of beliefs communicated by the agents, $O(nmb)$ (in the best case the complexity is $O(n \log mb)$ since the high of the behaviors tree is $\log m$). The diagnosing agent compares each agent's beliefs with the others', therefore the runtime complexity in the worst case is $O(n^2m^2b^2)$.

The complexity of this process can be improved by arranging the beliefs in a sorted order. Instead of comparing between the beliefs in double loop (lines 3–4 in Algorithm 4), we could first sort the beliefs (before the loop process) according to the propositions, and then compare between the beliefs, linearly. The sorting process for each agent is $O(mb\log(mb))$ and for n agents is $O(nmb\log(mb))$. Once the beliefs are sorted the complexity of the comparison process is $O(n^2mb)$. So, the total complexity in the worst case is $O(nmb\log(mb) + n^2mb)$. In teams where the number of agents is scaled-up we expect that $n > mb$ so the complexity is $O(n^2bm)$.

Example 7.2.1. *Example (7.1.1, Figures 7.1 and 7.2) assume the scout was chosen to make the diagnosis then the attackers send their beliefs after the transference from* {Execute Mission, Fly Flight, Fly Route} *to* {Execute Mission, Wait Point, Fly Route}*, to the scout. See in Figure 7.1 that the beliefs of A_1 and A_2 are:* {way point found=true $\wedge$ battle point scouted=false}*, a total of four beliefs were sent by communication. The belief of the scout is:* {way point found=false}*. Once the scout has the beliefs of all the agents, it compares between them and finds the contradiction. In our example, the diagnosis is that the attackers' belief is:* {way point found=true}*, in contrast to the scout's belief:* {way point found=false}.

7.2.2 Querying

In the previous algorithm the agents send all the beliefs that are associated with their behaviors. However, some of these beliefs may not be necessary for the diagnosis. We thus propose a novel selective monitoring algorithm, in which the diagnosing agent controls the communications, by initiating targeted queries which are intended to minimize the amount of communications. To do this, the diagnosing agent builds hypotheses as to the possible beliefs held by each agent, and then queries the agents as necessary to disambiguate these hypotheses.

Querying proceeds in three stages (Figure 7.3):

1. **Behavior recognition:** the diagnoser observes its peers and uses a behavior recognition process (see below) to identify their possibly-selected behavior paths, based on their observed actions.

2. **Belief recognition:** based on the hypothesized behavior paths it further hypothesizes the beliefs held by the teammates (which led them to select these behavior paths, by enabling sets of preconditions and termination conditions).

3. **Querying:** the diagnoser queries the diagnosed agents as needed to disambiguate between these belief hypotheses.

Once it knows about the relevant beliefs of each agent, it compares these beliefs to detect contradictory beliefs which explain the disagreement in behavior selection.

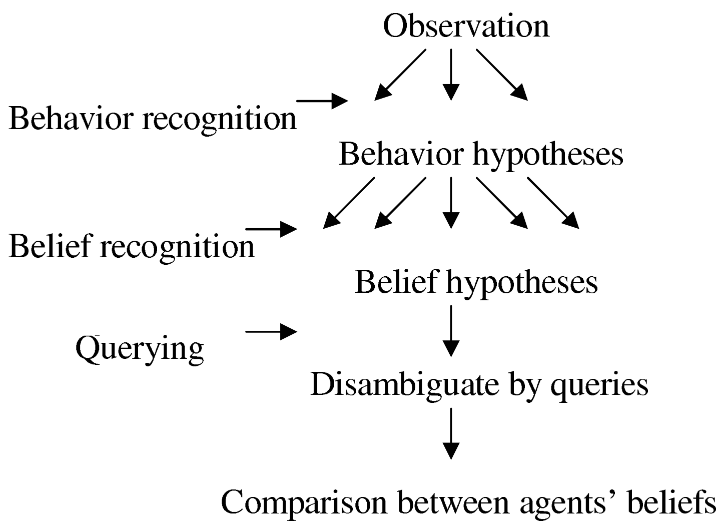


Figure 7.3: Querying process for a single agent.

Behavior Recognition

This process begins with *RESL*, a previously-published behavior recognition algorithm [Kaminka and Tambe, 2000], presented here briefly as a reminder. Under the assumption that each agent has knowledge of all the possible behavior paths available to each team-member, i.e., their behavior path library (an assumption commonly made in plan recognition), each observing agent creates a copy of the fully-expanded behavior hierarchy for each of its teammates. It then matches observed actions with the actions associated with each behavior. If a behavior matches, it is tagged. All tagged behaviors propagate their tags up the tree to their parents (and down to their children) such as to tag entire matching paths: These signify behavior recognition (plan recognition) hypotheses that are consistent with the observed actions of the team-member.

Example 7.2.2. *In Example 7.1.1 (Figures 7.1 and 7.2), a scout and two attackers are teammates carrying out a mission. They had flown in formation (*Fly Flight *behavior) for a while, when the attackers, say A_1 and A_2, landed (*Wait Point *behavior), while the scout continued to fly since it still did not find the way-point and so continued to execute the* Fly Flight *behavior.*

Suppose A_1 monitors A_2 and the scout. It recognizes that the scout's speed is 200, so it can hypothesize (according to the above behavior recognition algorithm) that the scout is executing either: {Execute Mission, Fly Flight, Fly Route} *or* {Execute Mission, Wait Point, Fly Route}*. In addition, A_1 concludes that A_2 is executing* {Execute Mission, Wait Point, Just Wait} *since its speed is 0. A_1 can not detect the fault, since its own behavior matches A_2's behavior and one of the behavior hypotheses of the scout. On the other hand, once the scout monitors A_1 and A_2, it recognizes that their speed is 0, so it concludes that they are executing* {Execute Mission, Wait Point, Just Wait}*, in contrast to its own behavior* {Execute Mission, Fly Flight, Fly Route}*. It can conclude that there is a fault: A_1 and A_2 selected* {Execute Mission, Wait Point, Just Wait}*, while it selected* {Execute Mission, Fly Flight, Fly Route}*.*

The next phase is to identify the reasons for the difference in the selection of the behaviors. The diagnosing agent should disambiguate the correct behavior path of each teammate among its behavior path hypotheses, and the beliefs that account for its selection. These steps are discussed in the next section (Section 7.2.2).

Belief Recognition

Once hypotheses for the selected behavior path of an agent are known to the observer, it may infer the possible beliefs of the observed agent by examining the pre-conditions and the termination conditions of each hypothesized behavior path. To do this, the observer must keep track of the last known behavior path(s) hypothesized to have been selected by the observed agent. As long as the behaviors remain the same, the only general conclusion the observer can make is that the termination conditions for the selected behavior paths have not been met. Thus it can infer that the observed agent currently believes the negation of the termination conditions of selected behavior paths.

When the observer recognizes a transition from one behavior path to another, it may conclude (for the instance in which the transition occurred) that the termination conditions of the previous behavior path, and the pre-conditions of the new behavior path are satisfied. In addition, the termination conditions of the new behavior path must not be satisfied; otherwise this new behavior path would not have been selected.

Therefore, the beliefs of the observed agent (at the moment of the transition) are: *(termination conditions of last behavior path)* $\wedge$ *(pre-conditions of current behavior path)* $\wedge\neg$ *(termination conditions of current behavior path).*

We use V_i^t to denote the set of behavior path hypotheses of agent a_i at time t. We use $PRE(V_{i_j}^t)$ to denote the set of precondition propositions and their truth value. We use $TER(V_{i_j}^t)$ to denote the set of termination propositions and their truth value. F_i^t denotes a set of *belief* hypotheses of agent a_i at time t.

The procedure `BELIEF_RECOGNITION` (Algorithm 5) receives as input the current-time V_i^t (as generated by the behavior recognition process), and the previous behavior path hypothesis set V_i^{t-1} and returns the belief hypotheses set F_i of the same agent (a_i). As mentioned above, the observer has knowledge of the behavior-tree of the observed agents, so the procedure could get as input the behavior path hypotheses of the observed agents.

Algorithm 5 BELIEF_RECOGNITION
(**input**: V_i^t, V_i^{t-1}
output: belief hypotheses set F_i)

```
1: $F_i^t \leftarrow \emptyset$
2: for all $v \in V_i^t$ do
3:   for all $r \in V_i^{t-1}$ do
4:     $F_i \leftarrow F_i \cup TER(r) \cup PRE(v) \cup \neg TER(v)$
5: return $F_i$
```

In the first line the set of the belief hypotheses is initialized as empty set. In line 2–3 the diagnosing agent goes over the current behavior hypotheses against the previous behavior hypotheses. In line 4 it generates the belief hypothesis as a result of the union of the termination conditions of the previous behavior hypothesis and the pre-conditions of the current behavior hypothesis and the termination conditions of the current behavior hypothesis.

Example 7.2.3. *Continuing Example 7.2.2, agent A_1 can infer the beliefs of the other agents as follows. As shown above, the behavior path hypotheses of the scout are:* V_{scout}^t = {{Execute Mission, Fly Flight, Fly Route},{Execute Mission, Wait Point, Fly Route}}. *As a result of belief recognition process we obtain its beliefs hypotheses:* F_{scout} = {{way point found=false},{way point found=true, battle point scouted=false}}. *The first belief hypothesis is derived from the first behavior path hypothesis while the second belief hypothesis is derived from the second behavior* *path hypothesis. In reference to agent A_2, its behavior path hypothesis is V_{A_2} =* {Execute Mission, Wait Point, Just Wait}*, therefore its belief is: F_{scout} =* {way point found=true*,* battle point scouted=false}*.*

Let us analyze the runtime and communication complexity of Algorithm 5. As mentioned above an observer agent infers the beliefs of the other agents by belief recognition process. The runtime complexity of this process depends on the number of agent's beliefs in a single path, the number of behavior path hypotheses, and the number of agents:

1. **The number of agent's beliefs in a single path.** The number of agent's beliefs per behavior path in the worst case has already been shown to be $O(bm)$, where m is the number of behaviors and b is the number of beliefs per behavior. But through belief recognition we combine the termination conditions of the previous behavior path with the pre-conditions and termination conditions of the current behavior path thus the number of beliefs is $O(2bm) = O(bm)$. Each belief proposition may be true or false, therefore the number of possible belief combinations per behavior path in the worst case is $O(2^{2bm})$.

2. **The number of behavior path hypotheses.** Suppose r denotes the number of behavior path hypotheses in the behavior hierarchy k-ary tree, (where k designates the branching factor, i.e., the number of children of each behavior). Then the number of possible paths is limited by the number of leaves. The number of leaves is at most $(m - m/k)$. It is likely that $r \ll m - m/k$ since only a few of the path possibilities of the tree are indeed possible paths for a certain recognized behavior.

3. **The number of agents.** This process is repeated for each observed agent so the runtime complexity in the worst case is: $O(nr2^{2bm})$.

The belief recognition process does not involve any communication so we do not present the communication complexity for this process.

Targeting Disambiguation Queries

Once belief hypotheses are known, the agent can send targeted queries to specific agents in order to disambiguate the hypotheses. The queries are selected in a manner that minimizes the expected number of queries. Intuitively, the agent prefers to ask first about propositions whose value, when known with certainty, will approximately split the hypotheses space.

For instance, suppose there are four hypotheses: $H1 = \{a, b, c, \neg h\}$, $H2 = \{a, b, d, \neg k\}$, $H3 = \{a, e, \neg m\}$, $H4 = \{f, g, \neg p\}$. a occurs in three of the four hypotheses, therefore if the value of a is queried and the response is $a = true$, then the three hypotheses that contain a are still active. On the other hand, b appears only in two of the hypotheses, and so it splits the hypotheses space. If $b = true$ then hypotheses $H1$ and $H2$ are active, and if $b = false$ the two other hypotheses are active. The other beliefs have one occurrence, therefore, like a, they divide the space to two unequal parts. In the best case only one hypothesis will be active, but in the worst case three hypotheses will be active.

Let us analyze the minimal number of queries necessary to disambiguate the hypotheses. A brute-force approach would have us evaluate the consequences of any sequence of queries to determine the optimal number of queries, but the computational complexity of this procedure is combinatorial in the number of beliefs.

Instead, we use a greedy one-step look ahead strategy based on entropy, similarly to its use in the "twenty questions" problem. The entropy function is taken from information theory [Shannon, 1948]:

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

$Entropy(S)$ calculates the entropy of belief S. c represents the number of values of belief S, and p_i is the proportion of S belonging to value i. The entropy function varies between 0 and $\log_2 c$. The entropy is close to the minimum (0), when the distribution of the values of belief S is not uniform. The more the entropy is close to the maximum, the more the distribution is uniform.

In our case each belief proposition has three possible values: $true$, $false$, $don't\ care$, and the maximal entropy is $\log_2 3 = 1.58$ (when the hypotheses space is distributed uniformly by a belief query). In each step we want to query as to the belief whose value will split the hypotheses space as uniformly as possible to different classes. Then, every remaining hypothesis is equally likely, which means that the next query is expected to leave only 1/3 of the hypotheses. In theory, if all queries equally divide remaining hypotheses to three groups, there will be only $O(\log x)$ queries, where x is the number of belief hypotheses. In the worst case, a total of $x - 1$ queries would be necessary, and in the best case, only one.

Example 7.2.4. *In Example 7.2.3, when agent A_1 models the others it recognizes* *that A_2 has only a single belief hypothesis, so its beliefs are known to A_1 without any query. The scout has two belief hypotheses, therefore only one query is required to disambiguate between them. The two hypotheses are:* {way point found=false} *and* {way point found=true, battle point scouted=false}*. The probability of* way point found=false *as well as of* way point found=true *is* 0.5 *since they occur one time in two hypotheses, the probability of* way point found=don't care *is* 0*, therefore,* E(way point found) $= -(-(0.5 \log_2 0.5) - (0.5 \log_2 0.5) - (0 \log_2 0)) = 1$*. The probability of* battle point scouted=false *is* 0.5 *as well as the probability of* battle point scouted=don't care *(since in the first hypothesis this belief does not appear), and the probability of* battle point scouted=true*is* 0*. Therefore,* E(battle point scouted) $= -(-(0.5 \log_2 0.5) - (0.5 \log_2 0.5) - (0 \log_2 0)) = 1$*. Both of the belief queries have the same entropy and therefore one of them is selected as a query to the scout arbitrarily. Assume* way point found *was selected and the response of the scout is* way point found=true*, then A_1 can conclude that the correct hypothesis of the scout is* {way point found=true $\wedge$ battle point scouted=false}*. Now it can find the diagnosis by comparing between the beliefs.*

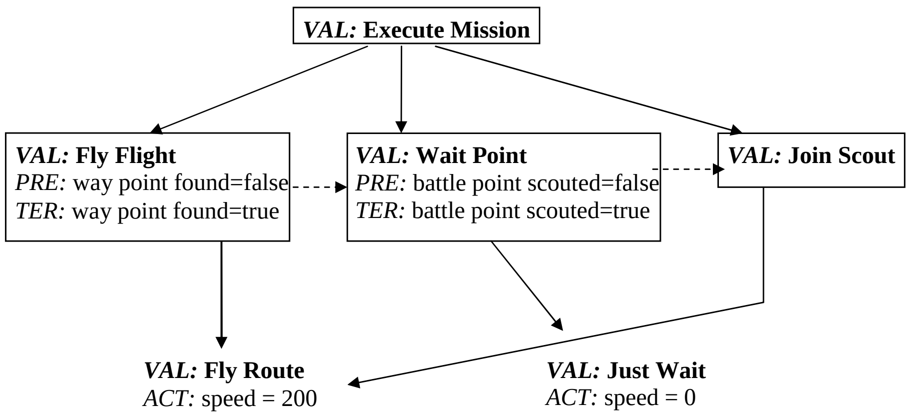
Once the belief hypotheses were disambiguated by querying, the diagnosing agent should compare between the beliefs of the agents. So we should add the runtime complexity of the comparisons between the beliefs as computed in section 7.2.1: $O((nbm)^2)$. Thus overall runtime complexity of the querying algorithm in the worst case is: $O(nr2^{2bm} + (nbm)^2)$.

The communication complexity is influenced by sending targeted queries to specific agents in order to disambiguate their hypotheses. As described above the worst-case complexity of the number of queries to one observed agent is the number of beliefs, $O(bm)$. The queries are sent to all the observed agents, so the messages transmission complexity in the worst case is: $O(nbm)$.

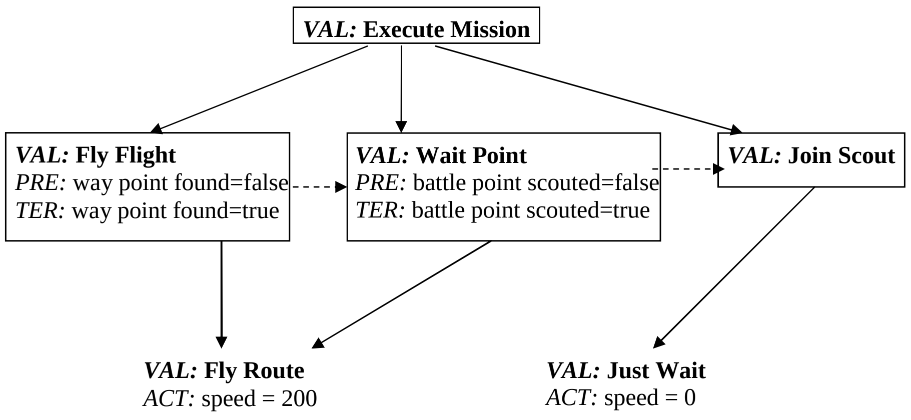
This complexity is similar to that of reporting algorithm (Section 7.2.1) where each agent sends its beliefs to the diagnosing agent ($O(nbm)$). But while in the reporting algorithm this complexity is $O(nb \log m)$ in the best case, here the average case can be expected to have a reduced number of messages, and in the best case it could be even one message.

Precedence Between Behaviors

The analysis above assumes any behavior can potentially be selected at any point. However, in some domains there may be known temporal orderings between behaviors. These eliminate hypotheses from being generated, if they do not agree with the temporal order in which behaviors can be selected.



(a) *Attacker*



(b) *Scout*

Figure 7.4: Agent behavior hierarchies, with ordering information.

Example 7.2.5. *In the ModSAF domain (see Example 7.1.1) suppose the permissible transitions are from* Fly Flight *to* Wait Point *and then to* Join Scout*, as seen in Figure 7.4—a and 7.4—b (the dashed lines represent possible transitions). Let us examine the following case: An attacker and a scout fly in formation in* Fly Flight *behavior, when a fault is detected. Suppose the scout makes the diagnosis. The attacker's speed is 200, so the scout can conclude, according to the behavior recognition process, that observed behavior paths are either* {Execute Mission, Fly Flight, Fly Route} *or* {Execute Mission, Join Scout, Fly Route}*. However, the transition from* Fly Flight *to* Join Scout *is impossible because the attacker could not have gone from* Fly Flight *to* Join Scout *directly without passing through* Wait Point*. So, with certainty it concludes that attacker's behavior path is* {Execute Mission, Fly Flight, Fly

Route}. *Obviously, it is better to disambiguate the beliefs of the attacker when we have only one behavior path hypothesis.*

To summarize, we presented two algorithms for generating and disambiguating social diagnosis hypotheses: *reporting*, in which all the agents send their beliefs that are associated with their selected behavior paths to the diagnosing agent, and *querying*, in which the diagnosing agent models the others by using the belief recognition process, and disambiguate their beliefs by querying them about certain beliefs. In the next section we will examine the question of who makes the diagnosis.

7.3 Selecting a Diagnosing Agent

Let us now turn to the first phase of social diagnosis, in which the agents that will carry out the diagnosis are selected. Several techniques are available. First, a design-time selection of one of the agents is the most trivial approach. Since the pre-selected agents do not necessarily know when a failure is detected (a different agent may have detected the failure), a failure state must be declared by the agents that have detected the failure, and communicated to the pre-selected agent, such that the pre-selected agents know to begin their task. A second technique that circumvents this need is to leave the diagnosis in the hands of those agents that have detected the failure, and allow them to proceed with the diagnosis without necessarily alerting the others unless absolutely necessary.

We present a third approach, in which selection of the diagnosing agent is based on its team-members' estimate of the number of queries that it will send out in order to arrive at a diagnosis, i.e., the number of queries that it will send out in the disambiguation phase of the diagnosis (previous section). The key to this approach is for each agent to essentially simulate its own reasoning in the second phase, as well as that of its teammates. Agents can then jointly select the agent with the best simulated results (i.e., the minimal number of queries).

Surprisingly, all agents can make the same selection without communicating, using a recursive modeling technique in which each agent models itself through its model of its teammates. This proceeds as follows. First, each agent uses the belief recognition algorithm to generate the belief hypotheses space for each team-member other than itself. To determine its own hypothesis space (as it appears to its peers), each agent uses recursive modeling, putting itself in the position of each one of its teammates and running the belief recognition process described above with respect to itself. Under the assumption that all agents utilize the same algorithm, and have access to the same observations, an agent's recursive model will yield the same results as the modeling process of its peers. At this point, each team-member can determine the agent with the minimal number of expected queries by using the strategy discussed in Section 7.2.2. In order to guarantee an agreement on the selected agent, each team-member has an ID number, which is determined and known in advance. In case there are two agents or more with the same minimal number of expected queries, the agent with the minimal ID is selected. This entire process is carried out strictly based on team-members' observations of one another, with no communications other than an announcement of a disagreement.

As mentioned, this algorithm assumes that all agents have access to the same observations. We can obtain the same observation for all the agents if we assume full observation in regard to particular sensors. Full observation could be possible especially in small teams and/or small physic area. But also in large areas there are sensors which provide full observation. For instance, a radar has a range of some kilometers, which enables determine the altitude and speed of a strange objects. In the ModSAF domain, in which we evaluated our algorithms, this assumption is true and indeed previous works utilized this assumption in order to make fault detection and plan recognition in the ModSAF domain [Kaminka and Tambe, 2000]. Obviously in domain where an agent has only partial observation, we are not guaranteed to obtain the same observation for all the agents and the agents may disagree on the the selection of the diagnosing agent.

The procedure `MINIMAL_QUERIES_DIAGNOSING_AGENT` (Algorithm 6) gets the behavior path hypotheses of all the agents in a team T: $V^t = \{V_i^t\}$ (that is obtained by behavior recognition process), and their previous behavior path $V^{t-1} = \{V_i^{t-1}\}$. For each observed agent it calls to `BELIEF_RECOGNITION` algorithm which returns the belief hypotheses set of the observed agent (lines 1–2). For each belief hypotheses the algorithm calculates its estimated number of queries N_i (line 5), and then returns the agent that has the minimal number of queries.

For instance, suppose there are three agents A, B and C. To determine the diagnosing agent, A models itself from B's perspective and considers the belief hypotheses that B has about A and C, given A's and C's observable actions. A also uses the belief recognition process described earlier to determine the number of belief hypotheses available about B's beliefs, C's beliefs, etc. It now simulates selecting queries by each agent, and selects the agent (say, C) with the minimal number of ex-

Algorithm 6 ESTIMATED_OPTIMAL
(**input**: V^t, V^{t-1}
output: agent a_i)

```
1: for all Agents $i$ do
2:     $F_i \leftarrow$ BELIEF_RECOGNITION($V_i^t$,$V_i^{t-1}$)
3:     $F \leftarrow F \bigcup F_i$
4: for all $F_i \in F$ do
5:     $N_i \leftarrow$ number of queries based on maximal information gain
6:     $N \leftarrow N \bigcup N_i$
7: return $\{a_i \in T | min(N_i)\}$
```

pected queries. B, and C also run the same process, and under the assumption that each agent's actions are equally observable to all, will arrive at the same conclusion.

Example 7.3.1. *In Example 7.2.3, the scout models the attackers, and models the attackers modeling the scout (itself). The belief hypotheses of each one of the attackers are:* {way point found=true ∧ battle point scouted=false}*, so no query is requested. The result of belief recognition (by recursive modeling) of the scout on itself is:* {*way point found=false*} *or* {way point found=true ∧ battle point scouted=false}*, only a single query is needed. These process is carried out by the attackers too, and they into the same results. Obviously, in this case the scout is selected to diagnose, since it is expected to send no queries, while if one of the attackers would be selected it would send one query.*

Chapter 8

Social Diagnosis: Scaling-Up

In this chapter we seek to enable social diagnosis in large-scale teams of behavior-based agents. We first develop techniques which use communications earlier in the diagnosis process (compared to previous work), in an attempt to stave off both the runtime associated with generation of diagnostic hypotheses, as well as later communications. These techniques include: (i) using initial queries to alleviate diagnostic reasoning (BEHAVIOR QUERYING); (ii) using communications in light-weight behavior recognition to focus on relevant beliefs that may be in conflict (SHARED BELIEFS).

These "communicate early" techniques enable a third method (GROUPING) in which the diagnosed agents are divided into groups based on their selected behavior and their role, such that all members of a group are in agreement, and at least one disagreement exists between any two groups. Then, only representative agents of each group are diagnosed, and the results are used for others in their group. By using grouping, we limit the required communication and computation which is done only among the representative agents, and thus make the approach applicable to large teams.

This chapter is organized as follows: We suggest three methods that tackle the runtime and communication complexities. In Section 8.1 we present the BEHAVIOR QUERYING technique which eliminates the behavior recognition process by querying about the selected behavior path. In Section 8.2 we present the technique SHARED BELIEFS which limits the belief recognition process by inferring only the propositions of the beliefs, not their value. finally in Section 8.3 we present the GROUPING technique which reduces the number of diagnosed agents by grouping together agents along disagreement lines, and selecting representative agents for diagnosis.

8.1 Behavior Querying

Generally a behavior is associated with several beliefs through its preconditions and termination conditions. Thus, each behavior path hypothesis may generate several belief hypotheses as previously described. Therefore, we expect the number of belief hypotheses to grow with the number of behavior path hypotheses.

The behavior recognition process is responsible for the growing number of behavior path hypotheses. Using this process, the diagnosing agent infers the hypothesized behavior paths associated with the action of the observed agents. We can eliminate the uncertainty in the behavior recognition process by disambiguating the observed agent's behavior path using communication, instead of inferring all its behavior path hypotheses. This goal is achieved by committing early in the diagnosis process to using communications, querying the observed agent about its behavior path. Once the diagnosing agent knows the behavior path of the monitored agent, it continues to build the belief hypotheses that are associated only with that behavior path. The advantage of this method is that by a single query about the behavior path of the observed agent, it eliminates all the queries about the belief hypotheses associated with other (incorrect) behavior path hypotheses.

We predict an improvement in terms of runtime since the BEHAVIOR QUERYING method eliminates the belief hypotheses computation of all the behavior path hypotheses except for the correct one. So instead of the linear complexity of behavior recognition (in the number of behaviors in the behavior hierarchy), the number of behaviors has no effect at all, and the resulting complexity is $O(1)$. We additionally predict an improvement in communications, since using communication early in the diagnosis process eliminates a large number of belief hypotheses, and so saves up communication that would have been done lately by disambiguating the correct beliefs by querying.

8.2 Shared Beliefs

The main factor that causes a high runtime of the QUERYING algorithm is the use of belief recognition process. For each diagnosed agent, runtime for this process grows *exponentially* in the number of beliefs associated with hypothesized recognized behavior paths. Even if the number of behavior path hypotheses is one, belief recognition will typically have multiple beliefs associated with it, and thus result in an exponential number of belief hypotheses.

The exponential complexity is affected by taking into consideration all the combinations between the possible values of every belief proposition $(true, false)$ that belong to the pre-condition and termination condition of the behavior path hypothesis. As we showed in Section 7.2.2 the number of agent's beliefs per behavior path in the worst case is $O(bm)$, where m is the number of behaviors and b is the number of beliefs per behavior. But through belief recognition we combine the termination conditions of the previous behavior path with the pre-conditions and termination conditions of the current behavior path thus the number of beliefs is $O(2bm) = O(bm)$. Each belief proposition may be true or false, therefore the number of possible belief combinations per behavior path is $O(2^{2bm})$.

We present a light-weight belief recognition technique whose complexity grows *polynomially* with the number of beliefs. The key to this technique is to infer only the propositions associated with a belief, without hypothesizing their values. In other words, the key is to infer that an agent has beliefs about p, without inferring what these beliefs are (i.e., whether the agent believes p or $\neg p$ is true). The diagnosing agent uses this technique to infer, for each agent, what propositions it holds. Then, for each pair of agents it queries for the values of propositions that are shared by the agent, and may thus be in conflict. The diagnosis is the union of the SHARED BELIEFS that were found in conflict.

We use V^t to denote the set of behavior path hypotheses of the agents in team T at time t (V_i^t denotes the set of behavior path hypotheses of agent a_i). We use $PRE(x)$ to denote the set of precondition belief propositions, and $TER(x)$ to denote the set of termination belief propositions, where $x \in V_i^t$, i.e., x is a path through the hierarchy.

The procedure `SHARED_BELIEFS` (Algorithm 7) receives as input the current-time V^t (as generated by the behavior recognition process), and the previous behavior path hypotheses set V^{t-1} and returns the diagnosis D.

In lines 1–2 we initialize Δ—a set of diagnosis and SB—a set of shared beliefs. In lines 3–4 the diagnosing agent goes over the behavior path hypothesis sets of every couple of agents, in order to compare between their behavior paths (V_i^t is the behavior path sets of agent a_i and V_j^t is the behavior path sets of agent a_j). Then in lines 5–6 the diagnosing agent goes over every two certain behavior paths of agents a_i and a_j, in order to compare between their associated beliefs.

In lines 7–8 the algorithm unions the belief propositions that are associated with the behavior path of the observed agents, separately for each agent. The associated

Algorithm 7 SHARED_BELIEFS
(**input**: V^t, V^{t-1}
output: diagnoses set Δ)

```
1: $SB \leftarrow \emptyset$
2: $D \leftarrow \emptyset$
3: for all $V_i^t \in V^t$ do
4:     for all $V_j^t \in V^t$ where $i \neq j$ do
5:         for all $x \in V_i^t$ do
6:             for all $y \in V_j^t$ do
7:                 $F_x \leftarrow PRE(x) \cup TER(x) \cup TER(V_i^{t-1})$
8:                 $F_y \leftarrow PRE(y) \cup TER(y) \cup TER(V_j^{t-1})$
9:                 $SB = F_x \cap F_y$
10:                for all $SB_i \in SB$ do
11:                    query agents $a_i$ and $a_j$ about the values of the proposition $SB_i$
12:                    if the $values$ of $SB_i$ are opposite then
13:                        $D = D \cup SB_i$
14: return $D$
```

belief propositions of agent a_i executing a behavior path V_i^t are (i) the preconditions ($PRE(V_i^t)$); (ii) termination conditions ($TER(V_i^t)$); and (iii) the termination condition of its previous behavior path $TER(V_i^{t-1})$ (since it terminated this behavior path). Then in line 9 the diagnosing agent finds the shared beliefs of the observed agents by intersecting between their belief propositions. Finally, in lines 10–13 it goes over through the SHARED BELIEFS and for each one of them it disambiguates its value by querying. If its value was found in conflict between the agents, it is added to the diagnosis set D.

Example 8.2.1. *Assume the preconditions and the termination conditions of the current behavior of agent A consider propositions p and q and its termination conditions of previous behavior consider propositions r. Those of agent B consider p and s by the preconditions and termination conditions of current behavior, and proposition r by termination conditions of previous behavior. p and r are the propositions shared by agent A and agent B. To determine whether A and B disagree, the diagnosing agent only needs to send queries about the value of p and r to agents A and B, since they are the only proposition relevant to both. For instance, a possible diagnosis is that agent A believes p while agent B believes $\neg p$ (assuming they agree on r).*

Assume n denotes the number of agents and r denotes the average number of behavior path hypotheses per agent, then the complexity of the algorithm is $O(n^2r^2b^2m^2)$ (bm is the complexity of the number of beliefs b per behavior path m). On the other hand, the complexity of QUERYING algorithm which uses belief recognition is $O(nrm^{2b} + n^2b^2m^2)$ (nrm^{2b} is the complexity of belief recognition for r behavior path hypotheses of n agents, $n^2b^2m^2$ is the complexity of the comparison between the beliefs of the agents). The main difference between the algorithms is that SHARED BELIEFS is polynomial in the number of beliefs while belief recognition is exponential in the number of beliefs. However, in a small number of beliefs the factor of the comparison between the agents is more significant.

Using this method, we expect that communications will grow with the number of agents, relative to the QUERYING algorithm since in teams we expect that most of the beliefs will be SHARED BELIEFS, so most of them are suspected.

8.3 Grouping

Regardless of how knowledge of the beliefs of teammates is inferred, the diagnosing agent must compare between the beliefs of the teammates after inferring those beliefs. This comparison is polynomial in the number of agents and in the number of beliefs. However, in a large-scale team, runtime may be too high in practice.

The GROUPING method abstracts the observed agents, grouping together agents that are in a similar state. It then uses a single agent from each group as a representative for all agents in its group. To determine the diagnosis, it only compares the beliefs of these representative agents, significantly reducing the total number of comparisons.

The process is based on the assumption that two or more agents that have both the same role in the team and the same behavior path will have the same beliefs, at least with respect to their selection of role and behavior path. Based on this assumption only representative agents of each role and behavior path must be diagnosed.

The GROUPING method thus relies on BEHAVIOR QUERYING (Section 8.1) and SHARED BELIEFS (Section 8.2). To determine the different role/behavior path combinations, the diagnosing agent first disambiguates the behavior path of each monitored agent using the BEHAVIOR QUERYING process. It then divides the team to groups based on their roles and behavior paths; this essentially divides the team along disagreement lines. It continues with the diagnosis process only for representative agents of each group (hereinafter: *representative agents*) by the SHARED BELIEFS method (actually it can continue also by QUERYING algorithm, but in Section 9.4.2 we will show that the SHARED BELIEFS technique is better). Finally, it uses the results of the diagnosis for the remaining members of the groups.

Example 8.3.1. *Assume a team of seven agents $T = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$, a_1, a_2 and a_3 have the role r_1, a_4, a_5 ,a_6 and a_7 have the role r_2. Using* BEHAVIOR QUERYING*, the diagnosing agent disambiguates the current behavior path of the observed agents. After this process, it finds that a_1, a_2, a_3, a_4 and a_5, are in behavior v_1 while a_6 and a_7 are in behavior v_2. By grouping the team according to their selected behavior paths and their roles, the diagnosing agent identifies three sub-groups: $T_1 = \{a_1, a_2, a_3\}$ (role r_1 and behavior path v_1); $T_2 = \{a_4, a_5\}$ (role r_2 and behavior path v_1); and $T_3 = \{a_6, a_7\}$ (r_2 and behavior path v_2).*

Now, the diagnosing agent continues the diagnosis, only considering the representative agents a_1, a_4 and a_6 (selected arbitrarily) using the SHARED BELIEFS algorithm. It finds the conflicts represented in the diagnosis set $D = \{\langle a_1 : b_1, a_4 : \neg b_1\rangle, \langle a_1 : b_2, A_6 : \neg b_2\rangle\}$. The diagnosing agent generalizes this diagnosis to the other members of the sub-groups $D = \{\langle a_1, a_2, a_3 : b_1, a_4, a_5 : \neg b_1\rangle, \langle a_1, a_2, a_3 : b_2, A_6, a_7 : \neg b_2\rangle\}$.

We hypothesize that this process will reduce both the number of messages as well as runtime. The diagnosis process would involve a significantly-reduced number of agents, as only the group representatives are diagnosed. This number is bounded from above, by the product of the number of roles in the team and the number of behavior paths. Assuming s denotes the number of roles and p denotes the number of behavior paths, then $s * p \ll n$ and so the whole diagnosis process in large-scale teams will be significantly faster. Communications will still grow in the number of agents, since the diagnosing agent has to disambiguate the behavior path of the agents by BEHAVIOR QUERYING in order to divide the team to groups, but again it will grow much slowly in the number of agents related to the QUERYING or REPORTING algorithms.

A potential disadvantage of this method lies with its assumption that agents in the same group will have the same beliefs, an assumption which may not always be correct. For instance, if the termination condition of a behavior Z is $p \vee q$, then an agent A may terminate this behavior because it believes that p is true (q is false), while an agent B which has the same role as A, may terminate the same behavior because it believes that q is true (and p is false). Both of the agents will then terminate Z, and may select the same new behavior, although their beliefs are not the same. However, we believe that this case is rare. It did not occur in our experiments.

Chapter 9

Team of Situated Agents: Evaluation and Discussion

In this chapter we empirically evaluate the algorithms presented in Chapter 7 and in Chapter 8 in diagnosing thousands of systematically—generated failure cases, occurring in a team of behavior-based agents in two different complex domains.

In section 9.1 we present six algorithms derived from the design space of social diagnosis presented in Chapter 7. In section 9.2 we draw general lessons about the design of social diagnosis algorithms from the empirical results. Specifically, the results show that centralizing the disambiguation process is a key factor in dramatically improving communications efficiency, but is not a determining factor in runtime efficiency. On the other hand, explicit reasoning about other agents is a key factor in determining runtime: Agents that reason explicitly about others incur significant computational costs, though they are sometimes able to reduce the amount of communications.

In section 9.3 we present four algorithms based on the techniques presented in Chapter 8. In section 9.4 we empirically examine these algorithms. We find that BEHAVIOR QUERYING reduces both runtime and communications. However, the SHARED BELIEFS technique does not scale well. Moreover, when combined, these techniques do not reduce communications nor runtime. Surprisingly, however, the GROUPING method (which is enabled by this disappointing combination), results in a diagnosis process which is highly scalable in both communication and computation.

9.1 Social Diagnosis Methods for Teams of Situated Agents

In Chapter 7 we presented a space of social diagnosis algorithms: Each algorithm operates in two phases, and we presented alternative techniques for each phase. For the selection of the diagnosing agent, we have the following methods: (i) rely on pre-selection by the designer; (ii) let the agents that detected the fault do the diagnosis; or (iii) choose the agent most likely to reduce communications (using the distributed recursive modeling technique described in Section 7.3). In terms of computing the diagnosis, two choices are available: Either have all agents communicate their beliefs to the selected agents (Section 7.2.1), or allow the diagnosing agents to actively query agents as to the state of their beliefs, while minimizing the number of queries as described above (Section 7.2.2).

These design alternatives define a space of diagnosis methods, corresponding to different combinations of the alternative algorithms in each phase. These are described in detail below. Table 9.1 summarizes the diagnosis possible methods in this space. The first column represents the algorithms of the selection of the diagnosing agent. The first row presents the algorithms of computing the diagnosis. The other cells in the table present the relevant methods according to the diagnosis space in the column and the row.

	query	**report**
pre-selected	method 5	method 1,method 3
detectors	method 2	N/A
minimal queries	method 4	method 6

Table 9.1: Summary of the diagnosis methods in the design space.

Method 1. The first design choice corresponds to arguably the most trivial diagnosis method, in which **all** agents are pre-selected to carry out the diagnosis. When a failure is detected (and is made known to all agents) each agent communicates all its relevant beliefs to the others so that each and every team-member has a copy of all beliefs, and therefore can do the diagnosis itself.

Method 2. Arguably, only a single agent really needs to have the final diagnosis in order to begin a recovery process. Thus in method 2, the agents that detected the disagreement automatically take it upon themselves to carry out the diagnosis, unbeknownst to each other, and their teammates (who did not detect the disagreement). Because their teammates may not know of the disagreement, the diagnosing agents cannot rely on their teammates to report their beliefs without being queried (in phase 2). Instead, they use the querying algorithm discussed in the previous section. Thus even other diagnosing agents will be queried.

Method 3. The next design choice corresponds to a diagnosis method in which the designer pre-selects one of the agents, arbitrarily. When a failure is discovered (and is made known to all agents), all team-members communicate immediately all their relevant beliefs to this pre-selected agent. While in method 1 all the agents make the diagnosis and report their beliefs each other, here only a single pre-defined agent makes the diagnosis.

Method 4. The fourth method attempts to reduce the communications. It uses the recursive modeling technique to have all team-members agree on which agent is to carry out the diagnosis (this requires the detection of the disagreement to be made known). Once the agent is selected (with no communications), it queries its teammates as needed.

Method 5. In this method the diagnosing agent is selected in advance by the designer, in contrast to method 4. It uses the querying method in order to make the diagnosis.

Method 6. This method uses the selection of the most likely agent to reduce the communication as basis. However, once this agent is selected, the other agents do not wait for its queries, and instead report to it.

In principle, there is one more method in which the beliefs of all the agents are reported to the fault-detecting agents, which make the diagnosis (the empty box in Table 9.1). But, we did not experiment with this algorithm, since we already examine methods where the agents report their beliefs to the diagnosing agent(s): in method 1 they report to all the other agents (all agents are pre-selected), and in method 3 they report to a single pre-selected agent. A method where all agents report to some diagnosing agents (here, fault detecting) will be bounded from above and from below by methods 1 and 3.

Table 9.2 summarizes the phases and the worst-case complexity of the different methods. Each method is presented in a different row. The first column shows the method (by #). The next two columns correspond to the different phases of the diagnosis process. The choice of algorithm is presented in each entry, along with a marking that signifies the number of agents that execute the selected technique for the phase in question. The last two columns present the worst-case complexity of

#	Selection	Disambiguation	Runtime	Communication
1	pre-selected (N)	$N \rightarrow N$ reporting	$O((nbm)^2)$	$O(n^2bm)$
2	detectors ($K \leq N$)	$K \rightarrow N$ querying	$O(nrm^{2b} + (nbm)^2)$	$O(n^2bm)$
3	pre-selected (1)	$N \rightarrow 1$ reporting	$O((nbm)^2)$	$O(nbm)$
4	minimal queries (1)	$1 \rightarrow N$ querying	$O(nrm^{2b} + (nbm)^2)$	$O(nbm)$
5	pre-selected (1)	$1 \rightarrow N$ querying	$O(nrm^{2b} + (nbm)^2)$	$O(nbm)$
6	minimal queries (1)	$N \rightarrow 1$ reporting	$O(nrm^{2b} + (nbm)^2)$	$O(nbm)$

Table 9.2: Summary of evaluated diagnosis methods and their runtime and communication worst-case complexity.

the runtime and communication, correspondingly.

For instance row 2 should be read as follows: In method 2, the agents selected to perform the disambiguation are those who detected the disagreement. K such agents exist (where K is smaller or equal than the total number of agents in the team, N), and they each execute the querying algorithm, such that K agents query N agents. In contrast, row 3 indicates that a single pre-selected agent executes the diagnosis, and it relies on reports from all agents to carry out the diagnosis, such that N agents report their beliefs to 1 agent.

We can see that the communication complexity of methods 1 and 2 has an n^2 factor, in contrast to the other methods, since the diagnosis is carried out by several agents (up to n).

The runtime complexity divides between methods 1 and 3, and the others. In methods 1 and 3 the diagnosing agent(s) do not model other agents, but they obtain their beliefs by reporting, and disambiguate the diagnosis only by comparing between these beliefs, thus the complexity is polynomial in the number of agents and beliefs. On the other hand, in methods 2, 4 and 5 the beliefs of the other agents are not reported to the diagnosing agent, so it infers them by belief recognition process, which is exponential in the number of beliefs, and then diagnoses by querying.

In method 6 the diagnosing agent makes the diagnosis by reporting algorithm (whose complexity is similar to that of method 1 and 3). The selection of the diagnosing agent uses the `MINIMAL_QUERIES_DIAGNOSING_AGENT` algorithm. As shown above, this algorithm uses belief recognition, a process whose complexity is exponential in the number of beliefs.

9.2 Empirical Evaluation: Design of Social Diagnosis

We turn now to an empirical evaluation of the diagnosis methods presented in the previous section in two domains: One inspired by a real-world application (ModSAF), and one artificial (TEST).

9.2.1 Real-World Domain

We examined the diagnosing methods on teams of behavior-based agents in a simulation of the ModSAF domain. We performed experiments in which method 1 to method 6 were systematically tested on different failure cases, while we varied the number of agents, the roles of the agent and the disagreements between the agents.

1. Number of agents: teams of two to thirty six agents.
2. Roles of agent: for each *n* agents (1) one attacker and *n-1* scouts; (2) *n-1* attackers and one scout; (3) *n/2* attackers and *n/2* scouts.
3. Disagreements: we systematically checked all possible disagreement cases for all team behavior paths. Thus overall, each method was tested 33,000 times, or in average 950 trials for every team size.

We detected methods 1–6 on all failure cases. In each test we recorded the number of messages sent by all the agents, and runtime of the diagnosis process. For example, in Table 9.3 we present the results of a single test, where one scout and nineteen attackers fly in formation in *Fly Flight* behavior, when eight of the attackers, $A1--A8$, transition to the *Wait Point* behavior while the other attackers and the scout continue to fly in formation. The diagnosis is that $A1--A8$ detected the *way-point* (their belief is: *way point found=true*), while the other agents did not detect it (their belief is: *way point found=false*).

The first column in Table 9.3 reports the method used. The second column presents the number of messages sent reporting on beliefs, or querying about their truth (one message per belief). The third column reports the number of messages sent informing agents of the existence of failures (we assume point-to-point communications). The last columns summarize the runtime of each agent in milliseconds.

For instance, the number of messages reporting on beliefs for method 3 is 46, and 380 failure messages were sent (i.e., all the agents that detected the failure informed

Method	**Messages**		**Runtime(msec)**		
	Belief	**Failure**	**A1–A8,A10–A19**	**A9**	**Scout**
1	912	380	2	2	2
2	608	0	23	23	23
3	46	380	0	2	0
4	30	380	20	25	20
5	31	380	0	25	0
6	46	380	21	21	21

Table 9.3: Results of diagnosing a specific failure case: one scout and nineteen attackers fly in formation in *Fly Flight* behavior, when eight of the attackers, $A1 - -A8$, transition to the *Wait Point* behavior while the other attackers and the scout continue to fly in formation.

the others). The runtime of all the teammates for method 3 is 0 milliseconds, except for A9, which was selected in advance to disambiguated the beliefs in this case, and therefore took 2 milliseconds. On the other hand, the runtime of all the teammates for method 4 is 20 milliseconds, except for A9, which disambiguated the beliefs in this case, and therefore took an additional 5 milliseconds (for a total of 25 milliseconds). In this example test, in methods 3 and 5 we selected A9 in advance to make the diagnosis, since it was also the agent selected by the recursive-modeling process in methods 4 and 6. Thus we can show the difference in runtime between these methods.

Figures 9.1 and 9.2 summarize the results of the experiments. In both figures, the horizontal axis shows the number of agents in the diagnosed team. Figure 9.1 presents the average number of belief messages Each data point (team size) is an average over 950 runs (failure messages were ignored in the figure, since their effect is negligible). Figure 9.2 presents the average runtime (in milliseconds) of those same tests. The runtime of each test was taken as the maximum of any of the agents in the test.

Both figures show grouping of the evaluated techniques. In Figure 9.1 (number of messages), methods 3 to 6 show a slow, approximately-linear growth (methods 3 and 6 cover each other), while methods 1 and 2 show a much faster non-linear growth. In Figure 9.2 (runtime), the grouping is different: Methods 1 and 3 grow significantly slower than the other methods (they overlap).

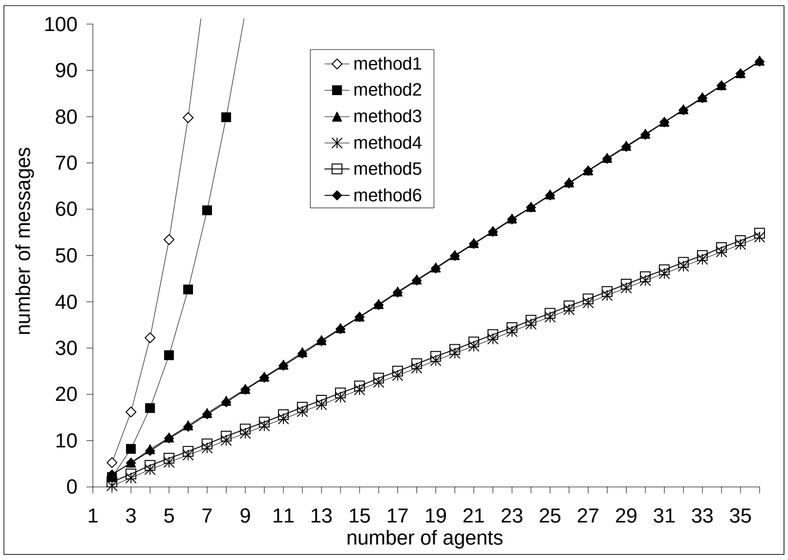


Figure 9.1: ModSAF: Average number of messages per failure case.

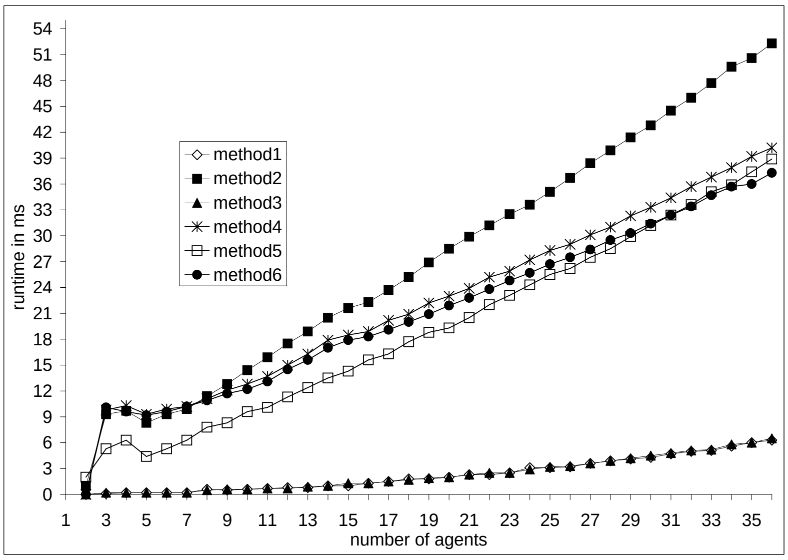


Figure 9.2: ModSAF: Average runtime in milliseconds per failure case.

According to Figure 9.1 the graphs of method 4 and 5 grow slower than the graphs of method 3 and 6, in contrast to their runtime performance (shown in Table 9.2). The reason for this is that method 4 and 5 use the querying algorithm, while method 3 and 6 use reporting algorithm. The communication complexity of the reporting algorithm in the best case is equal to the worst case: $O(nbm)$, since each agent always sends its own beliefs. On the other hand, while the complexity of the worst-case of the querying algorithm is $O(nbm)$, but in the best case it is $O(1)$. Averaged over thousands of tests the results of the querying algorithm are better than the reporting algorithm.

There are very small differences in Figure 9.1 between method 4 and method 5, as well as between method 3 and method 6, despite the fact that in methods 4 and 6 the minimal queries agent makes the diagnosis. The reason for this that the number of messages is almost the same (when the pre-defined agent, or the minimal queries agent makes the diagnosis) is that in the ModSAF domain all the agents have almost the same behavior hierarchy. As a result, the number of belief hypotheses of the minimal queries diagnosing agent is almost the same as the other agents, and so the benefits of the minimal queries diagnosing agent are not recognizable. In section 9.2.3, we will examine the benefit of the minimal queries diagnosing agent in more depth.

The first conclusion we draw from these figures is that runtime is mainly affected by the choice of a belief recognition process (Figure 9.2, Table 9.2). Methods (here, methods 1 and 3) that rely on the agents to report their relevant beliefs do not reason about the hypothesized beliefs of others. Therefore, their runtime is much smaller than methods (here, methods 2, 4 to 6) which hypothesize about the beliefs of others (methods 2, 4 and 5 use belief recognition in querying algorithm and method 6 uses belief recognition in selecting the minimal-queries diagnosing agent). However, as Figure 9.1 shows, the goal of reducing communications is actually achieved, as methods 4 to 6 do indeed result in less communications then method 3. The question of whether the cost in runtime is worth the reduction in communications is dependent on the domain.

We draw a second conclusion from Figure 9.1. Despite the additional savings provided by the minimal queries diagnosing agent algorithm, the choice of a centralized diagnosing agent is the main factor in qualitatively reducing the number of messages sent, as well as in shaping the growth curve as the number of agents is scaled up. These results contrast sharply with previous work in disagreement detection, in which distributed algorithms reduce communications [Kaminka and Tambe, 2000].

9.2.2 Evaluation in Controlled Settings

The experiments above were constrained to the parameters of the ModSAF domain, and thus limit the variance in the complexity of the agents. This section examine the diagnosis methods in settings where the number of beliefs (b, in Table 9.2), and the number of behavior path hypotheses (r) are controlled directly. For this aim we built a domain which simulates team with varied number of behavior-based agents. This domain, like the previous one, simulates teamwork in which the agents should agree on the selection of some pre-defined behaviors, concurrently. During the teamwork task the agents transition between the behaviors. The application simulates faults by controlling the selection of the behaviors by the agents, and causing disagreement in regard to the selected behavior. Unlike the MODSAF domain, in the domain of the simulated team, we control the number of beliefs and the number of behavior path hypotheses.

In the first set of experiments, we examine the influence of the number of beliefs (b in Table 9.2) on the runtime and the communication, of the diagnosis methods. For this goal, we performed experiments with a fixed number of agents (fifteen) and behavior path hypotheses (two), while the number of beliefs per behavior is varied from two to eight. The experiments tested with representative failure cases in a total number of 42.

The results of the communications in these experiments are presented in Figure 9.3 and that of the runtime are presented in Figure 9.4. Each data point is an average over six trials. The results in the graphs agree with the presented complexity analysis in Table 9.2. The communication complexity of all methods is approximately linear in the number of beliefs as shown in Figure 9.3. However, the growth of methods 4 and 5 is the slowest since the agent selected to carry out the diagnosis sends a minimal number of queries (querying algorithm), while the graph of method 1 grows much faster, since all the agents communicate with each other. On the other hand, Figure 9.4 shows that the runtime of methods 1 and 3 grows slowly and linearly (their runtime is closed to zero) while the other methods grow exponentially, in the number of beliefs, as predicted in Table 9.2.

In a second set of experiments, we examine the influence of the number of behavior path hypotheses (r in Table 9.2) on the runtime and the communication. In these tests the number of agents is fixed (thirty) as well as the number of beliefs per behavior (three), while the number of behavior path hypotheses is varied from two to ten. The experiments tested with representative failure cases in a total number of 315.

The results of the communications in the first tests are presented in Figure 9.5. Each data point is an average over approximately 35 trials. Here again, the results in the graphs agree with the complexity analysis in Table 9.2. The number of behavior path hypotheses have no influence on methods 1, 3 and 6, since in these methods the agents do not use behavior recognition (The graph of method 1 is out of the scope

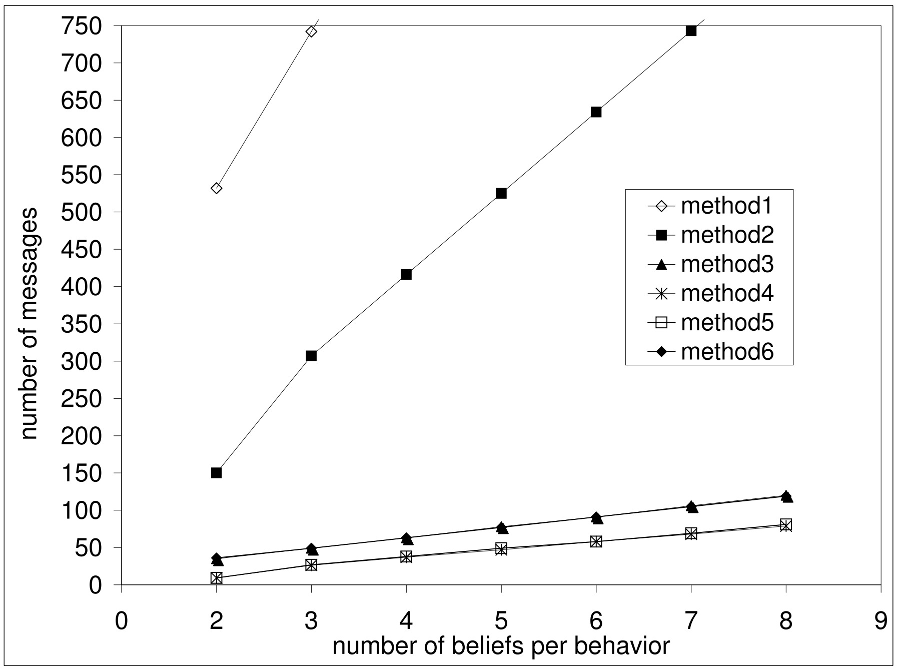


Figure 9.3: TEST domain: Average number of messages per failure case when varying number of beliefs. The number of agents is fixed at fifteen, and the number of behavior paths hypotheses is fixed at two.

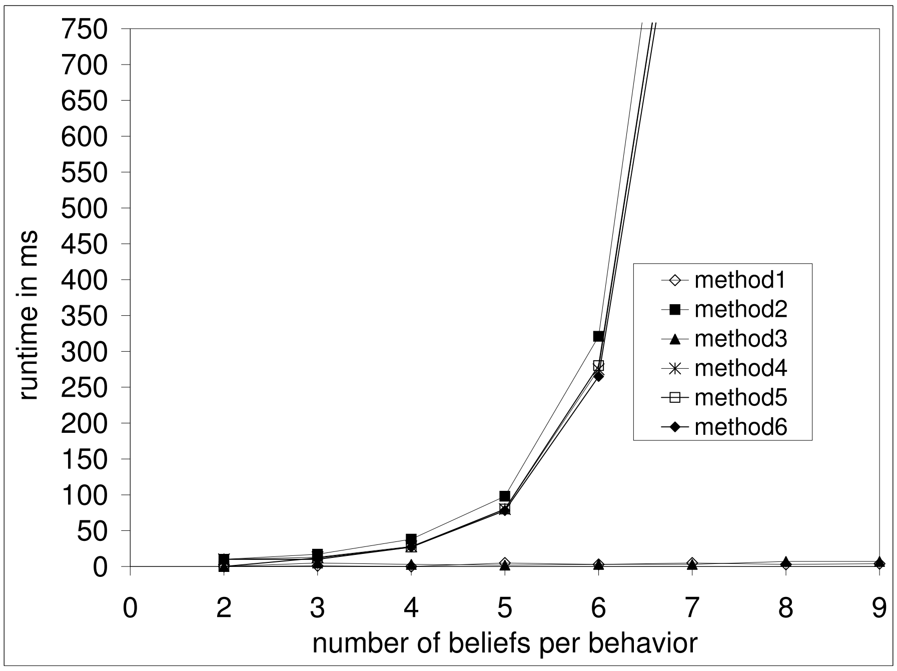


Figure 9.4: TEST domain: Average runtime per failure case when varying number of beliefs. The number of agents is fixed at fifteen, and the number of behavior paths hypotheses is fixed at two.

of the y axis and it is constant at 3132 messages). However, methods 2, 4 and 5 are affected by the number of behavior path hypotheses. According to the complexity in Table 9.2 we expect to obtain a linear curve in the number of messages (Figure 9.5), and indeed in practice the graphs' growth is linear. The reason is that the graphs' growth depends on the partition of the belief hypotheses space. As discussed in Section 7.2.2, the selected query by the diagnosing agent divides this space. Tt is bounded from above with the number of beliefs which influences on the graph to be **linear** with the number of beliefs. On the other hand it is bounded from below with the constant one which influences on the graph to be **constant** with the number of beliefs. Therefore, the graphs of methods 2, 4 and 5 involve constant, linearly and logarithmic growths.

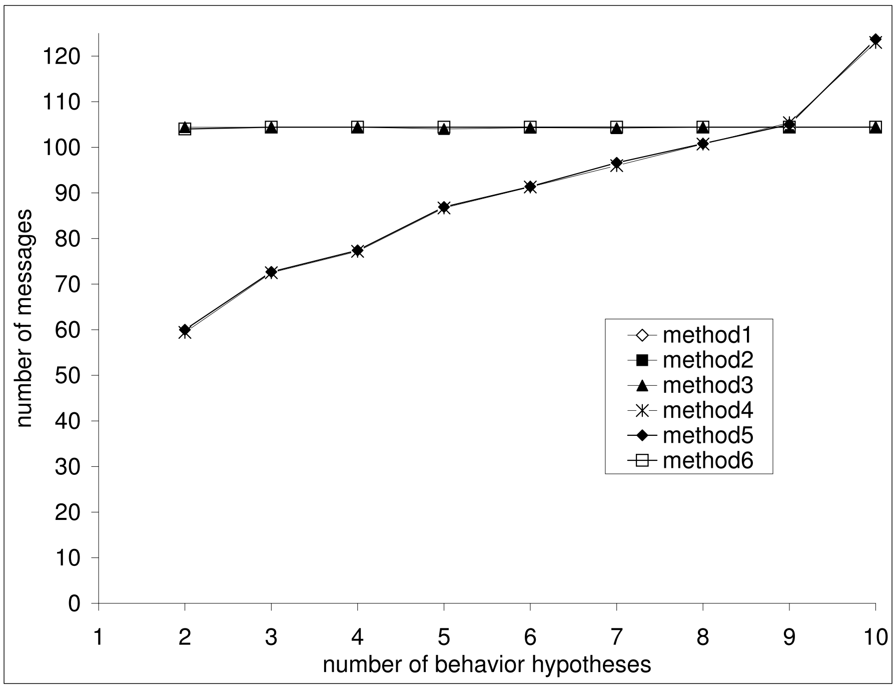


Figure 9.5: TEST domain: Average number of messages per failure case when varying number of behavior path hypotheses. The number of agents is fixed at thirty, and the number of beliefs is fixed at three.

Figure 9.6 shows the runtime results in these experiments. As expected, the runtime complexity of methods 2 and 4 to 6 grow quickly as the number of behavior path hypotheses grows since they use the querying algorithm where the runtime is affected by the number of hypotheses, while the graphs of method 1 and 3 are approximately fixed at close to 0 milliseconds, since they use the reporting algorithm which does not depend on the number of behavior path hypotheses.

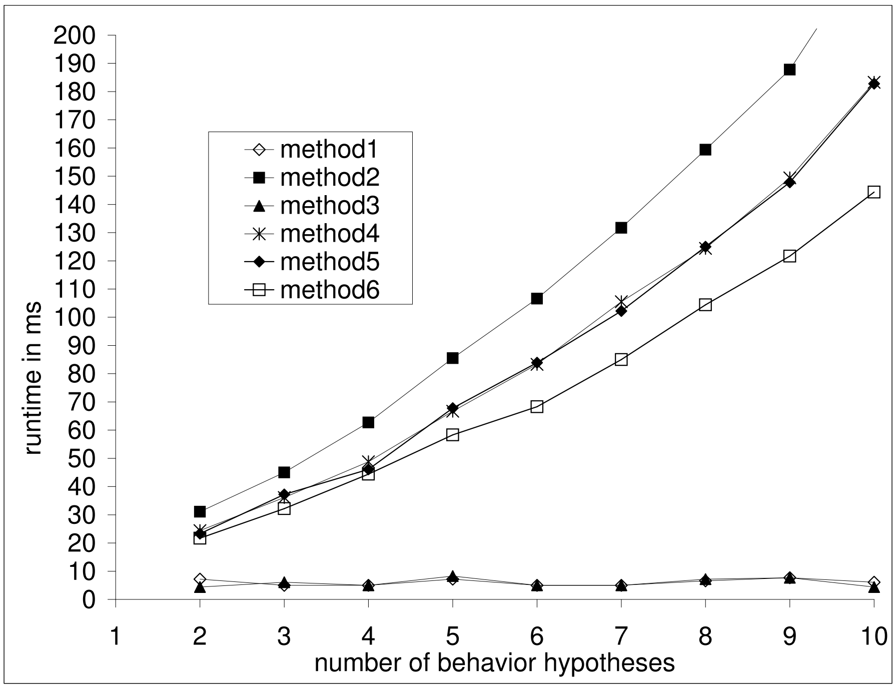


Figure 9.6: TEST domain: Average runtime per failure case when varying number of behavior path hypotheses. The number of agents is fixed at thirty, and the number of beliefs is fixed at three.

9.2.3 Minimal-Queries Diagnosing Agent

In a final set of experiments, we examine the efficiency of the selection of the minimal queries diagnosing agent, by comparing between methods 4 and 5. Both of these methods use the querying algorithm to make the diagnosis, but while the diagnosing agent in method 4 is expected to ask the minimal queries, in method 5 it is selected in advance.

The comparison of these methods in the ModSAF domain yields very little difference, since the number of the involved beliefs in all behavior path hypotheses of the agents is almost the same. As a result, any difference between the number of queries of the minimal queries diagnosing agent and the other agents is very small.

In contrast, in the following experiments, we control the number of beliefs of the behaviors of each agent separately. The number of agents is fixed (four) as well as the number of behavior path hypotheses (two), while the number of beliefs per behavior is varied from two to ten **only** for a single random agent while it is fixed (three) for the other agents. We expect that in method 4, this agent will be selected to make the diagnosis according to the selection of the `MINIMAL_QUERIES_DIAGNOSING_AGENT` algorithm (Algorithm 6).

Figure 9.7 summarizes the communication results in the experiments. Each data point is an average over six trials. The graph of method 4 shows a fixed number of messages, independently of the growth of the number of beliefs. The reason is that in this method the agent with the most number of beliefs is selected to make the diagnosis, while the other agents are queried for their beliefs (they have the same number of beliefs in all trials). On the other hand, the graph of method 5 is dependent on the number of beliefs of the random agent, since the diagnosing agent is selected randomly and it may be not the minimal queries diagnosing agent, so the number of sent messages grows. We can see that although the general tendency of the graph of method 5 is growing up, it decreases in the points of four and eight beliefs. The reason is that incidentally the diagnosing agent who was randomly selected has the most number of beliefs so it is predicted to query the minimal number of queries similarly to the minimal queries diagnosing agent in method 4.

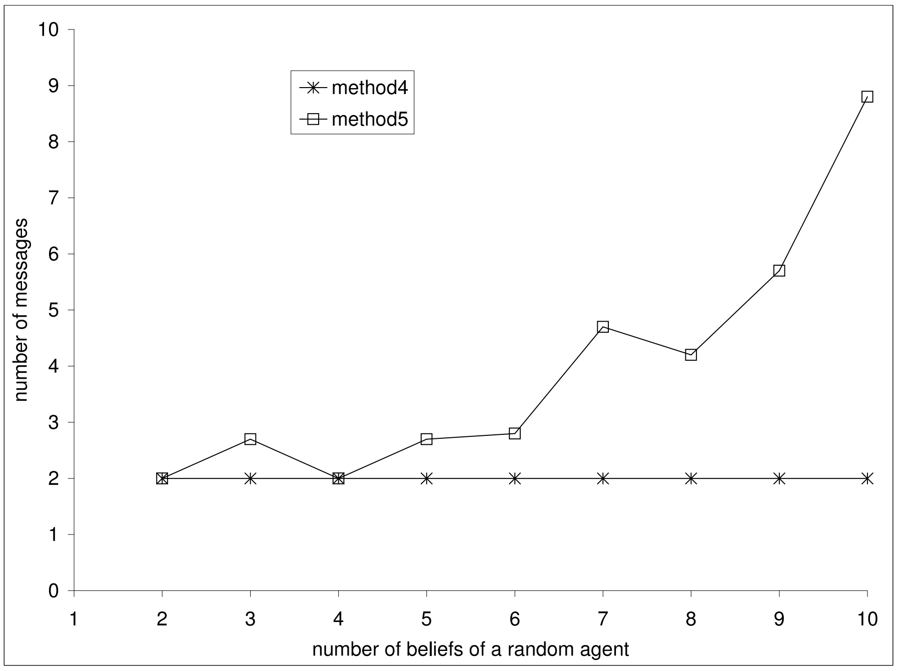


Figure 9.7: TEST domain: Average number of messages per failure case when varying number of beliefs per behavior for a single random agent. The number of agents is fixed at four, and the number of behavior path hypotheses is fixed at two.

Figure 9.8 summarizes the runtime results. We can see, as predicted, that both the runtime of method 4 as well as of method 5 grow exponentially, since both of them use belief recognition algorithm in the querying algorithm.

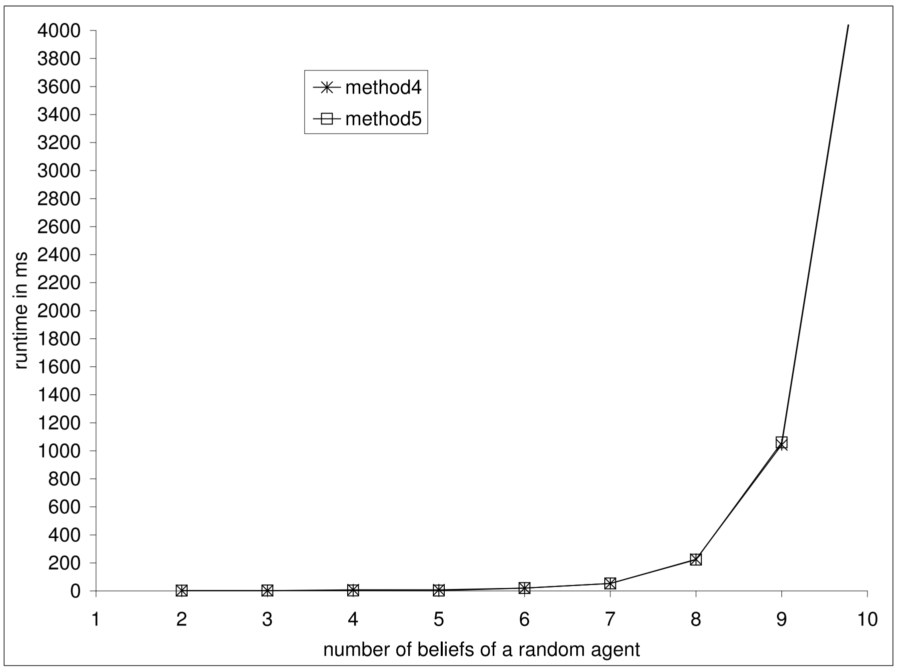


Figure 9.8: TEST domain: Average runtime per failure case when varying number of beliefs per behavior for a single random agent. The number of agents is fixed at four, and the number of behavior path hypotheses is fixed at two.

9.3 Diagnosis Methods for Large Teams

In Chapter 8 we presented three techniques which utilize these ideas: (i) BEHAVIOR QUERYING (Section 8.1) using targeted queries to alleviate diagnostic reasoning; (ii) SHARED BELIEFS (Section 8.2) using light-weight behavior recognition to focus on beliefs that may be in conflict; and (iii) GROUPING (Section 8.3) the agents by their role and behavior and then diagnosing each group based on representative agents.

In order to evaluate these techniques we present four diagnosis methods based on these techniques and on the original methods we reported in Section 9.1.

BEHAVIOR. The diagnosing agent uses only BEHAVIOR QUERYING (Section 8.1) in order to disambiguate the behavior path of the observed agents. Once the behavior path of each monitored agent is known, the diagnosing agent continues to diagnose using the remaining phases of the QUERYING algorithm (belief recognition and querying).

BELIEF. The diagnosing agent uses behavior recognition in order to build the behavior path hypotheses of the observed agents, then it continues with the SHARED BELIEFS method (Section 8.2) to find the suspect belief conflicts and generate the diagnosis.

BEHAVIOR+BELIEF. This method combines BEHAVIOR QUERYING and SHARED BELIEFS methods. The diagnosing agent uses BEHAVIOR QUERYING to determine the behavior path of the observed agent, and then continues to diagnose the disagreements using the SHARED BELIEFS method.

GROUPING. The last method adds a grouping technique (Section 8.3) to the BEHAVIOR+BELIEF combination. Once the behavior path of each monitored agent is known using BEHAVIOR QUERYING, it divides the team to groups according to their role and behavior path, and continues to compute the diagnosis using SHARED BELIEFS method against the representative agents of the groups.

We compare these methods to the original QUERYING algorithm (Section 7.2.2) and to the REPORTING method (Section 7.2.1), which relies on complete communication with no inference other than for the comparison step.

Method	**Runtime complexity**
REPORTING	$O(n^2b^2m^2)$
QUERYING	$O(nr2^{2bm} + n^2b^2m^2)$
BEHAVIOR	$O(n2^{2bm} + n^2b^2m^2)$
BELIEF	$O(r^2n^2b^2m^2)$
BEHAVIOR+BELIEF	$O(n^2b^2m^2)$
GROUPING	$O((sp)^2b^2m^2)$

Table 9.4: Summary of evaluated diagnosis methods and their runtime worst-case complexity.

Table 9.4 summarizes the worst-case complexity of the algorithms. The first two rows describe those methods presented in Section 9.1 (REPORTING and QUERYING). The complexity of REPORTING is affected by the comparison between the beliefs (b^2m^2) of n agents (n^2). QUERYING also performs the same comparison ($n^2b^2m^2$), but first it recognizes the beliefs by behavior recognition (r behavior path hypotheses of n agents) and belief recognition processes (2^{2bm}).

The next four rows present the different methods presented above the table. BEHAVIOR method is exactly as QUERYING except of the using of BEHAVIOR QUERYING instead of behavior recognition, and so it has only a single behavior path hypothesis instead of r. The next algorithm, BELIEF, uses behavior recognition to build the behavior path hypotheses (r), then it finds the shared beliefs by comparing between the belief propositions (bm) of n agents. BEHAVIOR+BELIEF combines BEHAVIOR QUERYING technique instead of behavior recognition in QUERYING algorithm (which replaces the r behavior path hypotheses by a single one), and SHARED BELIEFS technique instead of belief recognition in QUERYING algorithm ($n^2b^2m^2$). The last algorithm, GROUPING, uses the previous one (BEHAVIOR+BELIEF) but only on representative agents ($s * p$) instead of the whole group (n).

9.4 Empirical Evaluation of Diagnosis in Large Teams

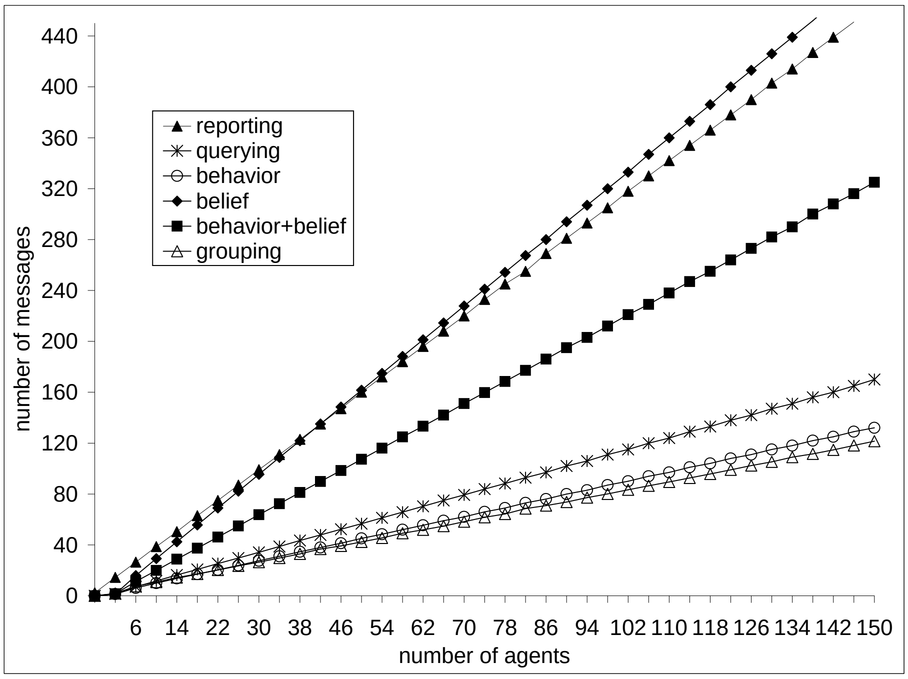
In the following sections we will empirically examine the performance of the methods presented in the previous section through thousands of tests in two domains (Sections 9.4.1 and 9.4.2, resp.).

9.4.1 Simulation of a Real-World Application

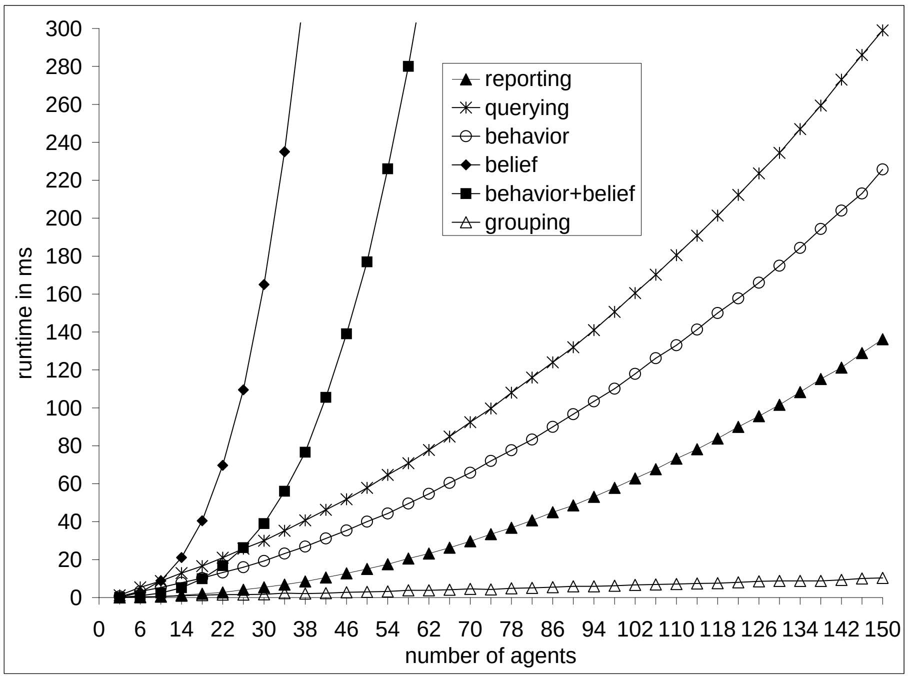
In Section 7.1 we describe the use of diagnosis algorithms in a simulation of a real-world application (ModSAF), a virtual battlefield environment containing teams of synthetic helicopter pilots. We recreated the agents' behavior hierarchy in this domain, and determined their behavior in large-scale settings by simulating disagreements in teams much larger than originally described.

We performed experiments in which we varied the number of synthetic pilots from 2 to 150 (in jumps of 4). For each team size (n agents), we varied the selected behavior path of each agent, and the role of the agents (two roles, *scouts* and *attackers).* We ran three sets of tests: (1) one attacker and $n-1$ scouts; (2) $n-1$ attackers and one scout; (3) $n/2$ attackers and $n/2$ scouts. Overall, for every n agents, we tested close to 60 failure cases, varying the behavior paths (4 options) selected by the agents. For each single test we measured the number of messages sent and the runtime by each one of the diagnosis methods.

Figure 9.9(a) summarizes the results of these experiments. It compares the different diagnosis methods in terms of the average number of belief messages they utilize. The X axis shows the number of agents in the diagnosed team and the Y axis presents the number of messages. Each data point is an average over approximately 60 trials.



(a) *ModSAF domain: number of messages.*



(b) *ModSAF domain: runtime.*

Figure 9.9: ModSAF domain: Diagnosis runtime and number of messages for 2–150 agents.

The growth of the SHARED BELIEFS method (BELIEF) appears similar to that of the REPORTING algorithm (REPORTING). We believe that this is because in teams, the behavior paths selected by different agents refer to the same propositions, to a large degree. Thus the number of shared beliefs (that are then communicated) is in fact very close to the total number of beliefs (which are all communicated in the REPORTING method).

The BEHAVIOR QUERYING method (BEHAVIOR) shows limited improvement relative to the QUERYING algorithm (QUERYING) graph. We believe this is because in the ModSAF domain there are only few possibilities of behavior path hypotheses and belief hypotheses, and as mentioned above (section 8.1) the benefit of this method is in the disambiguation of a high number of behavior path hypotheses and/or belief hypotheses by a single query about the behavior path of the diagnosed agents.

The combination between BEHAVIOR QUERYING and SHARED BELIEFS methods (BEHAVIOR+BELIEF) is worse than BEHAVIOR QUERYING and better than SHARED BELIEFS alone. The reason is that indeed it saves communication in the beginning of the diagnosis process by disambiguating the behavior path of the agents by one query, but then it uses more queries in order to disambiguate between the shared beliefs.

The GROUPING method is also better than the QUERYING algorithm as shown in Figure 9.9(a), since the diagnosis communication is done only against the representative agents of the groups. Although the number of the representative agents is fixed through the tests, communication depends linearly on the number of agents since each added agent is queried about its behavior path. In an application with a high number of behavior path hypotheses and/or belief hypotheses we predict a significant growth in the QUERYING graph in contrast to the GROUPING graph which will remain the same (since the communication growth is affected only by the queries that disambiguate the agents' behavior path).

Figure 9.9(b) presents the average runtime (in CPU milliseconds) of the different methods. The runtime of each test is taken as the maximum of any of the agents in the test. All the curves except GROUPING grow polynomially as expected from Table 9.4. Surprisingly, the SHARED BELIEFS (BELIEF) method grows much faster than QUERYING. The reason for this is that the SHARED BELIEFS method compares all the beliefs that are associated with all of the behavior path hypotheses of all the agents, **before** disambiguating the beliefs' values. This is done to determine what propositions are possibly shared between agents, and may thus be in conflict. On the other hand, in the QUERYING algorithm, the comparisons are done only between the beliefs of the agents **after** that have already been disambiguated, so only the actual beliefs of the agents are compared (although the inference preceding the querying is exponential in the number of beliefs). We can see it also in Table 9.4. The belief has additional factor r^2, the number of behavior path hypotheses, which affects the runtime.

The combination of the SHARED BELIEFS and BEHAVIOR QUERYING methods (BEHAVIOR+BELIEF) shows a slight improvement with respect to SHARED BELIEFS alone (BELIEF), since the comparisons are now done between the beliefs that are associated with only the behavior path hypothesis of the agents (instead of all the behavior path hypotheses, as Table 9.4 shows that the factor r^2 does not exist in BEHAVIOR+BELIEF). However, although Table 9.4 shows that BEHAVIOR+BELIEF is better than QUERYING, we should pay attention that the factor bm represents the worst case. In the BEHAVIOR+BELIEF algorithm all the hypothesized beliefs (of the single behavior path) are compared before disambiguating their value, while in the QUERYING algorithm the diagnosing agent compares the actual beliefs (after they have been disambiguated).

On the other hand, as expected, the BEHAVIOR QUERYING method (BEHAVIOR) improves the runtime relative to the QUERYING algorithm, since it saves the belief recognition of all the beliefs that are associated with the behavior path hypotheses that have not been disambiguated as the correct one. Indeed Table 9.4 shows that the factor r (number of behavior path hypotheses) does not exist in BEHAVIOR QUERYING. However, it is still polynomial in the number of agents, since agents' beliefs are compared.

Undoubtedly, the significant runtime improvement is in the GROUPING method, since its complexity is linear, rather than polynomial, as also evident in Figure 9.9(b). The reason is that the number of representative agents is fixed by the product of the number of behavior path hypotheses (p in Table 9.4) and the number of agents' roles (s in Table 9.4), so the number of comparisons between their beliefs is bounded, too. This result is surprising given the reliance of GROUPING on the BEHAVIOR+BELIEF combinations, which do not do well.

The conclusion we draw from these results is that while in general runtime grows polynomially in the number of agents (because of the comparisons), the GROUPING method reduces the complexity to a slow linear growth due to the fixed number of comparisons. In addition, the reduced number of comparisons causes a reduction in the number of messages. On the other hand, it seems according to the figures that the other two methods, BEHAVIOR QUERYING and shared beliefs, do not contribute to the reduction of either the runtime or the number of messages.

9.4.2 A Synthetic Domain

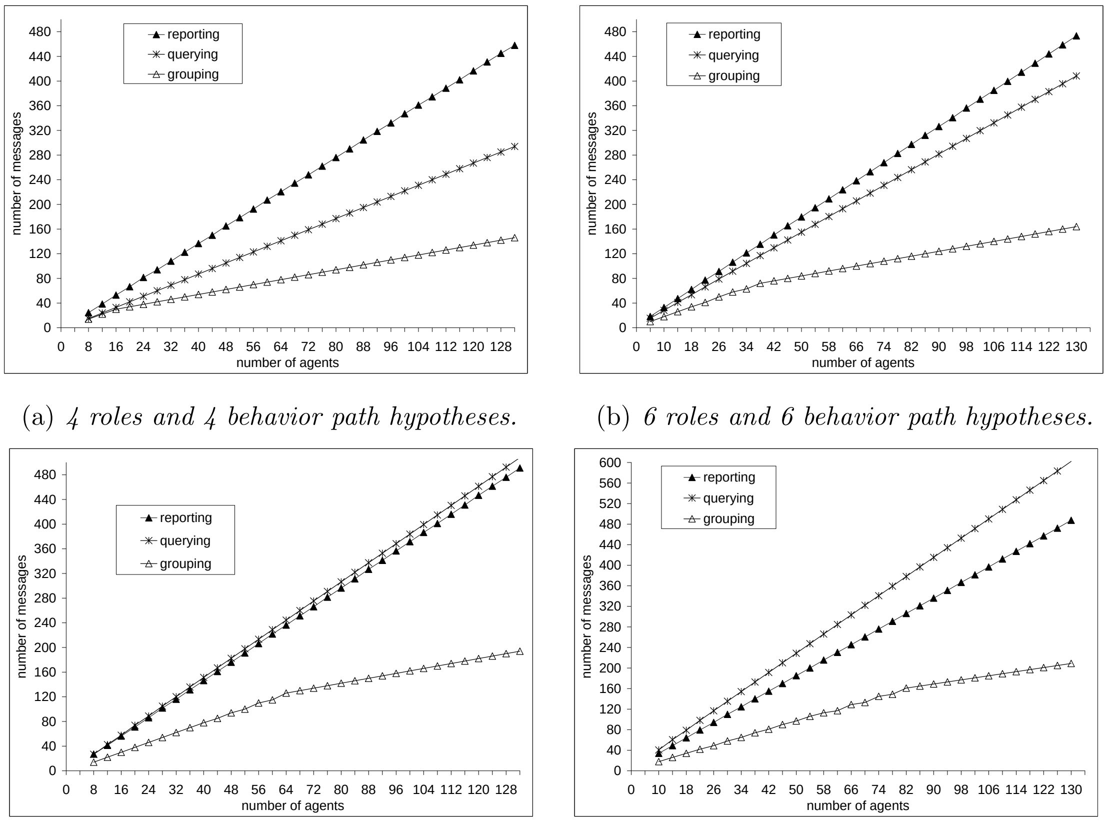
The conclusions in the former section lead us to two questions: First, to what degree do the results of the GROUPING method depend on the characteristics of the ModSAF domain—low number of agent roles (two) and behavior paths (four)? And second, are there benefits to BEHAVIOR QUERYING and the SHARED BELIEFS methods?

In order to address these questions we examine the diagnosis methods while varying parameters such as roles and behaviors. To do this, we have created an artificial domain called TEST, in which we vary (1) the number of agents, (2) the number of roles, (3) the number of behavior path hypotheses and (4) the number of beliefs per behavior. The actions in this domain are defined only to the degree that allows their recognition (as part of the diagnosis process). The behaviors do not correspond to any specific task, but are structured in a way that mimics the hierarchy of the ModSAF domain's behaviors.

GROUPING **Benefits.** A key feature of the GROUPING method is that the number of representative agents is bounded from above, by the minimum of (i) the number of agents in the team, and (ii) the number of groups. Since groups are distinguished during diagnosis based on the combination of roles and selected behaviors, the number of groups, for any disagreement, cannot exceed the product of the number of roles and number of behavior paths in the behavior hierarchy.

Figures 9.10 and 9.11 show the results from experiments with this feature. We arranged four experiments, in which we fixed the number of roles and the number of behavior paths in the behavior hierarchy at: (i) four (Figures 9.10(a),9.11(a)); (ii) six (Figures 9.10(b),9.11(b)); (iii) eight (Figures 9.10(c),9.11(c)); and (iv) nine (Figures 9.10(d),9.11(d)). Since groups are distinguished based on role-behavior path combination, the maximal number of groups is the product of these factors, namely, in the first experiment 16, in the second 36, in the third 64 and in the last one 81.

For each of the configurations, we ran the diagnosis methods in teams of up to 150 agents. Each test was examined in maximal disagreement (i.e., worst case), in the sense that every agent tried to select behaviors and roles different from its peers. For instance, for twelve agents in the second experiment, six roles are divided equally between the agents, and for each two agents that have the same role, they select different behavior paths. Overall, each data point in the figures is an average over these six trials.



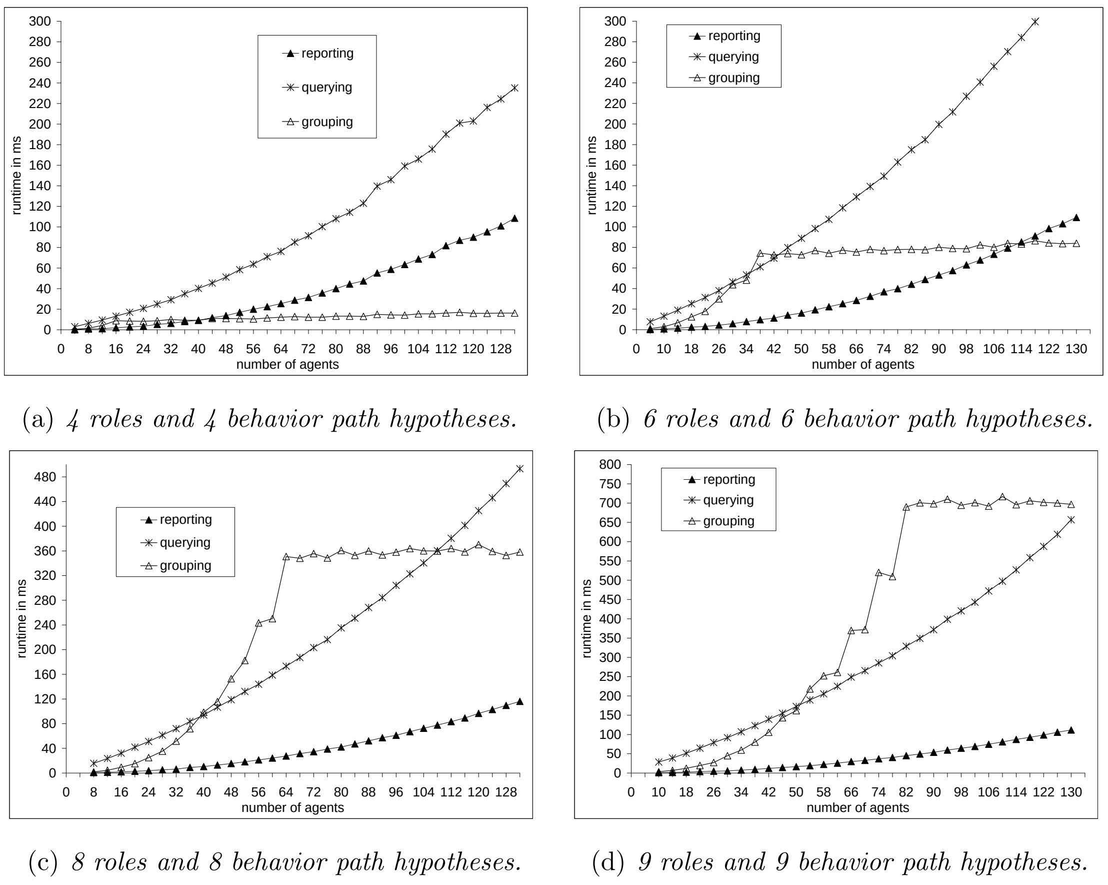
(a) *4 roles and 4 behavior path hypotheses.* (b) *6 roles and 6 behavior path hypotheses.*

(c) *8 roles and 8 behavior path hypotheses.* (d) *9 roles and 9 behavior path hypotheses.*

Figure 9.10: TEST domain: The number of messages of the GROUPING algorithm in large-scale team, in varied number of roles and behavior-path hypotheses.

Figure 9.10 shows the number of messages of the GROUPING method compared with QUERYING and REPORTING. We can see that around the point of the product of the number of roles and number of behavior paths, the linear graph of the GROUPING method changes its angle and grows much slower. The same phenomenon occurs in Figure 9.11, that shows average runtime in these experiments. The graph is approximately polynomial as long as the number of agents is smaller than the product of the number of roles and number of behavior paths, then the graph becomes approximately constant since this number is bounded.

We believe that the GROUPING method is suited for large-scale teams. As teams grow, the number of groups (and therefore the number of diagnosed representative agents) is likely to be much smaller than the total number of agents in the teams, even if we assume that the complexity of the different agents (in terms of roles and behaviors) would also be higher than in the experiments above. Figures 9.11(a) and 9.11(b) show that when the number of agents is less than 132, using the GROUPING



(a) *4 roles and 4 behavior path hypotheses.* (b) *6 roles and 6 behavior path hypotheses.*

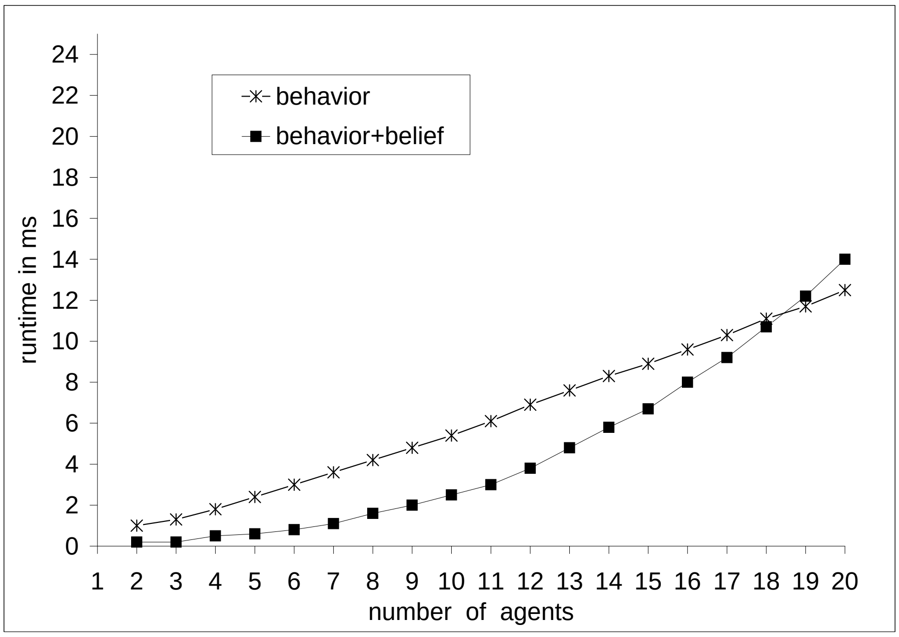
(c) *8 roles and 8 behavior path hypotheses.* (d) *9 roles and 9 behavior path hypotheses.*

Figure 9.11: TEST domain: Runtime of the GROUPING algorithm in large-scale team, when varying roles and behavior-path hypotheses.

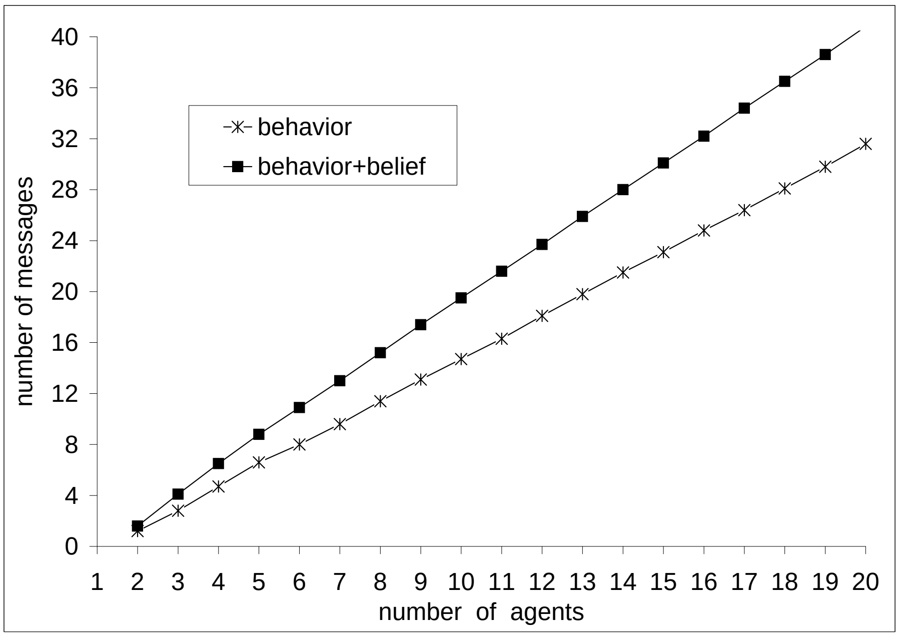
algorithm is preferable even to the REPORTING method. From the shape of the curves in Figures 9.11(c) and 9.11(d), we can conclude that in large-scale teams, GROUPING is preferable.

Let us now turn to examining the benefits of the BEHAVIOR QUERYING and SHARED BELIEFS methods. We believe there are two ways in which these methods can be beneficial to the diagnosis process: First, by combining them with the GROUPING method; and second, in settings involving a large number of behavior path hypotheses and number of beliefs.

Combining the Three Methods. The GROUPING method is composed of two stages: Dividing the agents into groups according to their role and selected behavior path; and diagnosing the representative agents of the groups, where the results are assumed to hold for the other agents. In order to diagnose the representative agents in the second stage, we can use either belief recognition and comparison between beliefs by the QUERYING algorithm or the SHARED BELIEFS method. Since the number of diagnosed agents is relatively small (only representative agents are



(a) *runtime.*



(b) *number of messages.*

Figure 9.12: ModSAF domain: comparison between BEHAVIOR and BEHAVIOR+BELIEF in small teams (2–20).

diagnosed), it is important to choose a method that works well in small teams. In the experiments we ran in the previous section (9.4.1), we preferred the SHARED BELIEFS method.

To evaluate this choice, Figures 9.12(b) and 9.12(a) show the communication and runtime results, respectively, of BEHAVIOR QUERYING (that uses BEHAVIOR QUERYING in order to disambiguate the behaviors, and then continues to the QUERYING algorithm) and the combination of the BEHAVIOR QUERYING and the SHARED BELIEFS, in diagnosing small teams (up to 20 agents, close to 60 trials per data point). We see that the two methods are close in terms of communications (Figure 9.12(b)) while the SHARED BELIEFS (BEHAVIOR+BELIEF) is better than QUERYING in terms of runtime (Figure 9.12(a)). However, we remind the reader that with larger team sizes, QUERYING runs faster than SHARED BELIEFS, and thus with a large number of groups generated by the GROUPING method, it may be preferable to diagnose representative agents using belief recognition, instead of SHARED BELIEFS.

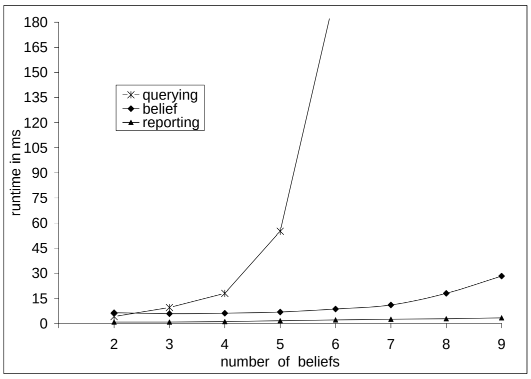
We can explain the difference between small teams and large teams by the difference in the complexity between BEHAVIOR ($O(n2^{2bm} + n^2b^2m^2)$) and BEHAVIOR+BELIEFS ($O(n^2b^2m^2)$) algorithms shown in Table 9.4. The algorithm BEHAVIOR uses belief recognition to disambiguate the correct beliefs of the observed agents ($O(n2^{2bm})$) and then diagnoses the disagreements by comparing these beliefs ($O(n^2b^2m^2)$). On the other hand, BEHAVIOR+BELIEF uses SHARED BELIEFS in order to make the diagnosis, and so it compares between all the beliefs of the observed agents **before** disambiguating the correct beliefs. Therefore, the coefficient of the number of agents b^2m^2 in the BEHAVIOR algorithm is smaller than the same coefficient in BEHAVIOR+BELIEFS. However, BEHAVIOR has one more factor in its complexity nm^{2b}, which grows only linearly in the number of agents. Therefore, in small teams the graph is affected by this factor, and so BEHAVIOR+BELIEF is better than BEHAVIOR. But in large teams this factor is insignificant relatively to the polynomial factor of $n^2b^2m^2$ (with a small coefficient), and so BEHAVIOR is better than BEHAVIOR+BELIEF due to the effect of the coefficient.

SHARED BELIEFS Benefits. A second benefit of SHARED BELIEFS is in the high number of beliefs and behavior path hypotheses. The complexity of SHARED BELIEFS method is polynomial in the number of beliefs (Section 8.2). This is in contrast to the QUERYING algorithm that grows exponentially in the number of beliefs. However, this computational advantage does not manifest itself in the ModSAF domain, since in the ModSAF domain tests, only the number of agents is varied where the number of beliefs is fixed and small.

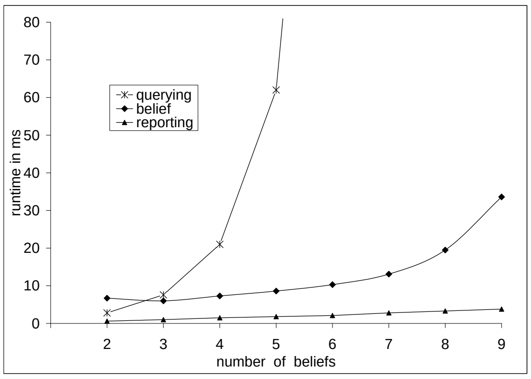
To examine the effects of this difference between SHARED BELIEFS and QUERYING, we compare them in settings involving a larger number of beliefs, in the TEST domain. In these experiments we vary the number of beliefs from two to nine per behavior path in a sequence of 8 tests from a varied number of agents from 11 to 18. Figure 9.13 summarizes the results of these experiments (6 trials per data point). The X axis shows the number of beliefs per behavior path and the Y axis shows the runtime in CPU milliseconds. Obviously, REPORTING shows the best results since it involves comparison between the correct beliefs and between hypothesized beliefs as in the SHARED BELIEFS algorithm. However, we can see that while the QUERYING graph grows exponentially, the SHARED BELIEFS graph grows polynomially very slowly. The relation between the curves consist along a varied number of agents (from 11 agents in 9.13(a) to 18 agents in 9.13(h)). The implicit conclusion is that in a domain that involves a high number of beliefs, SHARED BELIEFS would be preferable to QUERYING.

BEHAVIOR QUERYING **Benefits.** The BEHAVIOR QUERYING method has a similar benefit, with respect to a high number of behavior path hypotheses. As we have shown in Section 8.1 and in Table 9.4, the number of messages in the QUERYING method depends on the number of behavior path hypotheses. As the number of behavior path hypotheses grows, it typically multiplies the number of belief hypotheses, and this results in requiring many more queries to disambiguate the belief hypotheses. The intention behind BEHAVIOR QUERYING is to eliminate all behavior path hypotheses but one, by directly querying about the behavior path of the observed agent. In a domain where the potential number of behavior path hypotheses is small (e.g., only two in the ModSAF domain), the benefit of the BEHAVIOR QUERYING is not realized. Therefore, we examine it in the TEST domain. In this set of experiments the number of beliefs per behavior is fixed at three, while the number of behavior path hypotheses is varied from two to ten. We examined sets of tests along a varied number of agents from three to ten.

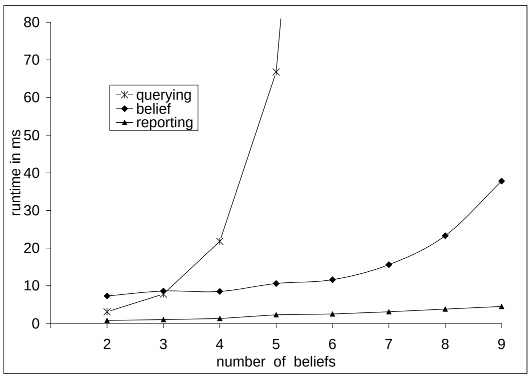
Figure 9.14 summarizes the results of the experiments. The X axis shows the number of behavior path hypotheses, while the Y axis shows the number of messages. Both the BEHAVIOR QUERYING method (BEHAVIOR) as well as the GROUPING method (that relies on the BEHAVIOR QUERYING) are essentially constant in the number of sent messages, since once the behavior path of the observed agent is disambiguated the rest of the process depends on the number of agents and the number of beliefs, where these parameters are fixed here. On the other hand, the QUERYING algorithm grows with the number of behavior path hypotheses. We conclude that BEHAVIOR QUERYING can be very beneficial in domains involving a large number of behavior path hypotheses.



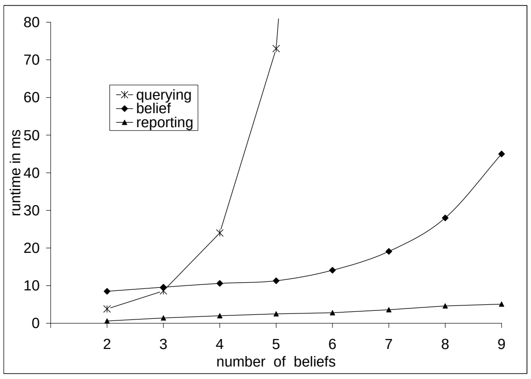
(a) *11 agents.*



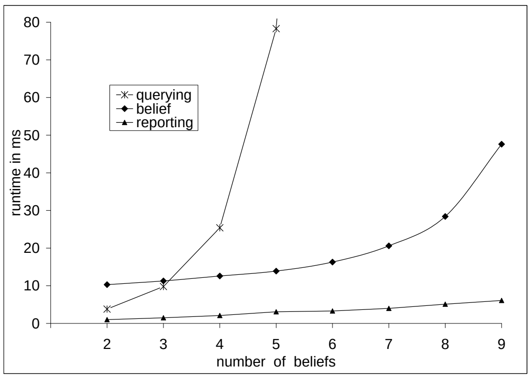
(b) *12 agents.*



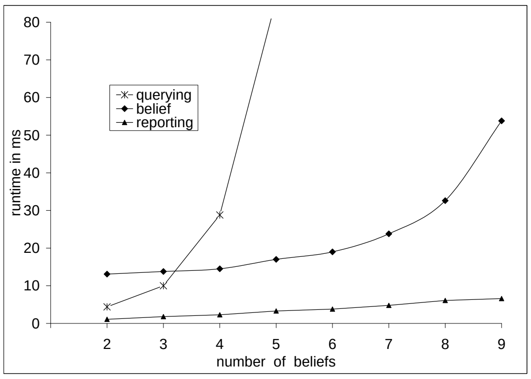
(c) *13 agents.*



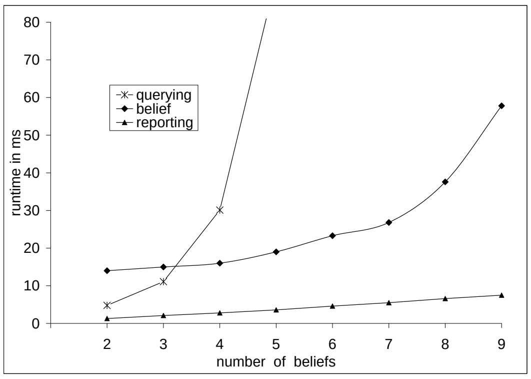
(d) *14 agents.*



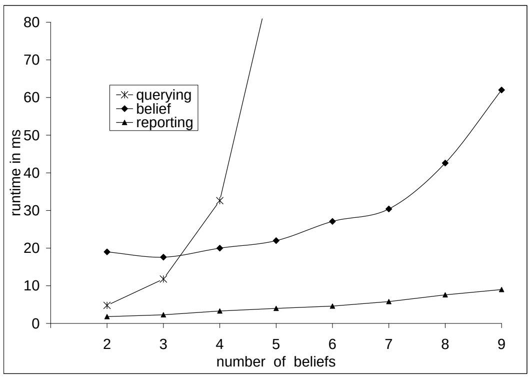
(e) *15 agents.*



(f) *16 agents.*

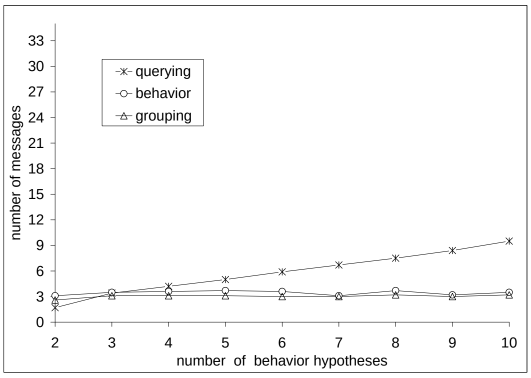


(g) *17 agents.*

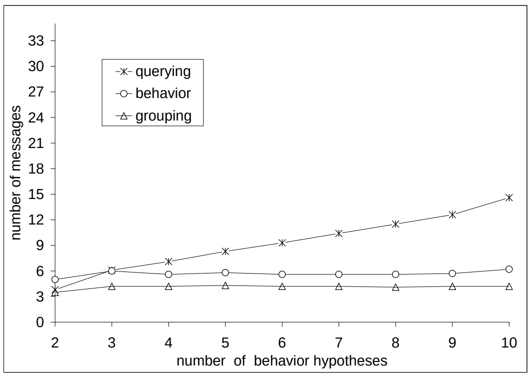


(h) *18 agents.*

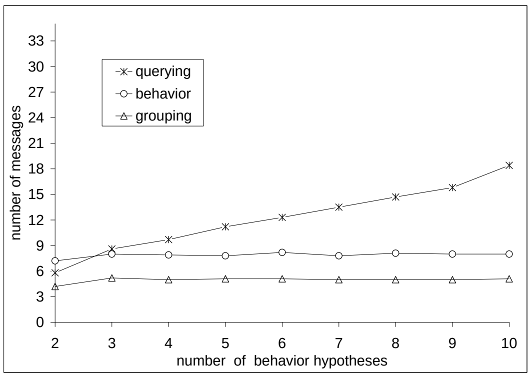
Figure 9.13: TEST domain: runtime in varying number of beliefs per behavior.



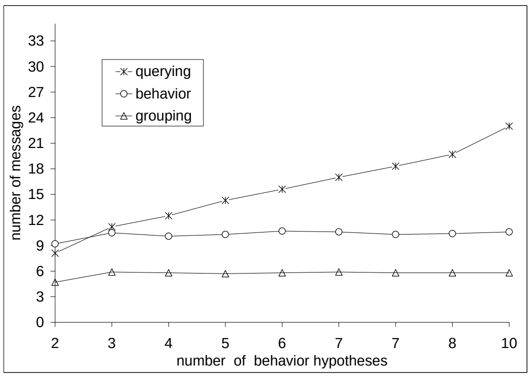
(a) *3 agents.*



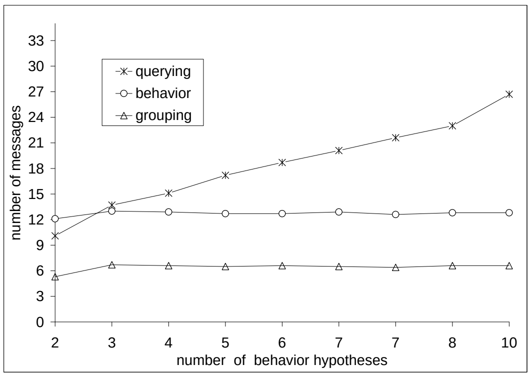
(b) *4 agents.*



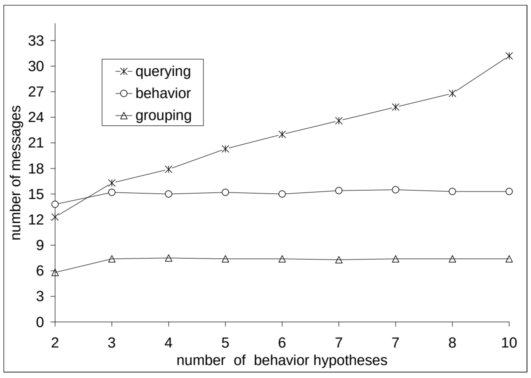
(c) *5 agents.*



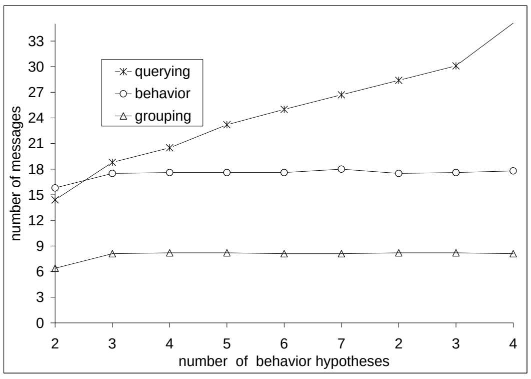
(d) *6 agents.*



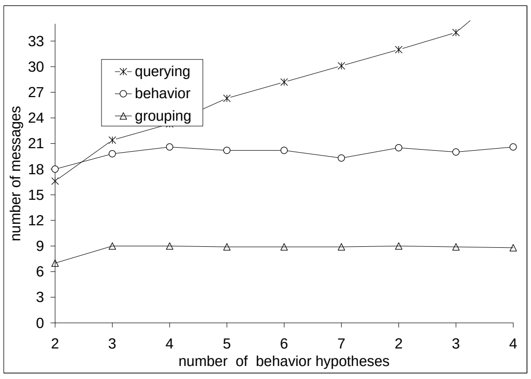
(e) *7 agents.*



(f) *8 agents.*



(g) *9 agents.*



(h) *10 agents.*

Figure 9.14: TEST domain: number of messages in varying number of behavior path hypotheses.

Chapter 10

Summary and Future Work

In this dissertation we coped with the problem of diagnosing coordination faults in teams. We discussed aspects of the problem that have not been addressed in the literature, and presented the following contributions:

1. Formalizing the problem in terms of model-based diagnosis, and proving its complexity is NP-Hard.

2. Presenting a distributed approach for the diagnosis, and evaluating its performance empirically.

3. Drawing lessons about a design space for coordination diagnosis.

4. Presenting algorithms and approaches for diagnosis of large-scale teams.

Here we will summarize the contribution in each one of these aspects. Finally in Section 10.5 we will present new directions for future work.

10.1 Formalism

We presented a novel formalization for diagnosing coordination failures in multi agent systems, in terms of model-based diagnosis. In our model the diagnoser does not know the inputs of the agents, instead it relies on a model of the coordination between the agents. We model such coordination using two coordination primitives (concurrence and mutual exclusion). In the diagnosis process the diagnoser observes, the actions of the agents, then it finds the candidate abnormal agents by the coordination model.

We defined both a consistency-based and abductive diagnosis versions of coordination diagnosis, and proposed centralized algorithms for both. The consistency-based approach finds the local conflicts between pairs of agents, then continues to compute the diagnosis by combining the conflicts using a minimal vertex cover algorithm. We showed that this approach is unsound, in that it may produce diagnoses that are impossible, in that they cannot be corrected. The second approach maps the abductive coordination diagnosis problem to that of satisfiability, finding a minimal set of truth-value changes that satisfy a given proposition. Here, our initial approach pre-computes all the possible coordination-satisfying action assignments, and then uses these during on-line diagnosis by comparing the actions of the agents to each one of the instances of the satisfying action assignments.

10.2 Distributed Algorithms

To counter limitations of centralized coordination diagnosis methods, such as computationally expensive in practice in terms of communications and run-time, or relying on a single diagnoser and thus risk a single point of failure. We presented an empirical investigation of distributed diagnosis algorithms, using distributed CSP algorithms as a basis. Two algorithms compute all minimal diagnoses: SBT_OFF (suitable for systems where the coordination is static), and SBT_ON (for dynamic coordination). One algorithm guarantees a single diagnosis (ABT), and two algorithms utilize a local search approach and therefore do not guarantee any solution (DSA,DBA).

We evaluated the algorithms with real and simulated robots, and concluded that there is a trade-off between the effectiveness of the algorithms in terms of communication and computation and the correctness of the diagnosis that the algorithms produce. However, The ABT algorithm provides a surprise: It runs faster, communicates less, and provides better diagnoses than the local search algorithms—in contrast to lessons in the distributed CSP literature [Zhang *et al.*, 2005].

10.3 A Design Space for Social Diagnosis

We presented a novel design space for methods of social diagnosis. Each such method operates in two phases, and we presented alternative techniques for each phase. For the selection of the diagnosing agent, we have the following methods: (i) *pre-*

selected agent(s), relying on pre-selection by the designer; (ii) *fault detecting agent(s)*, letting the agents that detected the fault do the diagnosis; or (iii) *minimal queries diagnosing agent*, choosing the agent most likely to reduce communications (using distributed recursive modeling). In terms of computing the diagnosis, two choices are available: (i) Reporting: have all agents communicate their beliefs to the selected agents, (ii) Querying: allow the diagnosing agents to actively query agents as to the state of their beliefs, while minimizing the number of queries. For each one of the methods we evaluated the complexity in terms of their communications and computation overheads.

The combination between these methods defines a space of six algorithms of diagnosing a team of behavior-based agents. We empirically and systematically evaluate the different combinations to draw general conclusions about the design of diagnosis algorithms.

A first conclusion is that centralizing the diagnosis disambiguation task is critical in reducing communications. The second conclusion is that techniques where agents reason explicitly about the beliefs of their peers are computationally inferior (in runtime) to techniques where agents do not reason about others. However, such computation does result in a slight reduction in communications.

10.4 Large-Scale Teams

A key challenge in scaling up social diagnosis was the need to reduce both communication and inference runtime, where normally a trade-off between them existed. We presented novel techniques that enable scalability of social diagnosis in two ways. First, we used communications early in the hypotheses generation process, to stave off unneeded reasoning, which ultimately leaded to unneeded communications. Second we suggested diagnosing only a limited number of representative agents (instead of all the agents). We presented three techniques which utilize these ideas: (i) BEHAVIOR QUERYING using targeted queries to alleviate diagnostic reasoning; (ii) SHARED BELIEFS using light-weight behavior recognition to focus on beliefs that may be in conflict; and (iii) GROUPING the agents by their role and behavior and then diagnosing each group based on representative agents.

We evaluated these techniques comparing them to REPORTING and QUERYING algorithms. We showed that BEHAVIOR QUERYING and SHARED BELIEFS techniques offer only limited benefits in teams where the number of agents is scaled-up. However, BEHAVIOR QUERYING allows grouping the diagnosed agents along disagreement lines, thus allowing focused diagnosis of only representative agents from each group. This method proved highly scalable both in communications and in runtime. We also showed that using SHARED BELIEFS in small teams is better than QUERYING, and so combining it with GROUPING, which makes the diagnosis only against the representative agents, provides good results.

In addition, we showed that BEHAVIOR QUERYING alone is scalable in the number of behavior path hypotheses, and that SHARED BELIEFS alone is scalable in the number of beliefs. Thus the contribution of this thesis is by providing techniques scalable in the number of agents and also scalable in the knowledge size of the agents.

10.5 Future Work

This dissertation began to address the coordination diagnosis challenges we counted in Chapter 1.2. Much work remains for future research.

We formalized the coordination diagnosis problem in terms of model-based diagnosis (Chapter 4). However, once the diagnosis process isolates the abnormal agents, we should complete the diagnosis process by diagnosing every single abnormal agent separately. Thus we will be able find the faulty components of the abnormal agents. The combination of coordination diagnosis with single system diagnosis could be done either in two different stages as described here, or in one process by modeling the components of the agents in the coordination model of the team. One challenge is to investigate these approaches, and compare their runtime and communication overhead.

In Chapter 5 we presented diagnosis algorithms based on distributed CSP algorithms. Generally these algorithms have some drawbacks. The complete algorithms (guarantee minimal diagnosis) are not effective since they must go over through all the combinations of the assignments of the agents. The non-complete algorithms are faster but do not guarantee minimal diagnosis no correct. Thus another challenge is to find a complete algorithm which finds minimal diagnosis without checking all the assignments. Moreover, a heuristic is necessary to compute the diagnosis in increased order. This enables to compute the diagnoses less than a given cardinality, where, the cardinality is the size of the diagnosis.

In diagnosing disagreements in teams of situated-agents (Chapter 7), all methods find only the contradictions between agent beliefs, where the beliefs are derived directly from the hypothesized behavior paths. But in complex behavior-based control systems, chains of inference may lead from one belief to the next. Our system is currently not able to back chain through such inference pathways, and thus is unable to draw conclusions beyond the beliefs that immediately tied to pre-conditions and termination conditions. Addressing this challenge is also high in our list.

In this thesis we did not address probabilities on the type of the faults of the agents, nor on the type or amount of the faulty agents. For instance, we did not prefer a diagnosis which hypothesizes a single faulty agent over a diagnosis that hypothesizes multiple faulty agents. In addition, we did not address problems like how to repair the faults automatically or how to recover the system. This pioneer thesis laid the foundations to the diagnosis problem in teams of multi-agent systems. It will be a challenge to address such points in the future.

Bibliography

[Ardissono *et al.*, 2005] Liliana Ardissono, Luca Console, Anna Goy, Giovanna Petrone, Claudia Picardi, Marino Segnan, and Daniele Theseider Duprpé. Cooperative model-based diagnosis of web services. In *16th International Workshop on Principles of Diagnosis (DX 05)*, pages 125–130, 2005.

[Biteus *et al.*, 2006] Jonas Biteus, Erik Frisk, and Mattias Nyberg. Distributed diagnosis by using a condensed local representation of the global diagnoses with minimal cardinality. In *17th International Workshop on Principles of Diagnosis (DX-06)*, pages 23–30, Spain, 2006.

[Brodie *et al.*, 2001] M. Brodie, I. Rich, and S Ma. Optimizing probe selection fault localization. In *International Workshop on Distributed Systems: Operations and Management (DSOM-2001)*, 2001.

[Brodie *et al.*, 2002] M. Brodie, I. Rich, and S. Ma. Intelligence probing: A cost-effective approach to fault diagnosis computer networks. *IBM Systems Journal*, 41(3):372–385, 2002.

[Bryant, 1992] Randal E. Bryant. Symbolic Boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, 24(3):293–318, 1992.

[Bylander, 1990] T. Bylander. Some causal models are deeper than others. *Artificial Intelligence in Medicine*, 2:123–128, 1990.

[Cohen and Levesque, 1991] P.R. Cohen and H.J. Levesque. Teamwork. *Nous*, 25(4):487–512, 1991.

[Console and Torasso, 1991] Luca Console and Pietro Torasso. A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence*, 7(3):133–141, 1991.

[Console *et al.*, 1989] Luca Console, Daniele Theseider Dupŕe, and Pietro Torasso. A theory of diagnosis for incomplete causal models. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1311–1317, 1989.

[Console *et al.*, 1991] Luca Console, Daniele Theseider Dupre, and Pietro Torasso. On the relationship between abduction and deduction. *Journal of Logic and Computation*, 1(5):661–690, 1991.

[Cox and Pietrzykowski, 1987] Philip T. Cox and Tomasz Pietrzykowski. General diagnosis by abductive inference. In *IEEE Symposium on Logic programming*, pages 183–189, 1987.

[Daigle *et al.*, 2006] M. Daigle, X. Koutsoukos, , and G. Biswas. Distributed diagnosis of coupled mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 3787–3794, 2006.

[Darwiche, 1998] Adnan Darwiche. Model-based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research*, 8:165–222, 1998.

[Davin and Modi, 2005] John Davin and Pragnesh Jay Modi. Impact of problem centralization in distributed constraint optimization algorithms. In *Proceedings of Autonomous Agents and Multi Agent Systems (AAMAS-05)*, pages 1057–1066, 2005.

[Davis and Hamscher, 1988] R. Davis and W. C. Hamscher. Model-based reasoning: Troubleshooting. In A. E. Shrobe, editor, *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence*, pages 297–346, 1988.

[Davis, 1984] Randall Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24(1-3):347–410, 1984.

[de Kleer and Sussman, 1980] Johan de Kleer and Gerald Jay Sussman. Propagation of constraints applied to circuit synthesis. In *Circuit Theory and Applications*, volume 8, pages 127–144. 1980.

[de Kleer and Williams, 1987] J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

[de Kleer, 1986] Johan de Kleer. An assumption-based truth maintenance system. *Artificial Intelligence*, 28:127–162, 1986.

[Dellarocas and Klein, 2000] Chrysanthos Dellarocas and Mark Klein. An experimental evaluation of domain-independent fault-handling services in open multi-agent systems. In *Proceedings of the Fourth International Conference on Multiagent Systems (ICMAS-00)*, pages 95–102, 2000.

[Durfee, 2001] Edmund H. Durfee. Scaling up agent coordination strategies. *IEEE Computer*, 34(7):39–46, July 2001.

[Firby, 1987] R. James Firby. An investigation into reactive planning in complex domains. In *American Association for Artificial Intelligence (AAAI-87)*, pages 196–201, 1987.

[Fröhlich *et al.*, 1997] Peter Fröhlich, Iara de Almeida Mora, Wolfgang Nejdl, and Michael Schröder. Diagnostic agents for distributed systems. In *ModelAge Workshop*, pages 173–186, 1997.

[Gerkey *et al.*, 2003] Brian P. Gerkey, Richard T. Vaughan, and Andrew Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the International Conference on Advanced Robotics*, pages 317–323, Coimbra, Portugal, Jul 2003.

[Grosz and Kraus, 1996] Barbara J. Grosz and Sarit Kraus. Collaborative plans for complex group actions. *Journal of Artificial Intelligence Research*, 86:269–358, 1996.

[Hamscher *et al.*, 1992] Walter Hamscher, Luca Console, and Johan de Kleer, editors. *Readings in Model-Based Diagnosis.* Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[Hirayama and Yokoo, 2005] Katsutoshi Hirayama and Makoto Yokoo. The distributed breakout algorithms. *Artificial Intelligence*, 161(1-2):89–115, 2005.

[Horling *et al.*, 1999] Bryan Horling, Victor R. Lesser, Regis Vincent, Ana Bazzan, and Ping Xuan. Diagnosis as an integral part of multi-agent adaptability. Technical Report CMPSCI Technical Report 1999-03, University of Massachusetts/Amherst, January 1999.

[Horling *et al.*, 2001] Bryan Horling, Brett Benyo, and Victor Lesser. Using Self-Diagnosis to Adapt Organizational Structures. In *Proceedings of the 5th International Conference on Autonomous Agents*, pages 529–536, Montreal, June 2001. ACM Press.

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

[Kaminka and Bowling, 2002] Gal A. Kaminka and Michael Bowling. Robust teams with many agents. In *Proceedings of Autonomous Agents and Multi Agent Systems (AAMAS-02)*, pages 729–736, 2002.

[Kaminka and Frenkel, 2005] Gal A. Kaminka and Inna Frenkel. Flexible teamwork in behavior-based robots. In *American Association for Artificial Intelligence (AAAI-05)*, pages 1355–1356, 2005.

[Kaminka and Tambe, 1998] Gal A. Kaminka and Milind Tambe. What's wrong with us? Improving robustness through social diagnosis. In *American Association for Artificial Intelligence (AAAI-98)*, pages 97–104, Madison, WI, 1998.

[Kaminka and Tambe, 2000] Gal A. Kaminka and Milind Tambe. Robust multi-agent teams via socially-attentive monitoring. *Journal of Artificial Intelligence Research*, 12:105–147, 2000.

[Kaminka *et al.*, 2004] Gal A. Kaminka, Yehuda Elmaliach, Inna Frenkel, Ruti Glick, Meir Kalech, and Tom Shpigelman. Towards a comprehensive framework for teamwork in behavior-based robots. In *IAS-8*. 2004.

[Kaminka, 2006] Gal A. Kaminka. Detecting disagreements in large-scale multi-agent teams. *Journal of Autonamous Agents and Multi-Agent Systems (JAAMAS)*, 2006. To appear.

[Klein and Dellarocas, 1999] Mark Klein and Chris Dellarocas. Exception handling in agent systems. In *Proceeding of the Third International Conference on Autonomous Agents*, pages 62–68, May 1999.

[Kraus *et al.*, 1998] Sarit Kraus, Sycara Katia, and Amir Evenchik. Reaching agreements through argumentation: a logical model and implementation. *Artificial Intelligence*, 104(1–2):1–69, 1998.

[Lamperti and Zanella, 2003] Gianfranco Lamperti and Marina Zanella. *Diagnosis of Active Systems.* Kluwer Academic Publishers, 2003.

[Letia and Netin, 1999] I. A. Letia and A. Netin. Distributed diagnosis by agents using ontologies. In *Bar-Ilan Symposium on the Foundations of Artificial Intelligence (BISFAI-99)*, 1999.

[Letia *et al.*, 2000] I. A. Letia, F. Craciun, Z. Kope, and A. Netin. Distributed diagnosis by BDI agents. In *IASTED International Conference Applied Informatics,*, pages 862–867, 2000.

[Luca console, 1990] Pietro Torasso Luca console. Hypothetical reasoning in causal models. *International Journal of Intelligent System*, 5(1):83–124, 1990.

[Lynch, 1996] Nancy A. Lynch. *Distributed Algorithms.* Morgan Kaufmann, 1996.

[Mataric, 1998] M. Mataric. Behavior-based robotics as a tool for synthesis of artificial behavior and analysis of natural behaviorbehavior-based robotics as a tool for synthesis of artificial behavior and analysis of natural behavior. *Trends in Cognitive Science*, 2(3):82–87, 1998.

[Matsubara *et al.*, 1998] Hitoshi Matsubara, Ian Frank, kumiko Tanaka-Ishii, Ituski Noda, Hideyuki Nakashima, and Koiti Hasida. Automatic soccer commentary and robocup. In Minoru Asada, editor, *the Second RoboCup Workshop (RoboCup-98)*, pages 7–22, Paris, France, 1998.

[Meisels *et al.*, 2002] A. Meisels, E. Kaplansky, I. Razgon, and R. Zivan. Comparing performance of distributed constraints processing algorithms. In *Proceedings of Autonomous Agents and Multi Agent Systems (AAMAS-02)*, pages 86–93, 2002.

[Micalizio *et al.*, 2004] R. Micalizio, P. Torasso, and G. Torta. On-line monitoring and diagnosis of multi-agent systems: a model based approach. In *Proceeding of European Conference on Artificial Intelligence (ECAI 2004)*, volume 16, pages 848–852, 2004.

[Newell, 1990] Allen Newell. *Unified Theories of Cognition.* Harvard University Press, Cambridge, Massachusetts, 1990.

[Niemelä, 1996] I. Niemelä. A tableau calculus for minimal model reasoning. In *Proceedings of the fifth workshop on theorem proving with analytic tableaux and related methods*, pages 278–294, 1996.

[Olivetti, 1992] N. Olivetti. A tableaux and sequent calculus for minimal model entailment. *Journal of automated reasoning*, 9:99–139, 1992.

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

[Pencolé *et al.*, 2002] Y. Pencolé, M.O. Cordier, and L. Rozé. Incremental decentralized diagnosis approach for the supervision of a telecommunication network. *IEEE Conference on Decision and Control (CDC'02)*, pages 435–440, December 2002.

[Peng and Reggia, 1990] Yun Peng and James A. Reggia. *Abductive inference models for diagnostic problem-solving.* Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[Poutakidis *et al.*, 2002] D. Poutakidis, L. Padgham, and M. Winikoff. Debugging multi-agent systems using design artifacts: The case of interaction protocols. In *Proceedings of Autonomous Agents and Multi Agent Systems (AAMAS-02)*, pages 960–967, 2002.

[Provan, 2002] Gregory Provan. A model-based diagnosis framework for distributed systems. In *13th International Workshop on Principles of Diagnosis (DX'02)*, pages 16–25, Austria, 2002.

[Pynadath *et al.*, 1999] David V. Pynadath, Milind Tambe, Nicolas Chauvat, and Lawrence Cavedon. Toward team-oriented programming. In *Proceedings of the Agents, Theories, Architectures and Languages (ATAL'99) Workshop (to be published in Springer Verlag "Intelligent Agents V")*, pages 77–91, 1999.

[Reiter, 1987] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–96, 1987.

[Roos and Witteveen, 2005] Nico Roos and Cees Witteveen. Diagnosis of plan execution and the executing agent. In *Proceedings of the Third European Workshop on Multi-Agent Systems (EUMAS-05)*, pages 502–503, 2005.

[Roos *et al.*, 2001] N. Roos, A. ten Teije, A. Bos, and C. Witteveen. Multi-agent diagnosis: an analysis. In *Proceedings of the Belgium-Dutch Conference on Artificial Intelligence (BNAIC-01)*, pages 221–228, 2001.

[Roos *et al.*, 2002] Nico Roos, Annette ten Teije, André Bos, and Cees Witteveen. Multi-agent diagnosis with spatially distributed knowlege. In *Proceedings of the Belgium-Dutch Conference on Artificial Intelligence (BNAIC-02)*, pages 275–282, 2002.

[Roos *et al.*, 2003a] N. Roos, A. ten Teije, A. Bos, and C. Witteveen. Multi-agent diagnosis with semantically distributed knowledge. In *Proceedings of the 15th Belgium-Dutch Conference on Artificial Intelligence (BNAIC-2003)*, pages 259–266, October 2003.

[Roos *et al.*, 2003b] Nico Roos, Annette ten Teije, and Cees Witteveen. A protocol for multi-agent diagnosis with spatially distributed knowledge. In *Proceedings of Autonomous Agents and Multi Agent Systems (AAMAS-03)*, pages 655–661, July 2003.

[Roos *et al.*, 2004] Nico Roos, Annette ten Teije, and Cees Witteveen. Reaching diagnostic agreement in multi-agent diagnosis. In *Proceedings of Autonomous Agents and Multi Agent Systems (AAMAS-04)*, pages 1254–1255, 2004.

[Sachenbacher and Williams, 2004] Martin Sachenbacher and Brian Williams. Diagnosis as semiring-based constraint optimization. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence ECAI-2004*, pages 873–877, 2004.

[Sampath *et al.*, 1995] M. Sampath, R. Sengupth, S. Lafortune, K. Sinnamohideen, and D.C. Teneketzis. Failure diagnosis using discrete-event models. *IEEE Transactions on Automatic Control Systems Technology*, 40(9):1555–1575, 1995.

[Sampath *et al.*, 1996] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D.C. Teneketzis. Failure diagnosis using discrete-event models. *IEEE Transactions on Control Systems Technology*, 4(2):105–124, 1996.

[Scerri *et al.*, 2005a] Paul Scerri, Joseph Andrew Giampapa, and Katia Sycara. Techniques and directions for building very large agent teams. In *2005 International Conference on Integration of Knowledge Intensive Multi-Agent Systems, KIMAS'05: Modeling, Exploration, and Engineering*, pages 79–84, April 2005.

[Scerri *et al.*, 2005b] Paul Scerri, Elizabeth Liao, Y. Xu, Michael Lewis, G. Lai, and Katia Sycara. Coordinating very large groups of wide area search munitions. *Theory and Algorithms for Cooperative Systems*, 2005.

[Scerri *et al.*, 2005c] Paul Scerri, Régis Vincent, and Roger Mailler. *Coordination of Large-Scale Multiagent Systems*. Springer, October 2005.

[Shannon, 1948] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–693, 1948.

[Skiena, 1990] Steven Skiena. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Addison-Wesley, 1990.

[Tambe *et al.*, 2005] M. Tambe, E. Bowring, H. Jung, G. Kaminka, R. Maheswaran, J. Marecki, P. J. Modi, R. Nair, S. Okamoto, J. P. Pearce, P. Paruchuri, D. Pynadath, P. Scerri, N. Schurr, and P. Varakantham. Conflicts in teamwork: hybrids to the rescue. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS–05)*, pages 3–10, 2005.

[Tambe, 1997] Milind Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[Tambe, 1998] M. Tambe. Implementing agent teams in dynamic multi-agent environments. *Applied Artificial Intelligence*, 12(2-3):189–210, 1998.

[Torasso and Torta, 2003] Pietro Torasso and Gianluca Torta. Computing minimum-cardinality diagnoses using OBDDs. *Advances in Artificial Intelligence (lecture notes in artificial intelligence)*, 2281:224–238, 2003.

[Wotawa, 2004] F. Wotawa. *e-Environement: Progress and Challenge*, volume 11 of *Research on Computing Science*, pages 334–347. México, 2004.

[Yokoo and Hirayama, 2000] Makoto Yokoo and Katsutoshi Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):185–207, 2000.

[Yokoo *et al.*, 1998] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Trans. Knowl. Data Eng.*, 10(5):673–685, 1998.

[Zhang *et al.*, 2005] Weixiong Zhang, Guandong Wang, Zhao Xing, and Lars Wittenburg. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence*, 161(1-2):55–87, 2005.

[Zivan and Meisels, 2006] Roie Zivan and Amnon Meisels. Concurrent search for distributed CSPs. *Artificial Intelligence*, 170(4–5):440–461, 2006.