A MUTUAL ASSISTANCE PROTOCOL FOR AGENT TEAMWORK

by

Narek Nalbandyan

B.Sc., Yerevan State University, 2008

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
MATHEMATICAL, COMPUTER, AND PHYSICAL SCIENCES
(COMPUTER SCIENCE)

THE UNIVERSITY OF NORTHERN BRITISH COLUMBIA

September 2011

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-87576-6

Our file Notre référence
ISBN: 978-0-494-87576-6

Abstract

This thesis proposes a novel protocol for incorporating helpful behavior into multiagent teamwork. In the proposed protocol, called the Mutual Assistance Protocol (MAP), an agent can use its own abilities and resources to advance a subtask assigned to another agent. The helpful act is performed only when the two agents jointly determine that it is in the interest of the team. The underlying design principle is that each agent assesses the team impact of changes in its own local plan. The distributed decision is reached through a bidding sequence similar to the Contract Net Protocol. The helpful act may consist in performing an action or in granting resources. The advantages of MAP over protocols that use unilateral help decisions are demonstrated through simulation experiments, using varying levels of mutual awareness in the team, dynamic disturbance in the environment, communication costs, and computation costs.

Table of Contents

List of Figures

Acknowledgements

I am cordially grateful to my supervisor, Dr. Jernej Polajnar, for his encouragement, patience, advice, and support during my study, which allowed me to gain an understanding of the subject and reach this stage. I am also thankful for his faith in this work, considering the somewhat difficult circumstances in which it was written.

I am heartily thankful to Dr. Desanka Polajnar and Dr. Iliya Bluskov for their help and for serving in my Supervisory Committee.

I would like to express my gratitude to Dr. Tamara Polajnar, as well as to my colleagues Omid Alemi, Arber Borici, and Ashton Fedler, for their input on the development of this Thesis.

I am also thankful to all of those who assisted me in one way or another during the completion of this Thesis.

This Thesis is dedicated to my family

Nvard Danielyan, Levon Nalbandyan,
and Arevik Nalbandyan

who selflessly supported me in every stage of my life,

enabling such a study to happen today.

Chapter 1

Introduction

Many tasks in human society can only be accomplished through *teamwork*, i.e., organized, coordinated activity of a group of individuals with common goals directed towards a particular task. Teamwork is increasingly important for solving a variety of complex tasks when no individual is able to accomplish the task alone (e.g., performing a surgery) or when accomplishing the task alone would be highly inefficient (e.g., harvesting a large field). A team structure usually involves a number of differentiated roles and relationships between them, such as hierarchy or peer cooperation rules. The teamwork participants work on different constituents of the collective task and may perform mutually supportive collaborative actions towards achieving the collective goal.

Teamwork has also become a major research topic in multiagent systems (MAS) [Levesque et al., 1990, Cohen and Levesque, 1991, Grosz and Kraus, 1996, Sycara and Lewis, 2004, Dunin-Keplicz and Verbrugge, 2010]. Agents are autonomous intelligent entities, delegated to solve problems, and capable of social interaction. They have been studied, in the context of several disciplines, for about three decades. Over the last decade, they have evolved into a mainstream computing technology [Wooldridge, 2009]. With the rise of networking and distributed computing, the central interest of

the agent research community became the study of multiagent systems, in which a group of agents engage in cooperative or competitive interactions [Wooldridge, 2009, Shoham and Leyton-Brown, 2009]. Within that context, a variety of formalizations have been developed to model MAS teamwork [Levesque et al., 1990, Wooldridge and Jennings, 1994, Grosz and Kraus, 1996, Aldewereld et al., 2004, Dunin-Keplicz and Verbrugge, 2010]. In addition, several platforms have been built to facilitate agent teamwork, such as GRATE [Jennings et al., 1992], STEAM [Tambe, 1997], CAST [Yen et al., 2001], and Cougaar [Helsinger and Wright, 2005]. The application systems based on agent teams include the MokSAF team planning system for time-critical tasks [Payne et al., 2000], the agent-based planning team training platform [Mountjoy and Ram, 2003], the system for human-agent teamwork in space applications [Sierhuis et al., 2003], and several systems for agent rescue teams [Hill et al., 2003, Marecki et al., 2005].

An essential aspect of MAS research and development is the design of protocols that specify how agents communicate and interact [Greaves et al., 2000, Paurobally et al., 2003, Dunn-Davies et al., 2005]. The rules and patterns of mutual interaction are often intuitively understood in human encounters, but require detailed and rigorous specification in MAS. Once an interaction protocol for a particular purpose has been precisely formulated, it can be formally studied with respect to correctness, efficiency, and properties of specific interest. It can then be optimized, standardized, and incorporated into MAS software development libraries and platforms. The resulting benefits include the interoperability between independently developed agents, a wider variety of design-time or even run-time choices among alternative interaction patterns, and a higher general level of architectural clarity, correctness, and efficiency of MAS software. There have been a variety of agent protocols developed for self-interested agent interactions such as auctions, negotiation, bargaining, or coordinated use of

resources Smith [1980], Foundation of Intelligent Physical Agents [2001a, 2000], Fiat et al. [2007], Ito et al. [2007]. The most successful and the most widely used among them has been the Contract Net Protocol (CNP) [Smith, 1980]. CNP is designed for distribution of tasks among agents in a manner resembling contract tenders. It has been used in a variety of application domains, such as open electronic marketplaces [Dellarocas and Klein, 1999]. The properties of CNP and its optimizations have been studied in the works by Sandholm [1999, 1993]. CNP has been standardized by the Foundation for Intelligent Physical Agents (FIPA) [Foundation of Intelligent Physical Agents, 2001b] and implemented as a library in the agent development package JADE [Java Agent Development Framework, 2004].

In the domain of agent teamwork, protocol research has mainly focused on team-wide issues such as the team formation, the assignment of roles and subtasks to agents, or the development of team-level plans, often building upon early work in the area of cooperative problem solving [Wooldridge and Jennings, 1994, 1999, Durfee, 1999]. One observation that motivates the research presented in this thesis is that the quality and efficiency of teamwork depend not only on its global aspects such as team structure, organization, resources, and planning, but also on smaller-scale collaborative practices within the team. It has been observed in scientific studies (e.g., [LePine et al., 2000]), and commonly accepted in management practices, that the capacity of team members for direct mutual assistance is an important ingredient of success in human teamwork. Consistent with this observation, there has been a growing interest in the study of helpful behavior in agent teamwork [Itoh, 1991, Miceli et al., 1994, Yen et al., 2004, Cao et al., 2005, Fan et al., 2005, Kamar et al., 2009, Polajnar et al., 2011]; however, that research interest has not, to the best of our knowledge, resulted in the formulation of specific protocols for helpful behavior.

In this thesis, we introduce a novel protocol, called the Mutual Assistance Protocol

(MAP), that incorporates helpful behavior into agent teamwork. In its underlying teamwork model, each team member works on a subtask with specified time and resource limits, for which it autonomously constructs its local plan. In a helpful act, one agent uses its own abilities and resources to help advance the subtask of another agent. The helpful act is performed only if the two agents jointly determine that it is in the interest of the team as a whole, using their individual beliefs about the state of the environment and the state of activities in the rest of the team. The message exchanges in MAP are similar as in CNP, but the decision criteria are based on team interest rather than individual self interest. (In fact, individual interest is not present in the model; the team interest relates to the achievement of team's objectives, not to collective social welfare based on individual interests.)

The purpose of MAP is to enable team members to respond to arising difficulties through direct mutual assistance, without the need to raise those difficulties at the global team level. The design of MAP avoids assumptions about the global team organization, its degree of centralization, or the techniques used for global plan construction, subtask assignment, and resource allocation. However, the design of MAP does assume that the global decisions in effect are sound and suitable for the team's task. MAP is not designed to help overcome team-level problems of structure or strategy, even if helpful behavior may alleviate such problems in the short term. Because of this, we regard MAP as a 'secondary' protocol, designed to support smoother teamwork in the presence of moderate challenges, while challenges of higher magnitude may require the use of 'primary' protocols for reorganization or replanning at the global level.

The design philosophy of MAP is to choose the decision mechanism for helpful behavior with a view towards the expected accuracy of agent beliefs on which the decisions are based. While we assume that agents in the team communicate truth-

fully, their beliefs are usually imperfect. However, different beliefs of the same agent may differ in accuracy, forcing the protocol designer to ponder which and whose beliefs to use in the decision criteria. The design principle adopted in our case is that, since we assume that the primary team organization is viable, the designer can view an individual belief that is highly relevant to the agent's role in the primary team organization as being credible enough to serve as a basis for decisions on secondary behavior. Applying this principle to our MAS model, we note that each agent constructs its own local plans, and therefore uses its individual beliefs to evaluate the team impact of each candidate plan. Accordingly, for deciding whether a helpful act should occur, we choose a distributed mechanism in which each agent only assesses the team impact of changes in its own local plan. We provide the specification of MAP in two cases: when the helpful behavior is expressed by performing an action, and when it is expressed by providing resources. In addition, we discuss two possible variations of MAP;one of them involves an alternative help-seeking mechanism, the other an alternative sequence of interactions in the help transaction.

For comparison, we construct two protocols in which the decision about help is made unilaterally, based on the beliefs of a single agent. In one of them the decision about whether to help is based on probabilistic beliefs of the helper alone (similar to the principle employed in [Kamar et al., 2009]); in the other, the decision is based on the probabilistic beliefs of the requester alone. Also included for comparison is a computation without a helpful behavior. We analyze the complexity of the four protocols with respect to computation and communication costs. We also compare the four protocols in a series of simulation experiments, using as the simulation test-bed a variation of the Colored Trails game [Grosz et al., 2004], which is a publicly available research test-bed for examining decision making in group of agents or people.

We present a set of simulations showing how the performance of an agent team

depends on each of the following four quantities: the level of mutual awareness among the team members; the level of dynamic disturbance in the environment; the cost of communication; and the cost of computation. In this context, we compare the performance of the four approaches to helpful behavior. In most cases, all three protocols for helpful behavior produce significant improvements over the computation without helpful behavior. The experiments show the superiority of MAP over the other protocols, especially when the agents' knowledge about the rest of the team decreases. MAP strongly outperforms both unilateral protocols except in situations where individual agents have a near-perfect knowledge about the rest of the team; the latter type of situation is uncommon in practice and often costly to achieve in distributed architectures.

The rest of this Thesis is structured as follows: Chapter 2 provides the background work, Chapter 3 presents the rationale for teamwork protocols for helpful behavior, Chapter 4 presents The Mutual Assistance Protocol (MAP), Chapter 5 provides the comparative performance evaluation of MAP and other protocols, and Chapter 6 presents the conclusions and future work.

Chapter 2

Related Work

This chapter contains the necessary background in areas of multiagent systems, cooperation and collaboration, teamwork in multiagent systems, agent interaction protocols, helpful behavior in teamwork, and expert teamwork, with the focus relevant to my thesis.

2.1 Multiagent Systems

There is no universally accepted definition of what exactly an agent or a multiagent system (MAS) is. Shoham and Leyton-Brown [2009] describe multiagent systems as "systems that combine multiple autonomous entities, each having diverging interests or different information or both." Wooldridge [2009] defines multiagent systems as "systems composed of multiple interacting computing elements, known as *agents*", where an agent is "a computer system situated in some environment and capable of autonomous action in this environment in order to meet its delegated objectives ".

Wooldridge and Jennings [1995] suggest that in order to satisfy their design objectives the agents need to be *proactive*, i.e., able to take initiatives in performing goal-directed actions; *reactive*, i.e., able to perceive the environment and react to its changes; and

social, i.e., able to interact with other agents and humans.

While in this work we are primarily interested in artificial agents, we find it useful to take the inclusive view of multiagent systems, allowing, for instance, the term to apply to a team consisting of humans and agents. Thus, while all definitions allow agents to be artificial entities, the same inclusive view also allow them to be humans or other living organisms.

An agent gets sensory inputs from the environment and produces as output actions that can influence the environment. An agent also has an internal state, which evolves depending on the sensory input and influences the actions of the agent. The environmental states can be represented using a utility function that assigns a real number to different states of the environment, letting the agent judge how desirable each state is for the agent.

In the last twenty years, researchers have expressed particular interest in practical reasoning agents. The practical reasoning agents have unique mental states, which help them to decide what to do and how to accomplish it. The most popular framework for constructing this type of agents is the *Belief-Desire-Intention (BDI)* framework, coming from the philosophical work of Bratman [1987]. In that model, the agent relies on perception to form *beliefs* (which may or may not be true) about the environment and about other agents. The agent uses these beliefs to construct the *desires*, which are states of the environment that the agent would like to achieve. The most suitable desires, which are chosen through deliberation, become *intentions*. The agent does not have any commitment towards desires, and they can be even mutually inconsistent or unachievable, but the agent should have some level of commitment towards its intentions and should consider them possible. There are many systems implemented using the BDI model. One of the first and the best known is the Pro-

cedural Reasoning System (PRS), constructed by Georgeff and Lansky [1987]. Other such systems include dMARS, AgentSpeak, JADEX, and Jason (see e.g., [Wooldridge, 2009]).

Rao and Georgeff [1995] formlize BDI reasoning using modal logic. Modal logic can express statements such as "necessarily true" or "possibly true" [Hughes and Cresswell, 1996, Blackburn et al., 2001]. However, the term "modal logic" is used more widely to cover a group of logics with similar rules, which are derivatives of the classical modal logic. Examples of such logics are the temporal logic or the epistemic logic. In temporal logic, one can express temporal notions such as p is 'henceforth true' or becomes 'eventually true', while epistemic logic allows to express epistemic notions such as what agent a knows, or what agent a knows about the knowledge of agent b.

There are many agent-based applications that are successfully used in the real world, such as the OASIS air traffic management system [Ljungberg and Lucas, 1992], the CIDIM power distribution system, based on ARCHON architecture [Jennings, 1994], and the FIRMA resource management project [Downing et al., 2001].

2.2 Multiagent Cooperation and Collaboration

In multiagent systems, agents may interact in different ways, involving cooperation, collaboration, competition, or combinations of these. In this thesis, I will be mainly interested in cases where agents cooperate and collaborate in order to achieve a common goal.

In order to work together, agents need to coordinate their activities [Wooldridge, 2009]. The coordination is necessary for synchronizing the agents' actions and avoiding extraneous activities. However, coordination in agent interaction does not imply

cooperation. Cooperation requires more than coordination from agents, and there is a difference between coordinated action that is not cooperative, and cooperative coordinated action. The example showing that is the scenario presented below cited from Searle [1990].

> As a result of a sudden downpour in the park, a group of people run to a tree in the middle of the park because it is the only available source of shelter. This may be coordinated behavior, but it is not cooperative action, as each person has the intention of avoiding becoming wet. But when the people are dancers, and the choreography calls for them to converge on a common point (the tree), this is cooperative action, although the individuals are performing exactly the same actions as before. The difference is that in latter case they each have the aim of meeting at the central point as a consequence of the overall aim of executing the dance.

This example illustrates that not every coordinated action is a cooperation. Cooperation involves several different types of activities, such as task sharing, information sharing, and dynamic coordination of multiagent activities [Wooldridge, 2009].

For our purposes, we will use the term "cooperation" to describe working together in the broadest sense, while "collaboration" is a more restrictive form of cooperation which implies commitment to a shared goal, usually the accomplishment of the action that the agents are trying collectively to accomplish. Self-interested individuals will cooperate if everyone benefits, even though their goals may differ, while during collaboration individuals must cooperate and have shared goals.

There are at least two main distinctions between cooperation (and collaboration) in multiagent systems and 'traditional' distributed systems [Wooldridge, 2009], specified below:

- In multiagent systems, agents can be designed by different people who have different goals. As a result, the agents may have no shared goals, which may require them to function strategically in order to obtain their targeted outcome.
- As agents act autonomously and make decisions at run-time, they must be able to coordinate their actions and cooperate with other agents dynamically. In contrast, in traditional distributed systems, units typically coordinate and cooperate according to the rules developed at design time.

The need for agent cooperation for problem solving has been recognized in late 1980s with the work of Durfee et al. [1989], where the authors explore the cooperation of agent-like entities, which have unique expertise and can solve problems. As there may be problems which no individual agent can solve, or solving the problem collectively would bring them more benefit (such as less usage of resources or more confidence regarding the solution quality), agents may decide to cooperate.

One of the most widely used approaches towards agent collaboration has been the use of joint intentions [Wooldridge, 2009]. Intentions provide stability and predictability that are essential to act in a changing environment [Cohen and Levesque, 1991, Levesque et al., 1990, Cohen and Levesque, 1990]. Being part of a collaborative process implies that, in addition to having individual intentions towards a certain goal, agents also must have certain intentions to commitments towards the other members. Such a mental structure provides stability to the collaborative activity.

2.3 Teamwork

Teamwork is the collaboration of individual agents towards accomplishing a particular task. In order to act as a team, the agents need to have committments to

shared goals and be in particular mental states while performing their actions. The study of agent teamwork is an active research area, and there have been many formal approaches towards the formalization of the semantics of agent teamwork. The examples include the fundamental works by Levesque et al. [1990], Cohen and Levesque [1991], Wooldridge and Jennings [1994], Grosz and Kraus [1996], and more resent works by Sycara and Lewis [2004], Aldewereld et al. [2004], Brzeziński et al. [2005], Dunin-Keplicz and Verbrugge [2010]. In addition, several platforms have been built to facilitate agent teamwork, such as GRATE [Jennings et al., 1992], STEAM [Tambe, 1997], CAST [Yen et al., 2001], and Cougaar [Helsinger and Wright, 2005]. The application systems based on agent teams include the MokSAF team planning system for time-critical tasks [Payne et al., 2000], the agent-based planning team training platform [Mountjoy and Ram, 2003], the system for human-agent teamwork in space applications [Sierhuis et al., 2003], and several systems for agent rescue teams [Marecki et al., 2005, Hill et al., 2003], to name a few.

The agents may recognize the need for collaboration during the execution of their tasks. This requires agents to have capabilities for deciding whether to form teams at run-time. Wooldridge and Jennings [1994, 1999] present a four-stage model of collaborative problem solving (CPS) and formalize it by expressing it in multi-modal logic. The four stages of the model are as follows:

1. Recognition

   CPS starts when an agent that has a goal recognizes the need for collaborative action related to that goal. An agent may recognize the need for collaboration for several reasons, such as the beliefs about inability or inefficiency to achieve the goal alone. However, these beliefs are not enough to initiate the collaboration. In order to have a capacity for collaboration regarding an agent's goal,

the agent must also have beliefs about the existence of a group of agents that can achieve the goal.

2. Team Formation

   In this stage, the agent that recognized the opportunity for collaboration requests others to collaborate. Other agents deliberate about the request, and if they agree to collaborate, they together form a team with a certain commitment to collective action. The team then agrees about the way towards achieving the goal. As the agents are rational, they will not agree to form a team if they do not believe that the goal is achievable.

3. Plan Formation

   Having beliefs about the existence of solution to achieve the goal, the team chooses an action that will take the group at least one step 'closer' to the goal. As there may be different such actions, the team chooses the best one through negotiation.

4. Team Action

   In this stage, the team executes the agreed plan of joint actions. The agents follow a certain convention to keep the relationship with other agents during the plan execution.

In their work, Cohen and Levesque [1991] and Levesque et al. [1990] specify the necessary conditions of the mental states of the agents in the team, using the notion of joint intentions. In their model, the authors show that the team of agents working on some common goal should treat that goal as a *weak achievement goal*, i.e., every agent in a team should consider the possibility that the other team member may have discovered that the goal is either achieved, unachievable, or irrelevant, and is on its

way of making that fact commonly known. If any agent in the team comes to the conclusion that the goal is accomplished, not achievable, or irrelevant, it should drop the goal, but before finishing its job, it should make the fact commonly known.

Among several other researchers, Smith and Cohen [1995] developed a semantics for an agent communications language, which is essential for team agents in order to communicate and share their beliefs and intentions. The authors showed that the establishment of semantics for an agent communications language can be done on the assumption that inter-agent communications form a task oriented dialogue, which is used by agents to build, maintain, and disband teams. The agents perform these activities through actions of communicative acts, called *speech acts.* The authors constructed basic and complex speech acts, such as *Assert, Request, Refuse,* etc., based on joint intentions theory, and show how agents can form and disband teams using series of speech acts. Tambe [1997] integrated the properties of mental states of the agents forming a team, as well as the speech acts necessary for the agent communication, demonstrating a flexible and reusable agent architecture via an implemented candidate STEAM. STEAM is based on joint intentions theory and captures concepts of team synchronization, constructs for monitoring joint intentions and repair, and decision-theoretic communication.

2.4 Agent Protocols

In order to accomplish certain tasks, agents engage in interactions, which may be sequences of actions and messages following a higher-level structure. Such a higher-level structure referring to a certain task is called an agent protocol [Paurobally et al., 2003, Paurobally and Cunningham, 2002, Miller and McBurney, 2008].

Agent protocols are essential in all but the most basic agent cooperative interactions, as they provide set of rules controlling the interaction [Greaves et al., 2000, Dunn-Davies et al., 2005]. They include agent interaction protocols and agent communication protocols. Agent interaction protocols are high-level protocols that prescribe what the agents should communicate to each-other when performing a certain type of task. They rely on communication protocols that regulate how agents communicate. Two well known standard agent communication protocols are KQML/KIF [Mayfield et al., 1996] and FIPA-ACL [Foundation of Intelligent Physical Agents, 1997]. In this thesis we are mainly interested in agent interaction protocols.

As agents may interact in a variety of circumstances, having a universal agent protocol is not realistic [Dunn-Davies et al., 2005], and different agent cooperations in different domains require specific protocols. Research on different types, properties, and optimizations of agent interaction protocols includes such as [Ball and Butler, 2006, Chen et al., 2007, Smith, 1980].

There have been a variety of agent protocols developed for self-interested agent interactions [Smith, 1980, Foundation of Intelligent Physical Agents, 2001a, 2000, Fiat et al., 2007, Ito et al., 2007]. One of the most successful and widely used among them has been the Contract Net Protocol (CNP) [Paurobally et al., 2004], originally developed by Smith [1980]. Smith's inspiration came from the method that companies use when putting contracts out to tender [Wooldridge, 2009]. The Contract Net Protocol is a high level protocol, originally designed for the cooperation of nodes during the distributed problem solving process. The main purpose of CNP is achieving a balance through task sharing, where the nodes with workload are able to find idle nodes in the net to perform the tasks. The nodes achieve such balance through contracting.

The original CNP algorithm consists of four main steps:

1. Task Announcement

   The node that creates the tasks or realizes that it needs help, advertises the task to other nodes through task announcement, and after announcing it becomes the manager of that task. Depending on the specific capabilities of other nodes, the manager may send the announcement to all other nodes with general broadcast, to a subset of nodes with limited broadcast, or to one single node with direct message.

2. Bidding

   The nodes in the net receive the task announcement, evaluate it according to the expertise the task requires, the price, and possibly some other factors. If the node realizes that it is suitable for the task, it submits a bid. The bid may specify the capabilities of the node which are important for the completion of the task, and possibly some other parameters.

3. Awarding

   The manager may receive many bids from different nodes for one task announcement, and based on the specifications of the bids, chooses the most appropriate candidate(s) to execute the task. The manager then informs the successful bidders through an *award* message. The manager rejects all other bids.

4. Executing

   The selected node(s) start the execution of the task. These nodes are called contractors of the task. After the completion of the task the contractors report the results to the manager. A node can be a manager of one task and a contractor of another task at the same time.

Along with the development of the Multiagent Systems, the Contract Net Protocol had to be modified to reflect approaches in the way current agents are designed [Weiss, 2001].

In addition to the early work on cooperative problem solviing, the Contract Net Protocol has been extensively used for cooperation among self interested agents. The actions of self interested agents are motivated by increasing the individual utility value to the agent. The modern Contract Net Protocol has additional utility margin calculations during the process of evaluation of the announcement. The agent evaluates the benefit associated with potential contract and the cost for executing the task in the contract, calculating the marginal benefit. Calculating the benefits from all available task announcements, the agent uses its rationality and bids for the one with highest value [Sandholm, 1999, 1993].

As shown in the steps of the algorithm and the analysis above, the interaction between agents in CNP is based on competitive negotiations by using contracts. Thus, though the CNP itself is designed for cooperative distributed problem solving purposes, the individual agents in the net are self-interested, meaning that the final choice may be the best for the manager and the contractor, but not for the group as a whole [Weiss, 2001]. Based on this property of self-interested agents in CNP, there have been several application areas researched and presented, such as the open electronic marketplaces [Dellarocas and Klein, 1999], where agents can buy and sell goods.

CNP has also been widely tested and analyzed in Multiagent Systems, both from the semantical and state machine model's perspective [Paurobally et al., 2004, Itabashi et al., 2002]. It has a number of advantages, such as the decentralized nature of decision making and the simplicity of the algorithm. In addition, CNP has been implemented in several agent development packages such as JADE [Java Agent De-

velopment Framework, 2004].

However, the modern approach to CNP may lead to a relevant problem, where the agents will not accept new task announcements if the marginal benefit of the new task is lower compared to their current available tasks. This situation may lead to the case where the whole system is stuck, whereas the individual agents have the maximum benefits. [Weiss, 2001].

There is also another bothering problem with the usage of CNP in Multiagent Systems, which is that the agents are not always truthful and may lie, if it would increase their benefit. Since the agents are self-interested and try to maximize their utility, such behaviors are possible.

Research on development of protocols for agent collaboration has been mainly focused on planning of the global structure of the team. The strategies for task sharing, result sharing, and global planning of the team are addressed in the work by Durfee [1999]. A general model for team formation is presented in the works by Wooldridge and Jennings [1994, 1999]. However, the development of protocols for achieving efficient inter-agent collaboration in cases when the global team structure is already planned, has received less attention. Some of the steps towards building such protocols is included in the work by and Kamar et al. [2009].

2.5 Helpful Behavior in Agent Teams

Both science and human experience indicate that towards designing efficient human teamwork, an essential factor is the helpful behavior among the team members. A team member performs helpful behavior by assisting another team member if it executes or gives information about a part of the task that is assigned to the other

team member. Helpful behavior in agent teams can be important as well, and there exists a research interest in modeling helpful behavior in artificial agents [Itoh, 1991, Miceli et al., 1994, Yen et al., 2004, Cao et al., 2005, Fan et al., 2005, Kamar et al., 2009]. Although agents are *designed* with certain capabilities and for certain roles, the need for helpful behavior in agent teams can arise for several reasons [Polajnar et al., 2011]. Firstly, the agents are usually situated in environments where potential faults and unexpected events may happen, putting the agents in situations not anticipated by their designers. Moreover, as the design of agents can often be costly, in some domains it may be preferred to design agents with certain standard capabilities in the domain of their design objectives, so that they can be reused during the execution of similar tasks. On the other hand, providing agents with potentially necessary capabilities may often be impractical, as the agents may have physical or computational components that cannot be cost-effectively transferred to every team member that could possibly need them. Therefore, in many environments an agent team, even though designed for specific application, may still need mechanisms for mutual assistance or dynamic reorganization.

The dynamic reorganization of agent roles is currently used in several self-healing systems, designed to provide fault tolerance [Dashofy et al., 2002]. The dynamic reorganization may be efficient when the changes in the environment are persistent. However, when the environment changes have intermittent behavior and are unpredictable, dynamic reorganization of agent roles does not solve the problem [Polajnar et al., 2011].

Cao et al. [2005] use the shared mental models to develop a formal model of proactive helpful behavior. The model provides means for proactive helpful behavior in cases when a team member fails its task and when a team member needs to achieve the conditions necessary for performing its task. The model enables the agents to

identify the help needs of the team members and perform actions to satisfy the team members' needs if they can.

Kamar et al. [2009] observe that, even though the team members have an incentive to help each-other because of their commitments to the shared goal, they still need to deliberate about the decision whether to help. Helpful actions are associated with some costs for the helping agent, which may result in costs for the team activity as a whole. The team costs may be of different types, such as the spent resources on execution of the help or communication, lost opportunities to perform other activities. The authors develop a general model for evaluating the potential benefit to the team related to performing a helpful action. The authors develop their model based on local probabilistic beliefs of the team members. The team members use these local beliefs to reason about the probabilities of the potential help needs of other team members and the potential team benefits or losses associated with providing such helps. However, these beliefs are based on local reasoning and not guaranteed to be accurate.

An approach towards decentralized decision making for the helpful action has been mentioned in the work by Polajnar et al. [2011], where the authors discuss the need for empathy in artificial agents and provide a simulation experiment where the team members achieve better results by performing empathy-driven helpful behavior in a decentralized manner.

2.6 Expert Teamwork

Many human teams are characterized by the property that individual members of the team have unique expertise that they contribute to the team [Cannon-Bowers

et al., 1993, Mohammed and Dumville, 2001, Cooke et al., 2000, Hoffman et al., 1995]. Expertise is a specialist knowledge or skill that cannot be easily transferred to another team member. Therefore, the decisions of expert members in teams are accepted, and the expert members are given autonomy of decision making. For our purposes, we call this kind of teams *expert* teams[1]. A good example of a human expert team is the surgical team, which typically consists of the main surgeon, an assisting surgeon, an anesthesiologist, and various supporting roles; each role involves specialized knowledge, as well as predefined rules and patterns of collaboration with other roles in the team. Besides role specialization, another general property of expert teamwork is that team decisions are based on expertise of specialized members. The decision mechanisms therefore must balance commitment to joint goals with the necessary expert autonomy. Cooke et al. [2000] observe that in human expert teams an essential factor for success is the limited overlapping knowledge among the team members. Such an overlap provides means for mutual assistance in case of difficulties, as well as certain predictability of the team members' decisions.

Expert teamwork has been the subject of research interest in multiagent settings as well [Polajnar et al., 2011, Singh, 1991a]. The incentives of such studies lie in the fact that many multiagent systems, too, often have specialized members in teams, and some of the members may have unique skills and abilities. Some application-specific research points on that too [Polajnar et al., 2008]. However, because of more complex properties of expert teams, the collaboration mechanisms among expert agents have been developed less compared to the collaboration mechanisms for homogeneous agent teams.

Polajnar et al. [2008] suggest that, depending on the team collaboration model,

[1]In our work, the term *expert* in phrase expert 'team' is distinct from the term *expert* in the phrase 'expert systems'. By saying *expert*, we mean the individual unique expertise of the team member

the communication among the members of an expert agent team can be achieved through both a passive environment and direct message exchange, depending on the purpose of the communication. Communication through environment is relevant for collaborative solving of complex problems involving distributed expertise. An example of such environment is the blackboard architecture, in which the team members can post or subscribe to different categories and having a flexibility of synchronous as well as asynchronous communication [Corkill, 1991]. The expert team members can also communicate client-server interactions that are properly designed as bidirectional message-passing transactions and should not be done in the environment.

As the members of an expert agent team have individual unique expertise, they know how to do certain things. This *know-how* [Singh, 1999, 1991b,a] refers to procedural knowledge - how to achieve or maintain some states of affairs - which is different from factual knowledge. Singh [1999] developed the theory of procedural knowledge, and showed that in many situations know-how is equally important as the knowledge of facts (know-that), and that know-how cannot be reduced to know-that. Thus, the capabilities of expert agents are represented by the combination of their knowledge and know-how.

Chapter 3

Designing a Protocol for Helpful Behavior

This chapter highlights the need for protocols that provide mechanisms for helpful behavior in agent teamwork. Section 3.1 describes the need for helpful behavior in agent teams, Section 3.2 presents the approach we use towards designing MAP.

3.1 The Case for Helpful Behavior in Agent Teams

The interest in human teamwork, its mechanisms of collaboration, and the techniques to improve its efficiency, has been steadily increasing in recent years, both in the realm of scientific studies and in practical management, to the point that most job interviews now directly address candidates' teamwork skills.

In human teamwork, the success of a team project depends on many factors. The team must have the necessary expertise and resources, clear formulation of tasks, an adequate structure, and proper planning at the global level. Assuming that all these prerequisites are in place, the team's performance on complex tasks still critically depends on the effectiveness of internal interactions in the team. In many situations,

an important ingredient of effective collaboration is the readiness of team members to help each other.

Helpful behavior in agent teamwork has also attracted interest of a number of researchers [Itoh, 1991, Miceli et al., 1994, Yen et al., 2004, Cao et al., 2005, Fan et al., 2005, Kamar et al., 2009, Polajnar et al., 2011]. A strong motive for such studies is the emergence of mixed human and artificial agent teams, in which many of the human rules of social behavior are expected to apply. The potential significance of helpful behavior in teams consisting purely of artificial agents is discussed in [Polajnar et al., 2011]. The authors point out that, although artificial agents are *designed* with certain capabilities and for certain roles, the need for helpful behavior can arise for several reasons. The agents are often situated in environments where faults and unexpected events may happen, which may put the agents in situations not anticipated by their designers. Furthermore, as the design of agents can often be costly, in some domains it may be preferred to design agents with certain standard capabilities in the domain of their design objectives, so that they can be reused during execution of similar tasks. Thus artificial agents, like humans, may have abilities beyond their immediate roles, which can prove valuable to the team in unexpected situations, in particular if helpful behavior is supported by suitable protocols. On the other hand, providing agents with all potentially useful capabilities could be impractical, as the agents may have physical or computational components that cannot be cost-effectively transferred to every team member that could possibly need them. Therefore, in many environments an agent team, even if designed for a specific application, may still need mechanisms for mutual assistance.

In order to incorporate helpful behavior into agent teamwork, one needs to develop suitable interaction protocols. The existing studies in helpful behavior among agents have so far, to the best of our knowledge, not produced such protocols. One of the few

approaches in that direction is the work by Kamar et al. [2009], where the authors provide methods for reasoning about helpful behavior based on local probabilistic beliefs of individual agents.

In this thesis we make the case for the development of teamwork protocols for helpful behavior. When such protocols are developed, adopted, and possibly standardized, developers will be able to independently design agents that can interact and cooperate according to a set of well defined mutual assistance rules that are known in advance. A stable protocol definition provides a basis for theoretical and empirical studies of its advantages, limitations, and costs. It also allows incorporation of generic protocol versions into development toolkits and libraries, leading to major savings in development time. As argued by Miller and McBurney [2008], agents could decide dynamically when they need such protocols, thus having an option of adding helpful behavior at run time as needed. As always when software is developed for systematic reuse, and particularly in the realm of free software, there are opportunities to attain higher quality of a protocol design and implementation through feedback and collaboration of many experts on its successive refinements over longer periods.

Based on this rationale, we proceed to consider some design objectives and principles for a teamwork protocol for helpful behavior. We introduce the protocol itself in the next chapter.

3.2 The Protocol Design Objectives and Approach

As in other teamwork literature, we assume that agents in the team are truthful in their interactions. Helpful behavior differs from self-interested cooperation (e.g., CNP) in that it is motivated by team benefit rather than individual agent's benefit.

Despite this absence of fundamental conflict of interest, the viewpoints of team members can differ because they do not have the same expertise and do not have identical information about the state of the environment or the state of teamwork in progress; and since they have a degree of individual decision autonomy, their decisions may not always be in accord with each other. Moreover, the decisions of individual members may not always serve the team interest as it would be perceived on the basis of the total knowledge collectively held by all members. Thus, an intended helpful act, based on limited knowledge of an individual team member, may in fact not be helpful to the team, even if the team as a whole knows enough to detect the problem. Yet, in most complex systems it would be quite unrealistic to postulate that each individual team member must know everything the team knows. In a centralized team organization, one might require the team leader to know all that the team knows; but it is well known that such solutions are vulnerable to failure and do not scale as the system size increases. Intuitively, a strict subjection of helpful acts to central approval by the team leader preempts one of the key purposes of helpful behavior, which is to overcome local problems without raising them to the global level.

The observations above reinforce the need for carefully designed protocols that regulate helpful behavior in teams. In particular, the protocol designer must consider whether the agents involved in the decision on performing a helpful act are likely to possess the relevant information. In a recent study of helpful behavior [Kamar 2009], a team member uses its own beliefs to unilaterally decide whether to help another agent; no communication is needed prior to the decision. A different approach is used in [Polajnar 2011], an article on empathy-based helpful behavior: the agent requesting help interacts with agents willing to offer help, possibly leading to a bilateral (or multilateral) agreement. We adopt the latter principle as a basis for the protocol introduced in this thesis. The reasons motivating this choice are discussed below.

In our approach, helpful behavior is viewed as a supplementary, corrective mechanism, whose purpose is to improve the performance "in the small", at a fine level of granularity of the operations performed, without altering the general team structure and organization, agent role design, and global plan construction. Our model therefore situates helpful behavior in a context where these general, high-level aspects of team operation are fixed. It is possible that mutual assistance is needed because the events arising in the environment have exposed certain weaknesses of the current team organization or global plan. But if the inadequacies are of such magnitude that they require team reorganization, role redesign or reassignment, or replanning at the global level, then the situation is beyond the scope of our present study.

Consistent with the above restrictions, and in order to keep our model of agent team as general as possible, we assume as little as possible about the degree of centralization of the team organization. The assumption that we do make is that each agent performs a certain subtask, for which it autonomously develops a plan of its own actions, intended to meet the time and cost limits prescribed by the team. In that context, we consider two kinds of helpful acts. First, an agent may perform an action on behalf of another, using its own time and cost budget. Second, an agent may grant a portion of its own cost budget as assistance to another. In order to not depend on any features that some teams might not have, we rely on decentralized mechanisms for deciding whether an agent should help another. About the MAS infrastructure we also assume relatively little; primarily, the communication cost of consulting about help and its eventual delivery must not be prohibitive. The intent is to have a protocol for helpful behavior that is applicable across a wide range of MAS team models. Clearly, some of those models may support it better than others, and some may include effective alternative mechanisms for similar purposes.

In order to come up with a cost-effective plan, the agent must be able to generate

and evaluate alternative candidate plans. The candidate plans are compared with respect to team interest rather than the agent's self-interest. To this end the team member arguably has a harder duty than a self-interested contractor in CNP, because its decision criteria involve non-local circumstances such as the state of the team's environment beyond its own scope of perception, the progress of work elsewhere in the team, and dependencies between subtasks within the global plan. The agent's belief set used in the comparative evaluation of plans thus consists of two subsets: its *local beliefs*, based on its own perception of the environment external to the team, and its *context beliefs*, developed in interaction with the team, that contain team-level information relevant to the individual. We do not specify the team-level information that the agent has, but we do assume that it is sufficient for a sound assessment of the *team impact* of each candidate plan. If that is not the case, then the team organization itself is ineffective.

The above analysis motivates our position that, within the general assumptions we have outlined, each helpful act in agent teamwork should occur based on a distributed agreement among its participants, rather than based on a unilateral decision. For instance, consider the case when agent A contemplates a helpful act of performing an action within the current plan of agent B. In [Kamar 2009], A unilaterally assesses the team impact of help as the difference between the team utility of the scenario with help and the team utility of the scenario without help. The calculation of team utility is based on A's own beliefs about the probabilities of team members' actions. Note that a decision to help implies a change of plans for both A and B. In an effective team organization that complies with our assumptions, A should indeed be able to assess the team impact of its own plan change, in the same way as it assesses its own candidate plans. The presumed ability of A to properly assess the team impact of B's plan change may or may not exist; its absence does not seem to imply that the

underlying team organization is ineffective. We prefer to opt for a distributed decision mechanism that lets B assess the team impact of its own plan change. We already know that B has that ability because it routinely assesses its own candidate plans.

The purpose of MAP is to enable team members to respond to arising difficulties through direct mutual assistance, without the need to raise those difficulties at the global team level. The design of MAP avoids assumptions about the global team organization, its degree of centralization, or the techniques used for global plan construction, subtask assignment, and resource allocation. However, the design of MAP does assume that the global decisions in effect are sound and suitable for the team's task. MAP is not designed to help overcome team-level problems of structure or strategy, even if helpful behavior may alleviate such problems in the short term. Because of this, we regard MAP as a 'secondary' protocol, designed to support smoother teamwork in the presence of moderate challenges, while challenges of higher magnitude may require the use of 'primary' protocols for reorganization or replanning at the global level.

Chapter 4

The Mutual Assistance Protocol

This chapter introduces a mechanism for incorporating helpful behavior into agent teamwork and formalizes it as the Mutual Assistance Protocol (MAP). Section 4.1 formulates the general theoretical framework of MAP; Sections 4.2 and 4.3 provide the formalization of MAP, Section 4.4 presents a discussion on team-level influences on MAP, and Section 4.5 discusses two variations of MAP.

4.1 The General Framework of MAP

Many agent teams work in settings where there exists an overall plan for the main task, with projected timing and cost associated with each of its subtasks that individual members of the team perform. Examples of such teams include different engineering and planning teams. The team organization may involve dependencies among the subtasks, as some subtasks may need others as constituents. As the environment in which the team operates may be dynamic, unexpected events may happen during the execution of subtasks, requiring the agents to change their plans towards achieving the subtasks. Such changed plans may not be able to satisfy the given time and cost requirements, impacting not only the affected agent's performance, but

also other dependent team members' subtasks and the overall performance of the team. Situations of this type require mechanisms that improve the team robustness towards the changes in the environment in the sense that the team performance is not degraded significantly. Such a robustness can be provided through incorporating helpful behavior mechanisms among team members. As a first step towards defining such a mechanism, we describe the team collaboration process at an abstract level. In our model, we do not specify how the global plan for task assignments is constructed and assume that the member executing the subtask has potentially enough capabilities to do that. In addition, we assume that the changes in the environment in which the team operates are moderate, so that they do not result in a need for changing the overall team structure. We also assume that the agents are able to evaluate their plans and can calculate accurate team impact of the chosen plan. While such calculations need not return exactly accurate values, we assume that their approximation returns reasonably accurate values.

In our MAS model, a team of agents $A_1, A_2, \ldots, A_n$, $n > 1$, operates in an environment E. The team is assigned a task T, with each agent A_i currently assigned a subtask T_i, along with a requested completion time limit $Deadline(T_i)$ and cost budget $TotalCost(T_i)$; we do not specify how subtasks, deadlines, and budgets are assigned, but assume that those assignments are stable unless we indicate otherwise.

The environment E is *deterministic* in the sense that, when an action α is performed on a given state of E, the resulting state of E is uniquely determined. The environment is *dynamic* in the sense that its state evolves over time due to both agent actions and other unspecified factors. We refer to the latter changes as *events* in the environment; the team cannot predict the future events but may need to react to them. It is the duty of each agent A_i to perceive the events in a particular environment segment E_i and react to them as necessary. The design of agent roles

in a team and the distribution of subtasks among its members often rely on specific expectations regarding event patterns. We informally speak of *unexpected behavior* or *unexpected events* to indicate that the actual event patterns do not match such expectations.

Each agent A_i forms a set of *local beliefs* B_i based on its own observation of the external environment and its own progress in performing its subtask, as well as a set of *context beliefs* C_i based on interactions with the rest of the team. Beliefs are logical statements representing the agent's view of the world. The beliefs sets B_i and C_i take values from the domain called *BeliefSets*. As both belief sets evolve in time, we write $B_i(t)$ and $C_i(t)$ when their dependence on time t is explicitly discussed, but omit the argument otherwise.

The agents can modify the state of the environment by performing actions from a given finite set called *Actions*. Each agent A_i has an associated set $Actions_i \subseteq Actions$, representing the actions that A_i is capable of performing, along with the functions $cost_i$: $Actions_i \longrightarrow \mathbb{R}_+$, and $duration_i$: $Actions_i \longrightarrow \mathbb{R}_+$ (where $\mathbb{R}_+$ is the set of non-negative reals), representing the cost and time that A_i requires for performing the action. In general, the agents can be specialized and have different capabilities. The duration and cost of an action α performed by an agent A_i can vastly differ from the duration and cost of the same action α performed by a different agent A_j. Each agent A_i is aware of the duration and cost of each action α that it performs; in general, it may not know the duration and cost of actions performed by others.

In order to perform its subtask, agent A_i can autonomously generate a *plan*. A plan is a sequences of actions that an agent A_i can perform; it belongs to the domain $Plans_i = Actions_i^*$. When A_i generates a plan $\pi_i = \alpha_i^1 \ldots \alpha_i^k$, $k \geq 1$ in $Plans_i$, it uses

the duration and cost of each action α_i^ℓ to calculate the total duration and total cost of π_i. The agent generates a set of alternative plans, discards those that fail to meet the prescribed deadline and budget, and selects the best plan for the remaining subset. The best candidate is selected based on team interest as perceived by A_i, represented by the A_i's *team utility function*

$$u_i\colon \mathit{Plans}_i \times \mathit{BeliefSets} \times \mathit{BeliefSets} \longrightarrow \mathbb{R}$$

where $\mathbb{R}$ is the set of reals. The team utility function value $u_i(\pi_i, B_i, C_i)$ represents the A_i's estimate of the team's overall utility of following its global plan within which A_i's own plan is π_i. In general this estimate is imperfect because A_i has a partial knowledge about the total state of team's activities.

If the team organization is to be effective, it is an essential design requirement that A_i should be able to accurately compare its candidate plans. Given any two candidate plans π and π', the difference

$$\Delta_i(\pi, \pi') = u_i(\pi, B_i, C_i) - u_i(\pi', B_i, C_i)$$

must accurately reflect the *team impact* of A_i's choosing π over π'. It is worth noting that what matters is the difference rather than u_i itself. (For instance, if u_i has additive components that do not depend on A_i's plan, they will cancel out and their accuracy is immaterial.) For these reasons we henceforth assume that *the protocols for mainstream collaboration within the team are so designed that the context beliefs C_i are always sufficient to enable A_i to accurately calculate the team impact of choosing one plan over another.* By 'accurately' we mean that the level of accuracy is good enough to meet the requirements of the application; the actual criteria in practice are domain specific. We shall not elaborate how these context beliefs are formed and updated, since our interest is not in the primary collaborative mechanisms but in

the secondary ones, namely in protocols for helpful behavior. However, our design of MAP relies and depends on the assumption that each agent can competently assess the team impact of its choice among its own candidate plans. Our rationale is that if within our general team model that assumption is not met, then the team organization is fundamentally ineffective and its refinement through addition of helpful behavior is unwarranted.

As the team operates in a dynamic environment, some of the events happening there may result in a need for a plan change for some agents of the team. Moreover, as there may be subtask dependencies in the team, such a plan change may lead to plan changes for other agents as well. If agent A_i has to change its original plan π_i to satisfy the requirements, given the updated beliefs B_i and C_i, it attempts to generate a new plan for the rest of the subtask. If it is possible to replan the rest of the execution of a subtask, so that the changes are handled within the deadline and available resource limits, the agent proceeds with that plan. Otherwise it may raise its concerns at the team level, possibly leading to reallocation of deadlines and resources to subtasks, or to structural changes in the global plan; such considerations are beyond our current scope. Given that the global interventions are costly, it may benefit the team if the difficulty can be overcome through helpful behavior between the team members. Thus, when a protocol for helpful behavior is available in the teamwork model, the agent asks for help before raising its concerns to the global team level. An agent may ask for help for two reasons: when it has difficulties performing an action because of the lack of capabilities or because of the the lack of resources. We discuss these two cases separately in the next two sections. For each case, we provide the description of the help request generation, deliberation about whether to help, and deliberation about choosing the most suitable help offer.

4.2 Action MAP: Help by Performing an Action

In this section we provide a formalization of the Mutual Assistance Protocol in the case when an agent does not have adequate capabilities to perform an action of its plan effectively and needs to ask for help.

When each of the candidate plans generated by the agent A_i misses the watermark, which indicates that the plan comes close to failing in satisfying the deadline or resource requirements, A_i recognizes the need to ask for help. It selects a potential plan π_i and decides to request help in performing a specific action α_i^k within π_i. The helpful act needs to be performed within a certain time interval $[t_1, t_2]$, in order to fit with the rest of π_i. On the one hand, the execution of α_i^k may start no earlier than the time t_1 when the necessary inputs for α_i^k are ready. On the other hand, the execution of α_i^k must be completed no later than the projected deadline t_2 for α_i^k, so that the remainder of the plan after the execution of α_i^k can be completed on time. Let $\bar{\alpha}_i^k[t_1, t_2]$ be the action identical to α_i^k, but performed free of cost in the time interval $[t_1, t_2]$. Let $\pi_i' = help^+(\pi_i, \alpha_i^k, [t_1, t_2])$ denote the resulting plan of agent A_i that results from the substitution of $\bar{\alpha}_i^k[t_1, t_2]$ for α_i^k in the original plan π_i. In order to evaluate the importance of the requested help action, A_i uses the team utility function $u_i(\pi_i', B_i, C_i)$ to calculate the team utility of the plan π_i' in which the action α_i^k is performed free of cost at the time interval $[t_1, t_2]$.

The difference between the team utility function values of the plans π_i' and π_i reflects the *team benefit*, from the requesting agent's perspective, of having the specified action α_i^k performed by another agent:

$$\Delta_i(\pi_i', \pi_i) = u_i(\pi_i', B_i, C_i) - u_i(\pi_i, B_i, C_i)$$

Having recognized the action for which it needs help, the time interval of the needed help, and the team benefit of the requested action, A_i broadcasts a help request to the team members:

$$Broadcast(HelpRequest(A_i, \alpha_i^k, [t_1, t_2], \Delta_i(\pi_i', \pi_i)))$$

When a team member agent A_j receives the help request, it has to decide whether to agree to help. The agent first checks whether it has the capabilities to perform the requested action in the time interval provided in the help request. If A_j does not have the capabilities, it ignores the help request. If A_j has adequate capabilities to help A_i by performing the action α_i^k, it considers the change of its own plan π_j to a new plan $\pi_j'' = help^-(\pi_j, \alpha_i^k, [t_1, t_2])$ in which it additionally performs the action α_i^k in the time interval $[t_1, t_2]$. A_j then calculates the team utility function for the plan π_j''.

The difference between the utility function values for the plans π_j and π_j'' reflects the *team loss*, from the A_j's perspective, if it, in addition to the actions of its own plan, also performs the action α_i^k in the time interval $[t_1, t_2]$:

$$\Delta_j(\pi_j, \pi_j'') = u_j(\pi_j, B_j, C_j) - u_j(\pi_j'', B_j, C_j)$$

A_j uses the team loss value in order to deliberate whether to agree to help to agent A_i. If the calculated team loss value is smaller than the team benefit value of the same action specified in the request, then A_j agrees to help. Otherwise it ignores the help request. The difference between the team benefit value specified in the request and team loss value of the same action calculated by the helping agent is called the *net team impact*:

$$NetImpact_{ij}(\alpha_i^k) = \Delta_i(\pi_i', \pi_i) - \Delta_j(\pi_j, \pi_j'')$$

If the net team impact is a positive number, then A_j agrees to help and bids for the requested action α_i^k, sending the value of the net impact for α_i^k within its bid message to A_i. The bid must be delivered within the fixed time δ after receiving the help request:

$$Bid(A_j, \alpha_i^k, NetImpact_{ij}(\alpha_i^k))$$

When A_i receives the submitted bids, it chooses the bid that it views as the most favorable for the team, namely the bid with the highest net impact.

One should note that the reasoning about team impact in MAP is inherently approximate, in that no agent assesses the simulataneous impact of both local plan changes. In our view, this is outweighed by the fact that each agent assesses its local circumstances with which it is highly familiar.

We are now ready for a complete definition of the Mutual Assistance Protocol. It is specified in the three steps below. Step zero represents the preliminary step in the team operation, which is out of the scope of this Thesis.

0. The team members get individual subtasks according to their abilities, with the corresponding deadlines and the total allowed costs for completing the subtask. The team members also get the context beliefs regarding the execution of the subtask. They make the initial plan for accomplishing the given subtask.

1. During the execution of the subtask, environmental or context belief changes may happen, because of which a team member agent may recognize that it needs help for completing its subtask. In such a case, the agent makes an alternative plan, decides about the action for which it needs help along with the time interval when the help needs to be delivered, and calculates the team benefit if it is assisted by a team member. The agent then broadcasts a help

request, in which it mentions the particular action for which it needs help, the calculated time interval, and the amount that the team will benefit if such a help is provided.

2. Each team-member agent that is qualified for performing the requested subtask assesses the request and calculates the cost that the team will suffer if the agent executes the requested help. If the calculated team cost is lower than the team benefit mentioned in the request, the agent bids for the help request, specifying the team loss for executing the help. Otherwise the agent ignores the request.

3. The requesting agent receives all the bids from team members and selects the bid which has the most favorable impact on the team. If no offer comes for the requested action, the agent generates another alternative plan.

4.3 Resource MAP: Help by Providing Resources

This section presents the Mutual Assistance Protocol in the case when an agent has run out of resources during the execution of the plan and needs additional resources from the team members in order to complete the plan.

As the environment is dynamic, and unpredicted events may happen, the agent may ask for resources only for the next action of the plan. After recognizing the need for additional resources, the agent A_i calculates the amount of resources D_i that it lacks. This amount may change over time, as the changes in the environment may force A_i to change its plan, and particularly, the next action.

In order to evaluate the importance of the requested amount of resources, A_i uses a team utility function

$$\tilde{u_i}\colon \mathit{Plans}_i \times \mathbb{R}_+ \longrightarrow \mathbb{R}$$

The team utility function value $\tilde{u}_i(\pi_i, R_i)$ (where R_i is the remaining resources of agent A_i) represents the A_i's estimate of the team's overall utility if A_i follows the plan π_i in which A_i has resources R_i. The function value $\tilde{u}_i(\pi_i, R_i + D_i)$ represents the team impact if the plan π_i is executed with additional resources D_i.

The difference between the team utility function values represents the team benefit, from the requesting agent's perspective, if the specified amount of resources is provided to the agent A_i.

$$\tilde{\Delta}_i(\pi_i, D_i, R_i) = \tilde{u}_i(\pi_i, R_i + D_i) - \tilde{u}_i(\pi_i, R_i)$$

Having recognized the amount of resources needed and the the team benefit associated with the requested amount, A_i broadcasts a help request to the team members:

$$Broadcast(HelpRequest(A_i, D_i, \tilde{\Delta}_i(\pi_i, D_i, R_i)))$$

When a team member agent A_j receives the help request, it has to decide whether it should help A_i with providing the full or partial amount of resources mentioned in the request. There may be situations when no individual team member is able to provide the full amount of resources mentioned in the help request without making its own situation worse than the one of the agent A_i. However, by contributing to A_i partially, the team members together may accumulate the necessary amount of resources requested by A_i. When deliberating whether to help A_i and with how much resources to help, an agent A_j uses the notion of proportionality; A_j deliberates whether at that time point there is a non-empty set Q_j of possible resource amounts d_j it can offer, and a coefficient q_j, such that the proportional team loss associated with the absence of that resource amounts in Q_j is less than equal to the coefficient q_j, whch is less than the proportional team benefit of the agent A_i associated with

the existence of the additional resources D_i. The deliberation by A_j is performed by calculating the utility function $\tilde{u_j}(\pi_j, R_j)$ and $\tilde{u_j}(\pi_j, R_j - d_j)$ to determine the team impact if A_j executes the plan π_j with the amount of resources R_j and $R_j - d_j$, respectively.

$\tilde{\Delta_j}(\pi_j, d_j, R_j) = \tilde{u_j}(\pi_j, R_j) - \tilde{u_j}(\pi_j, R_j - d_j)$ reflects the team loss of the agent A_j for giving away a resource amount d_j. Thus, the proportionality condition for agent A_j leads to the following:

$$\frac{\tilde{\Delta_j}(\pi_j, d_j, R_j)}{d_j} \leq q_j < \frac{\tilde{\Delta_i}(\pi_i, D_i, R_i)}{D_i} \tag{4.1}$$

If 4.1 does not hold for any amount of resources, then agent A_j ignores the help request. Otherwise A_j submits a bid, specifying the set of amounts it is able to transfer, along with the team loss coefficient associated with the provided set:

$$Bid(A_j, A_i, Q_j, q_j)$$

Agent A_i receives the submitted bids and checks whether the collective amount of resources offered in the all bids together is greater or equal to the amount that it requested from the team. If the team collectively was not able to provide the agent A_i with the necessary amount of resources, then A_i may ignore all the bids or select the combination of bids most beneficial to the team, depending on the purpose of the requested resources. If the collectively offered resources in the bids are greater or equal to the requested amount, A_i chooses the the combination of bids which provide the agent with the requested amount of resources:

$$\sum_{k=1}^{r} d_{j_k} = D_i$$

and minimize the upper bound on the team loss:

$$\sum_{k=1}^{r} q_{j_k} * d_{j_k} \ \textit{is minimal}$$

The proportionality condition then ensures that the combined team loss of all helpers remains lower than the team benefit specified in the help request. Since for all $k \in \{1, \ldots, r\}$,

$$\frac{\tilde{\Delta}_{j_k}(\pi_{j_k}, d_{j_k}, R_{j_k})}{d_{j_k}} \leq q_{j_k} < \frac{\tilde{\Delta}_i(\pi_i, D_i, R_i)}{D_i}$$

it follows that

$$\begin{aligned}
\Sigma_{k=1}^{r} \tilde{\Delta}_{j_k}(\pi_{j_k}, d_{j_k}, R_{j_k}) &\leq \Sigma_{k=1}^{r} q_{j_k} d_{j_k} \\
&\leq \max_k q_{j_k} \Sigma_{k=1}^{r} d_{j_k} \\
&= \max_k q_{j_k} D_i \\
&< \frac{\tilde{\Delta}_i(\pi_i, D_i, R_i)}{D_i} D_i \\
&= \tilde{\Delta}_i(\pi_i, D_i, R_i)
\end{aligned}$$

The complete steps of the Mutual Assistance Protocol in case of the resource needs are specified below:

0. The team members get individual subtasks according to their abilities, with the

corresponding deadlines and the total allowed costs for completing the subtask. The team members also get the context beliefs regarding the execution of the subtask. They make the initial plan for accomplishing the given subtask.

1. During the execution of the subtask, an agent may be in a need for additional resources. In such a case, the agent requests help from the team members, specifying the amount of resources needed, and the possible team benefit if such an amount of resources is provided.

2. Each team-member agent assesses the request and decides whether it can provide any amount of resources, such that the proportional team loss associated with the loss of that resources is lower than the proportional team benefit provided in the help request. If such an amount exists, the agent finds the maximum amount satisfying the criteria and submits a bid for the help request, specifying the offering amount of resources and the corresponding team loss.

3. The requesting agent receives all the bids from team members and checks whether the collective amount of bidded resources are enough to accept any bidded amount. If so, then the agent selects the bid(s) which guarantee the requested amount of resource income, in the meantime causing the team the minimum amount of damage. Otherwise the agent may or may not take any of the resources offered in the bids, depending on the purpose of the request.

4.4 Team-Level Influences on MAP

In this section, we discuss how in MAP the plans of agents may be influenced because of plan changes of certain team members.

Consider a team of agents working on a project task divided into seven subtasks,

as shown in Figure 4.1. The subtasks are represented by rectangles, and the dependencies among the subtasks are represented by arrows. Subtasks T_1 to T_4 are executed in parallel, after completion of which the execution of the remaining subtasks starts. Each subtask has its projected duration and total expenses, as shown in the figure. The number above each subtask represents the projected time estimate for the completion of the subtask, whereas the number in the top-right corner of the subtask represents the estimated expenses for the completion of the subtask. The critical path in the project is represented by red arrows, which includes the subtasks T_4, T_6, and T_7. The project diagram with its associated schedules and cost information is posted on a commonly accessible area and serves as context beliefs for the team agents.

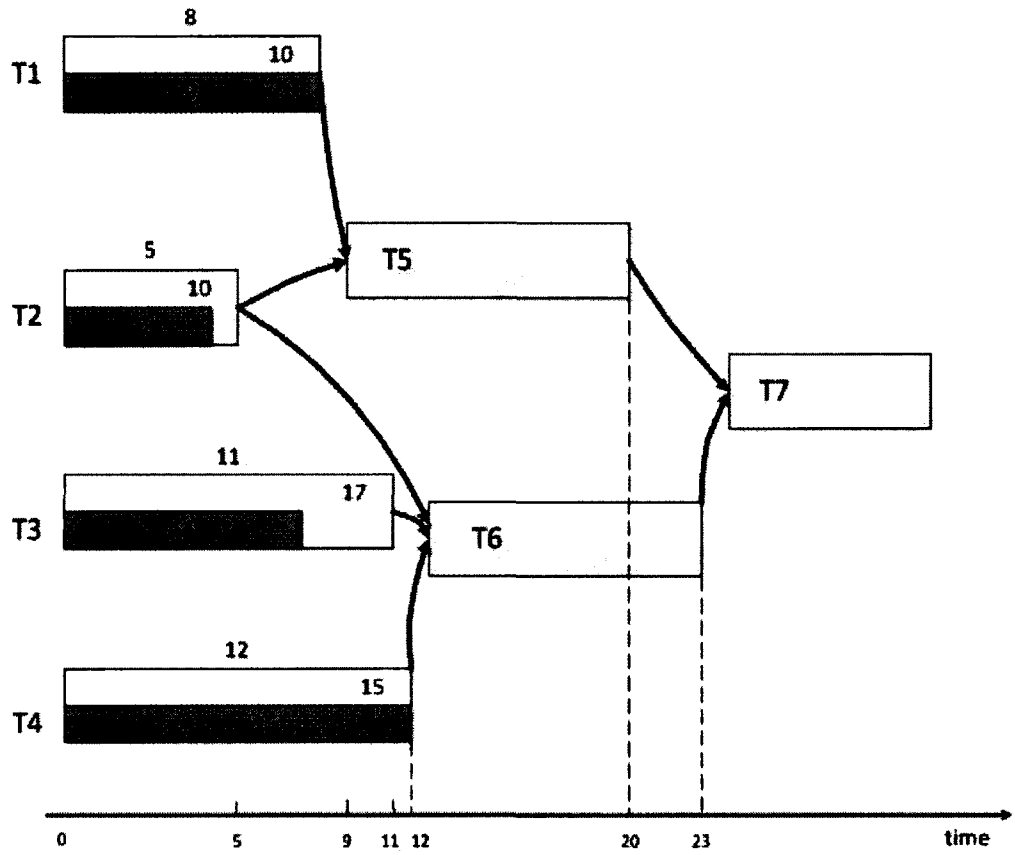

Figure 4.1: A Diagram for Team Project

When the project starts, agents in charge of executing the subtasks T_1 to T_4 generate individual plans based on their abilities, so that the plan satisfies the given time and cost requirements. The constructed plans may have different time and cost values compared to what the team estimates are. Once the agents select their plans, they post their schedules and the costs on the commonly accessible diagram, thus updating their own and some of their team members' context beliefs. In Figure 4.1, the durations of the individual plans of agents are represented by the shaded area inside the subtasks, and the associated costs of the chosen plans are represented by

the numbers inside the shaded regions.

The diagram also gives to the team members information about the possible allowed lateness of the completion of each subtask and relevant information about available resources. For instance, if because of environmental changes the agent A_2 experiences minor delays for the completion of subtask T_2, such a delay will not cause the team any problems, as the accomplishment of the subtask T_2 will be needed for other subtasks only when the subtask T_1 or T_3 and T_4 together are completed. However, major delays of T_2 may delay the completion of the overall project, affecting the team performance. While the delay of T_2 with a consequent delay of T_5 may not be vital, the case of consequent delay of T_6 may cause the team significant damage. On the other hand, spending additional amount of resources for the completion of T_2 may affect the team noticeably. Thus, agent A_2 has certain time flexibility for completing T_2 and should attempt on completing T_2 spending fewer resources. The situation is different for the task T_4, which is on the critical path, and any delay of the completion of T_4 may lead to the delay of the entire project, affecting the team significantly. In such a case, when the timely completion of the subtask is critical, completing that subtask by spending additional resources may cause the team less damage than completing it by spending additional time.

The above-mentioned cases reflect the actual methods for calculating the team impact utility functions in team models like the one in Figure 4.1. They also show that these functions can be efficiently calculated by agents, given the characteristics of the subtask and the chosen plan.

The utility functions of team impact are used very frequently by the team member agents. The usage of these functions is more prominent when the agents or other team members need help, or when they need to consider change of the plan. The agents

may need to change their plan either because of environmental changes that make the agents to choose a different plan to satisfy the requirements, or because of the changes of context beliefs, for which the agent should react accordingly. Having discussed the usage of utility functions in the case of environmental changes above, we now describe its usage in case of changes of context beliefs.

Even when an agent is operating according to the plan satisfying its current requirements, it may need to choose a different plan because of context belief changes, which may trigger changes of the requirements. An example of such a situation is illustrated in Figure 4.1. In the example, the agent A_4 is executing the plan which satisfies its time and cost requirements. However, as the subtasks T_2 and primarily T_3 are ahead of their schedules and will likely be completed significantly sooner than it was estimated, the team may decide to accelerate the completion of T_4, aiming at earlier completion of the entire project (if the earlier completion of the project is important for the team). In this case, the team may change the requirements of the subtask T_4 and request A_4 to complete the task T_4 earlier. While the early completion of T_4 may influence the team positively, it may not be beneficial if it exceeds its limits. For instance, it would be pointless to request acceleration of T_4 for more than 3 time units, as there is no benefit to have T_6 completed before the completion of T_5 (Figure 4.1).

The global planning issues of the type discussed above affect the helpful behavior through the context beliefs of individual agents. The context beliefs represent the local information of a team member agent about the team state, relevant to the completion of its individual subtask. Similar to the mecanisms of updating the agent's local beliefs B_i through a belief-revision function based on the perception of the environment, context beliefs are updated through a belief-revision function based on the information input from the rest of the team. They usually contain information

about changes of plans of team members whose subtask completion schedules may in one way or another influence the plans of certain other team members. As the subtasks of the team may have inter-dependencies, having an up to date context information is critical for each team member in order to judge its actions and be a part of the team. In addition, the context beliefs inform agents about the priorities of the team regarding the time or resource expenses while executing the subtask.

As discussed in Section 4.1, some of the unexpected events may lead to changes of plans for some agents. Such changes will usually result in spending different amount of resources or time for the completion of the agents' subtasks. As there may be inter-dependencies among the subtasks, in order to increase the effectiveness of the team, changes of the schedules of some tasks may require appropriate changes of schedules for the other dependent subtasks. These observations result in a conclusion that the communication of relevant and timely context information among the team members is a critical part for the design of efficient teamwork.

While sharing individual context information is an important part for team operation, deciding with whom to share the context requires deliberation. As the agent may have incomplete information about the set of team members for which its activities may turn out to be relevant, it needs to make the changes of its plan publicly available. However, sharing every information with every team member may be inefficient, as some of them may not need to know anything about the changes of the plan of a particular agent. One approach towards organizing the context sharing is the usage of a commonly accessible area, where agents may share their plan changes. While this method solves the problem, it may require the agents to spend additional time and resources for looking up the necessary information for them.

In order to provide efficiency in updating the relevant context beliefs, one would

need some sort of publish/subscribe mechanism which ensures that without broadcasting the updating information or posting on a commonly accessible area, the team members do get the relevant information needed for proper execution of their subtasks. We did not discuss the mechanisms of global planning of the team and distribution of the subtasks as it is not in the scope of this Thesis. Depending on such mechanisms or the global structure of the team, certain mechanism may be deployed in which the team members will be subscribed to proper information sources and will get the necessary updates of their context beliefs without performing an excessive search or getting the updates through a broadcast message.

4.5 Variations of MAP

This section presents two variations of Mutual Assistance Protocol. The variations include discussions about an alternative mechanism for performing a help request, and about an alternative approach for designing MAP.

4.5.1 Simultaneous Help-Seeking Mechanism in Action MAP

When an agent realizes that it needs help, in some situations it may prefer to generate several candidate plans towards achieving its goal and send simultaneous help requests for the actions for which the agent needs help. Moreover, there may be situations when the agent may prefer to send help requests for different actions belonging to the same candidate plan. Below we provide a mechanism for selecting the most preferable bid in situations mentioned above.

As the agent A_i may have requested help for actions belonging to different plans, and even for different actions belonging to the same plan, finding the most favorable

bid requires deliberation. For the bids related to the same action α_i^k, A_i compares the net team impact values $NetImpact_{ij}(\alpha_i^k)$ of each bid and chooses the bid with maximum net benefit value. In this way, A_i is able to find the best bid for each action that requires help. For the bids related to actions belonging to different candidate plans, the agent, again, selects the best bid by comparing the net team impact values. The best bid is chosen according to the maximum value of net team impact, as that is the bid which will bring the team the highest overall benefit.

We now consider the case when there are bids for different actions belonging to the same plan. If the bids are unrelated (i.e., the success in executing an action does not depend on the acceptance of the bid for another action), then A_i first selects the bid with highest value of net team impact. That bid will be associated with some action for which A_i had requested help. Let us call that action α_i^p. The agent A_i then finds the bid with second highest net team impact and its associated action α_i^q. In order to decide whether, in addition to accepting the bid for α_i^p, it should accept the bid for α_i^q as well, A_i recalculates the team benefit value $\Delta_i'(\pi_i', \pi_i)$ for action α_i^q, assuming that the action α_i^p is performed for free. This team benefit value $\Delta_i'(\pi_i', \pi_i)$ for α_i^q will be different (lower) than the original value, as now the calculation assumes that another action α_i^p is already performed free of cost. However, if the recalculated team benefit value $\Delta_i'(\pi_i', \pi_i)$ for action α_i^q is still higher than the team loss value $\Delta_j'(\pi_j, \pi_j'')$ derived from the bid, the agent accepts the bid for action α_i^q as well. Otherwise it ignores the bid for α_i^q. A_i performs this reasoning for all the bids that contain different actions belonging to the same plan.

If there are related bids (i.e., the success in executing a certain action depends on the acceptance of the bid for another action), then A_i makes them unrelated by combining the related bids into one compound bid, whose team benefit and team loss values are the sum of its components. After performing this step and having only

unrelated bids, A_i can perform the previous step in order to find the best bid or the combination of best bids.

Next, suppose that A_i receives a combination of bids for the same action, bids for different actions belonging to the same plan, and bids for different actions from different plans. In order to choose the best offer(s), A_i should first find the best bid for each action, then find the best options related to a particular plan, and finally compare the best options of each plan in order to choose the most beneficial option among all bids.

4.5.2 Helper-Initiated MAP

The design of the Mutual Assistance Protocol is based on the requester-initiated interaction, when an agent recognizing the need for help initiates the interaction with team members for the sake of getting help in the interest of the team. However, the same principles used in the design of MAP can be used in developing a helper-initiated protocol, in which a team member a_i that is well ahead of the deadline of its subtask or has a large spare amount of resources broadcasts a message to the team members indicating its readiness to help in a certain action α_i^k or with certain amount of resources D_i, also broadcasting the team loss Δ_i associated with the potential help. The cases of helper-initiated Action-MAP and helper-initiated Resource-MAP are discussed below separately.

Helper-Initiated Action MAP

In the case of helper-initiated Action MAP, the agent A_i that is ahead of its schedule or has a large amount of spare resources broadcasts a message to the team members, indicating the willingness to help the team members by performing an action α_i^k that costs A_i little time or little resources. In the broadcasted message, A_i

also specifies the team loss Δ_i that such a potential help would entail. Each team member A_j receives the broadcasted message and checks whether it needs the offered help action in any stage of its currently-chosen plan. If so, A_j calculates the team benefit Δ_j if it is assisted by A_i in performing the action α_i^k in the time interval $[t_1, t_2]$ in the plan π_j. If the calculated team benefit of A_j is higher than the team loss provided in the broadcasted message by A_i ($\Delta_j > \Delta_i$), then A_j bids to the help offer sent by A_i, also mentioning the time interval in which A_j needs the helpful action to be executed, and the net team impact ($\Delta_j - \Delta_i$) associated with the potential help. A_i receives the bids from the team members reacting to the help offer, deliberates whether it can perform the offered action within the time interval specified in the bid, and among the feasible bids A_i chooses the bid with maximum net team impact value. The help is then delivered to the agent for which the net team impact value is the maximum.

Helper-Initiated Resource-MAP

In the case of helper-initiated Resource MAP, the agent A_i that has a spare amount of resources broadcasts a message to the team members, specifying the willingness to help them by transferring an amount of resources from the set Q_i. Each amount D_i in Q_i must be such that it does not endanger the performance of A_i's own subtask T_i, and is associated with relatively low team loss value. This is expressed by the condition $\tilde{\Delta}_i(\pi_i, D_i, R_i)/D_i < q_i$ i.e., that the relative team loss is lower than the fixed value of the coefficient q_i. Within the broadcast message, A_i specifies the set Q_i and the coefficient q_i. Each team member A_j receives the broadcasted message and deliberates whether it needs any additional resource points d_j for improving the team benefit associated with the completion of its subtask T_j;if so, A_j responds with a bid message specifying the desired set of values Q_j and a fixed coefficient q_j, where $q_j > q_i$ and $\tilde{\Delta}_j(\pi_j, d_j, R_j)/d_j$ is greater than or equal to q_j for each d_j in Q_j. Similar

to the analysis in Section 4.3, these conditions ensure that the combined team benefit of all agents receiving help is greater than the team loss of the helper. A_i receives the bids and selects the ones with highest proportional team benefit values.

4.5.3 MAP of Achievement and Maintenance Tasks

So far, we have discussed the usage of the Mutual Assistance Protocol in achievement tasks, where the team members are working to make certain propositions true, and thus complete their subtasks. However, in practice there are also team collaboration models where maintenance tasks (i.e., tasks where the value of proposition needs to be maintained over time) are equally important [Kaminka et al., 2007]. This section discusses the usage and applicability of MAP for maintenance tasks.

As a team member, an agent executing a maintenance subtask knows the importance of its subtask for the team and the possible loss to the team in the case of maintenance failure. As there may be different severities of failures, in each case the team may have different amount of loss.

Based on these observations, the Mutual Assistance Protocol presented in Section 4.2 and Section 4.3 can be similarly applied to maintenance tasks. Upon being assigned a subtask, an agent A_i generates an initial plan $\pi_i = (\alpha_i^1, \ldots, \alpha_i^k)$ that guarantees the maintenance of the subtask during the team operation. If, at time t_c an unexpected event happens in the environment or in the context beliefs of the agent A_i that makes A_i to choose a different plan π_i' for which agent A_i either does not have adequate capabilities or sufficient resources, it may ask for help. The value of the team benefit for maintaining the subtask can be either known at priori or calculated at the time of an emergency.

Chapter 5

Evaluation

This chapter presents a simulation study of teamwork that includes helpful behavior based on the Mutual Assistance Protocol. Specifically, we conduct a series of experiments that compare the teamwork performance resulting from the use of MAP versus two other help methods, in which the requesting or helping agents unilaterally decide about the need and usefulness of the helpful act. Also included for comparison is teamwork without helpful behavior. The study of MAP in this chapter is limited to helpful acts in which an agent performs an action on behalf of another, as formalized in Section 4.2 (the resource assistance version of Section 4.3 is not included). The simulation environment is based on a variation of the Colored Trails (CT) game, which is a publicly available research test-bed for examining decision-making in group of agents or people[1] [Gal et al., 2010]. The following sections describe the test-bed used in the evaluation (Section 5.1), the approaches with which MAP was compared (Section 5.2), the configuration parameter settings (Section 5.3), and the experimental results (Section 5.4).

[1]The Colored Trails software can be accessed at http://www.eecs.harvard.edu/ai/ct

5.1 The Test Bed for Simulation Experiments

In the CT game, the players are randomly located on a rectangular board divided into colored squares. Each player has a supply of chips whose colors belong to the set of board colors. There are also goal squares for players, located at certain positions on the board, and the players get points for reaching the goal squares. At each turn, a player can move to a neighboring square of the board by spending a chip that has the color of that square. The purpose of the agents is to reach their goal locations, collecting the maximum number of points for the team.

The variation of the CT game used in this thesis has been developed specifically for the study of helpful behavior in teamwork and implemented independently[2]. The rules of the game are adjusted to the purposes of our current study. The players represent software agents $A_1, \ldots, A_n$, $n > 1$, that collaborate as a team. The game proceeds in synchronous rounds, with each agent trying to make a move in each round; it is legal for multiple agents to be on the same square at the same time. The game ends when no agent can make a move; all agents remain in the game until the end.

At the start of the game, each agent A_i is assigned an initial location on the board, a unique goal with a specified location and amount g_i of reward points (representing the agent's subtask), and an initial budget $s_i = d_i a$ of resource points, where d_i is the distance (expressed as number of moves) along the shortest path from the agent's initial location to its goal, and a a positive fixed budget for each move. During the game, the agent receives another fixed (non-negative) amount a' for each completed move, as a reward for intermediate progress, and collects the final reward g_i if and when it reaches the goal.

[2]The game has been developed by Omid Alemi (with the participation of Ashton Fedler) for his research and kindly provided to us for our experiments.

Each move by A_i represents an action performed by the agent and has an associated cost in resource points, paid from the agent's total budget. The board has a fixed set of colors, $C_1, \ldots, C_m$, $m > 1$, which affect the costs of the moves as specified in a predefined cost matrix c. Whenever A_i moves to a field of color C_j, the cost of the move is $c_{ij} > 0$. The value of the cost matrix element c_{ij} represents the level of ability of the agent A_i in performing the type of action represented by the color C_j: the lower the value, the higher the agent's expertise. The values can vary widely to represent the diversity of specialties in the team. A_i knows its abilities (i.e., its vector c_i of the cost matrix) and has full visibility of the board (including the locations of other agents and their goals). Each agent initially selects a path to its goal. In our experiments, each agent chooses the lowest-cost path among all shortest paths to its goal and commits to that path for the rest of the game. The chosen path may turn out to be less than optimal, as the colors of individual board fields can change during the game. These color changes represent independent events in the dynamic environment; they are unrelated to the agents' actions. The probability of color change is uniform for all board squares; we refer to it as the *disturbance level.*

During the game, each agent maintains a budget of resource points and a budget of rewards points.Initially, the agent receives resource points proportional to its distance from the goal, spends them at each move while they last, and blocks after that. Each agent earns a fixed amount of reward points for each step,a fixed amount of reward points for reaching the goal, plus the remaining amount of resource points as bonus for reaching the goal. A blocked agent remains in the game and may become active again (for instance, if the color of the next square on its path changes and makes the move affordable). The game ends when no agent can make a further move. At the end of the game, the agent's budget represent its score, and the total score across all agents is the team score. The objective of the game is to maximize the team score.

So far the game is hardly interesting, as the agents have fixed choices and need no strategy. The only element that prevents its outcome from being predictable from the start is the dynamic behavior of the environment, represented by board squares that change color. The variation in agent behavior is introduced next, as we give the agents the ability to help each other; we compare different methods of deciding whether to help. This aspect of the game profile is suited to our current purposes: as we focus on helpful behavior, we prefer to have other dimensions of agent behavior vary as little as possible.

The rule of helpful behavior states that, when an agent A_i faces a move to a square of color C_k, it is legal for another agent A_j to pay for the move at the cost $c_{jk} + o$, where o is a fixed *overhead* cost associated with each helpful act. The rule models the situation where an agent, facing the prospect of performing an action for which it is poorly qualified, receives help from an expert that can perform the action itself efficiently but has its efficiency reduced by the overhead of arranging the helpful act. If $c_{jk} + o < c_{ik}$, the helpful act objectively benefits the team.

As discussed in Chapter 4, the main difficulty in deciding whether to help is that the beliefs of an individual agent about the abilities of others, and about other relevant circumstances in the rest of the team, may not be accurate enough. In our experiments this uncertainty is modeled by the *mutual awareness probability* p, defined as follows. Each agent knows the discrete finite set of values $\mathcal{C}$ from which the values of cost matrix entries c_{jk} are chosen. Agent A_i does not know the value of the cost vector c_j of another agent A_j, but for each k has a probabilistic belief about the likelihood that the value of c_{jk} equals a particular element of $\mathcal{C}$. With the probability p, A_i guesses the correct value of c_{jk}, and with the probability $1 - p$ it makes a uniformly random guess of any value in $\mathcal{C}$ (including the correct value, which makes the likelihood of correct guess equal $p + (1 - p)/|\mathcal{C}|$). As p varies from 0 to 1, the mutual awareness of

individual abilities in the team varies from complete ignorance to perfect knowledge. In a series of simulation experiments we explore the impact of such variations upon the comparative performance of MAP and two other methods for deciding whether the help of one teammate to another benefits the team.

5.2 Methods for Deciding whether to Help

In protocols for helpful behavior, agents must decide, based on their beliefs, whether or not it is in the interest of the team that a helpful act should take place. In this section we restrict our attention to helpful acts in which one agent executes an action on behalf of another. In that context, we consider MAP and two alternative methods for deciding whether such a helpful act should occur. As the fourth possibility we include the approach with no helpful behavior. Later in this chapter, these four approaches to helpful behavior will be mutually compared for efficiency through simulation experiments using the test bed described in the previous section. We first describe the decision mechanisms of all methods (5.2.1) and then discuss their computation and communication costs (5.2.2). The mechanisms are described and analyzed as implemented in the simulation; many other variations of the same mechanisms are possible.

5.2.1 The Decision Mechanisms

For simplicity of protocol descriptions, we assume that the agents use a synchronous message-passing communication model, in the sense that they can send messages to each other in synchronous rounds, operating in lockstep based on a common clock. The communication channels are reliable in the sense that all messages get

delivered without delay. The send primitives include unicast and team-wide broadcast. If in a given round an agent receives a message that prompts a response, the agent sends that response in the next round (regardless of how much computation it requires). This reduces the message count by making the absence of a message potentially informative. For instance, a lack of response to an offer in the next round is interpreted as a negative response; no rejection message is needed. The choice of the current communication model as a vehicle for presentation has no bearing on the possibility of implementing the protocols in other models.

Always-Help: Decision by Recipient of Help

This is a *unilateral* method in the sense that a single agent decides whether help should occur. The agent that needs help uses its own beliefs to determine, for every other agent in the team, if that agent can provide the help with a resulting benefit for the team. It constructs the list of all agents that meet the criterion, ranked according to the expected team benefit. If the list of candidates is nonempty, it sends each candidate a request message. Each candidate responds (without questioning the judgment on team interest), except in the case that it is unable to deliver the requested help. The requester sends an acceptance message to the responder with the highest presumed team benefit.

Proactive-Help: Decision by Provider of Help

This is also a unilateral method, but based on the initiative and judgment by the potential provider of help. Each agent uses its own beliefs to determine, for every other agent on the team, whether it can provide help to that agent with a resulting benefit to the team. If the list of candidates satisfying the criterion is nonempty, the agent offers help to the one with the highest presumed team benefit. When an agent receives a single help offer in a given round, it responds with an acceptance message

(without questioning the team benefit or performing other deliberation). In the case of multiple offers in the same round, it randomly chooses one offer and responds with an acceptance message. A version of unilateral proactive help protocol is used in [Kamar et al 2009].

MAP: Distributed Decision by Recipient and Provider of Help

In our *Mutual Assistance Protocol (MAP)*, the decision is effected through a distributed agreement between the recipient and the provider of help, with each agent using its beliefs to assess the team impact of its local change of plan. The agent that needs help in performing an action assesses the *team benefit* of its switching to a different local plan in which the action has a cost of zero. It broadcasts the assessment in its request message to everyone else on the team. An agent that receives the request assesses the *team loss* that would result from adding the new action to its local plan in order to fulfill the help request. It responds to the request if the *team impact*, calculated as the difference between the team benefit and team loss is positive, and includes the value of the team impact in its response message. The requester accepts help from the responder with the highest value of team impact.

No-Help: Absence of Helpful Behavior

This is the approach in which no helpful acts are ever considered, and each agent relies exclusively on its own resources to complete its subtask. Its purpose is to provide a reference against which the performance of protocols for helpful behavior can be measured.

The two unilateral decision mechanisms described above fundamentally differ from MAP in that they require an agent to assess the team impact of a local plan change in another agent, while in MAP each agent assesses only the team impact of its own local plan changes. For this reason it is reasonable to expect that the performance of

protocols that use unilateral decision mechanisms should more critically depend on the level of mutual knowledge in the team to a greater degree than the performance of MAP. This dependency is examined through simulation experiments later in this chapter.

5.2.2 The Costs of Computation and Communication

The computation cost is dominated by the calculations of team utility values. Specifically, we calculate the *team impact* of agent A_i's help to agent A_j as the difference between the *team benefit* (from A_j's not having to perform an action) and the *team loss* (from A_i's having to perform an additional action); we assume that the cost of each of these two component calculations is a fixed constant value c. We also assume that other computational costs involved in the decision are considered negligible by comparison. The communication costs are based on the number of messages, with each broadcast having a fixed cost of b and each unicast of s. The total number of agents in the team is n, while $k, k_1, \ldots$ denote the variable numbers of agents participating in certain interactions. Note that the message count could be different in another communication model. Also, in architectures where broadcast must be implemented through unicast messages, its fixed cost of b is replaced by $(n-1)s$.

Always-Help

The costs of individual protocol steps in a successful help transaction are as follows:

1. The requesting agent computes the team benefit, at cost c; for each of the other $n-1$ team members it computes the team loss, at the total cost $(n-1)c$; and it sends messages to k agents (where $1 \leq k \leq n-1$) for which the computed team impact is positive.

2. Out of the k candidates for providing help, k_1 send response messages (where $1 \leq k_1 \leq k$), at total cost ks.

3. The requester sends an acceptance message to one of the candidates, at cost s.

Thus, during one successful help transaction the team as a whole spends an amount of resources equal to:

$$C_s(Always\_Help) = c + ((n-1)c + ks) + k_1 s + s = nc + (k + k_1 + 1)s$$

In the worst case, when the requesting agent sends the request message to every team member, and each of them confirms its availability to perform the help, the cost of the team in a successful help transaction equals to:

$$C_w(Always\_Help) = c + ((n-1)c + (n-1)s) + (n-1)s + s = nc + (2n-1)s$$

The help transaction is not guaranteed to be successful. It can fail if no team member responds to the help request sent by the requesting agent. If the transaction fails, the cost to the team is equal to:

$$C_f(Always\_Help) = c + ((n-1)c + ks) = nc + ks$$

Proactive-Help

The costs of individual protocol steps in a successful help transaction are as follows:

1. The $n-1$ team members compute the team benefit of the potentially struggling team member, at cost $(n-1)c$, and compute the team loss at cost $(n-1)c$, in total spending $2(n-1)c$.

2. Assuming that for k of them (where $1 \leq k \leq n-1$) the computed team impact is positive, k agents send a message to the potentially struggling agent, at cost ks.

3. The agent sends an acceptance message to one of the offering agents, at cost s.

Thus, during one successful help transaction the team as a whole spends an amount of resources equal to:

$$C_s(Proactive\_Help) = 2(n-1)c + ks + s = 2(n-1)c + (k+1)s$$

In the worst case, when every agent in a team decides to help the potentially struggling agent, the cost of the team in a successful help transaction becomes equal to:

$$C_w(Proactive\_Help) = 2(n-1)c + (n-1)s + s = (2n-2)c + ns$$

The help transaction in Proactive-Help method can fail if, after computing the team impact, no team member decides to offer help to the potentially struggling agent. In this case, the cost of the team equals to:

$$C_f(Proactive\_Help) = 2(n-1)c$$

MAP

The costs of individual protocol steps in a successful help transaction are as follows:

1. The requesting agent computes the team benefit, at cost c, and broadcasts the request to the team members, at cost b.

2. Each member computes the team loss, at cost c; assuming that for k of them (where $1 \leq k \leq n-1$) the computed team impact is positive, k agents send a message to the requesting agent, at cost ks.

3. The agent sends an acceptance message to one of the candidates, at cost s.

Thus, during one successful help transaction the team as a whole spends an amount of resources equal to:

$$C_s(MAP) = (c + b) + ((n-1)c + ks) + s = nc + (k+1)s + b$$

In the worst case, when the broadcast message must be sent as a sequence of unicast messages, and when requesting agent gets bids from each team member, the cost of the team in a successful help transaction equals to:

$$C_w(MAP) = (c + (n-1)s) + ((n-1)c + (n-1)s) + s = nc + (2n-1)s$$

The help transaction can fail if, after computing the team loss, no team member responds to the help request sent by the requesting agent. If the transaction fails, the cost of the team is equal to:

$$C_f(MAP) = c + b + (n-1)c = nc + b$$

Note that in the worst case, the costs are the same for MAP and Always-Help, and for the C_s the costs depend on the parameters k, k_1, and b. Relation to the Proactive Help is not that straightforward and depends on the values of c and s.

5.3 The Configuration Parameter Settings

We initialize the simulation parameters for our experiments as follows. We choose a board of size $(10, 10)$, with six colors; the number of agents per team is eight; and the goal reward is 200 points. The cost vector for each agent includes a high cost of 50 for three of the colors (randomly chosen), and a lower cost for the other three, in each case randomly chosen from the set $\{1, 5, 20, 25\}$. Thus each agent has low capabilities for three types of actions, and high capabilities, to a varying degree, for the other three. The reward for accomplishing each step on the chosen path is 10 points; the initial allocated resources for each step towards the goal is 20 points; the cost of sending a message is initialized to 0.1 points; the cost of computing the team benefit

and loss values is initialized 0.1 points; the overhead cost for performing a helpful action is initialized to 20 points; the disturbance level on the board is initialized to 0 percent; the percentage representing awareness of each-others' abilities is initialized to 100.

During the experiments we calculate the average team scores of the MAP, No-Help, Always-Help, and Proactive-Help methods while varying: the probability representing mutual awareness of abilities (shown as percentage in the graphs); the disturbance level, i.e., the frequency of color changes on the board (also shown as percentage); and the communication and computation costs. For each chosen configuration of the parameters, we calculate the team scores for each of the four methods, averaged over 60,000 simulation runs.

The results of the experiments are presented in the next section. In each run, the behaviors of four agent teams, each using a different help method but otherwise identical to the others, are simulated in parallel. The corresponding agents in four approaches are under exact same constraints, i.e., they choose the same path towards the goal, have the same resources and capabilities, etc. The only difference is that the agents in the four teams use different help methods to collectively achieve their goals.

5.4 The Experimental Results

This section presents the results of experiments in which we vary the level of mutual awareness among the team members; the level of dynamic disturbance in the environment; the cost of communication; and the cost of computation.

5.4.1 The Impact of Mutual Awareness on Team Score

In this section, we compare the MAP protocol with No-Help, Always-Help, and Proactive-Help methods by varying the mutual awareness of agents about each-others' capabilities. We design the same experiments with different values of disturbance, communication costs, and computation costs. Specifically, for disturbance we use low (10 percent) and high (40 percent) values, for communication cost we use low (0.1) and high (1) values, and for computation cost we use low (0.1) and high (1) values.

Figure 5.1 shows the team scores of the compared methods depending on the mutual awareness of the team members about each-others' abilities, when the teams are operating in an environment with low disturbance, in which the computation and communication costs are low. As the figure shows, MAP and No-Help methods do not depend on the percentage of mutual awareness among the team members, as none of them uses unilateral probabilistic reasoning for helpful behavior. However, along with the increase of the awareness percentage of the team members, both Always-Help and Proactive-Help methods noticeably improve their performances, outperforming the No-Help method. As in the Always-Help method the agents perform less communication than in the MAP method, at certain high percentage of mutual awareness (in this case, when the probability is equal to one), Always-Help method produces better results than MAP (note that this result assumes that broadcast is implemented as $n-1$ unicast messages; even with this assumption the worst case performance of both methods is the same, as calculated in Section 5.2). However, in the majority of team models the perfect mutual awareness among the team members does not exist. In addition, even in the case of perfect mutual awareness, the difference between the performances of Always-Help and MAP methods is insignificant.

The behavior of the Proactive-Help method along with the increase of the mutual

awareness is a little different; although the performance of the Proactive-Help team improves significantly, it may not be able to outperform MAP even when perfect awareness among the team member exists. The reason for such a behavior is the property of Proactive-Help agents, according to which each agent performs checks at each turn to observe whether there have been any changes in the team members' paths, and if there are such changes, it computes the impact of the team if it helps the team member. All these computations bring with them additional costs to the Proactive-Help agents, affecting their performance. Because of such additional computations, sometimes even if the perfect awareness among the team members exists, Proactive-Help team may perform no better than MAP.

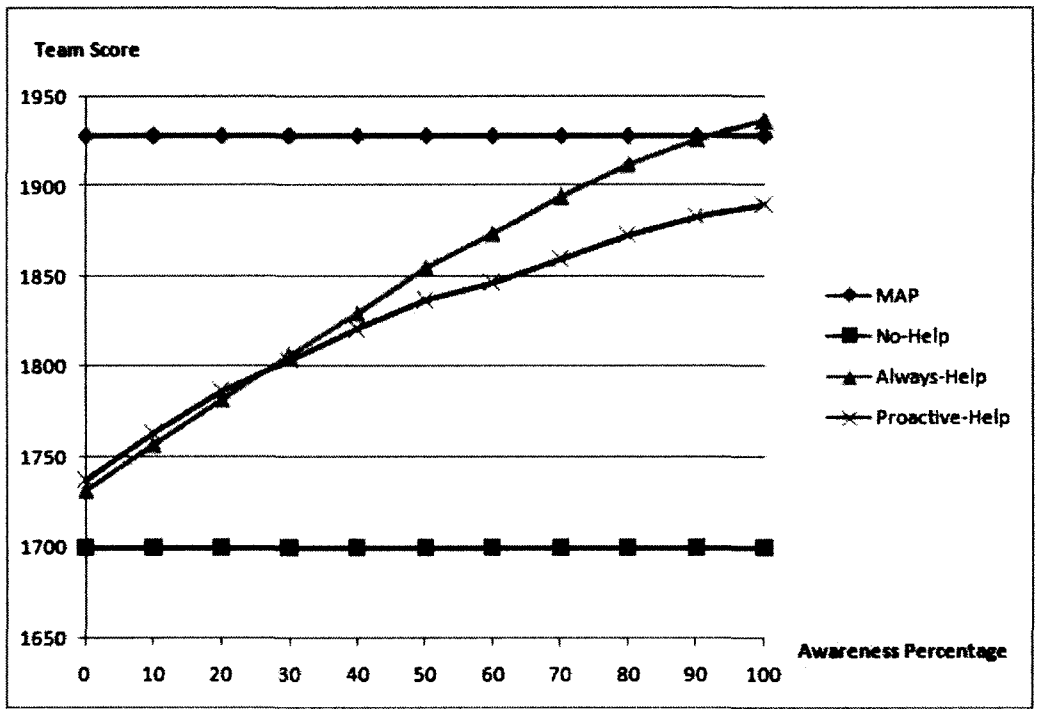

Figure 5.1: The Impact of Mutual Awareness on Team Score in the case of Low Disturbance, Low Communication Cost, and Low Computation Cost

Figure 5.2 presents the analogous performances of the compared methods when the disturbance is low, computation cost is low, and the communication cost is high. When the awareness percentage becomes close to 100, because of its excessive number

of communications associated with high cost, MAP produces slightly worse results compared to Always-Help and Proactive-Help approaches. However, MAP continues to be dominant in awareness percentages lower than 90 percent. From the figure we also conclude that in the case of high communication cost and low computation cost, the Proactive-Help method performs better than the Always-Help method, in low and moderate percentages of mutual awareness. When the mutual awareness among the members becomes close to 100 percent, Always-Help method outperforms the Proactive-Help method because of the more optimal choice of the helping agent.

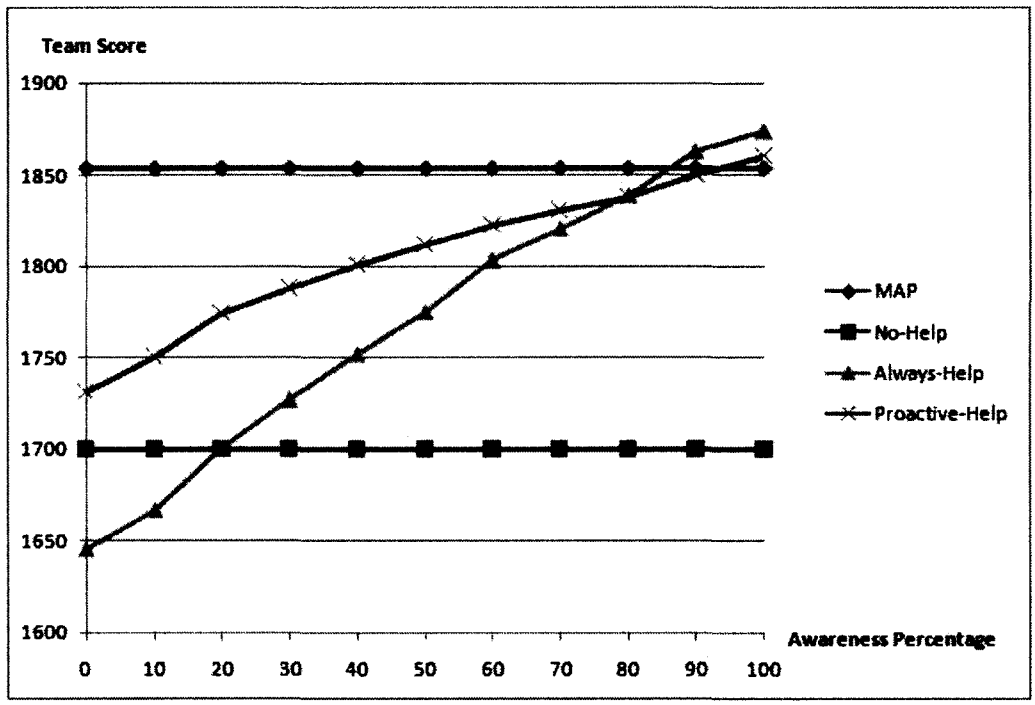

Figure 5.2: The Impact of Mutual Awareness on Team Score in the case of Low Disturbance, High Communication Cost, and Low Computation Cost

Figures 5.3 presents the performances of the teams when the disturbance is low, the computation cost is high, and the communication cost is low. As seen in the figure, when the computation costs are high, the Proactive-Help method performs even worse than the No-Help approach, despite the increase of the percentage of mutual awareness. The reason for such a result for Proactive-Help method is the

excessive amount of computation during the team operation, which, because of the high computation cost, affects the team performance dramatically.

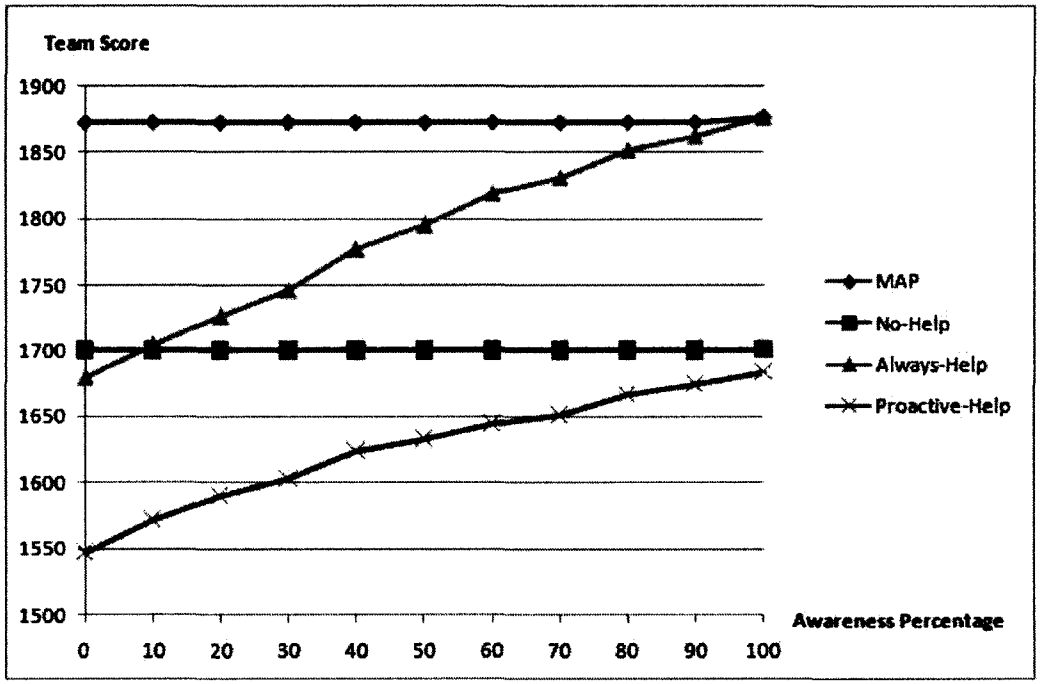

Figure 5.3: The Impact of Mutual Awareness on Team Score in the case of Low Disturbance, Low Communication Cost, and High Computation Cost

Figure 5.4 presents the team scores of the methods when the disturbance is low, the computation and communication costs are high. In these settings, the Proactive-Help method produces the worst results because of its excessive number of computations associated with high cost. The Always-Help method performs worse than the No-Help method in low awareness percentages, but dramatically increases its performance once the mutual awareness among the team members increases. This is justified with the observation that along with the increase of the awareness probability, the decisions of Always-Help method become closer to optimal, in the meantime taking the same amount of resources in deliberation and communication.

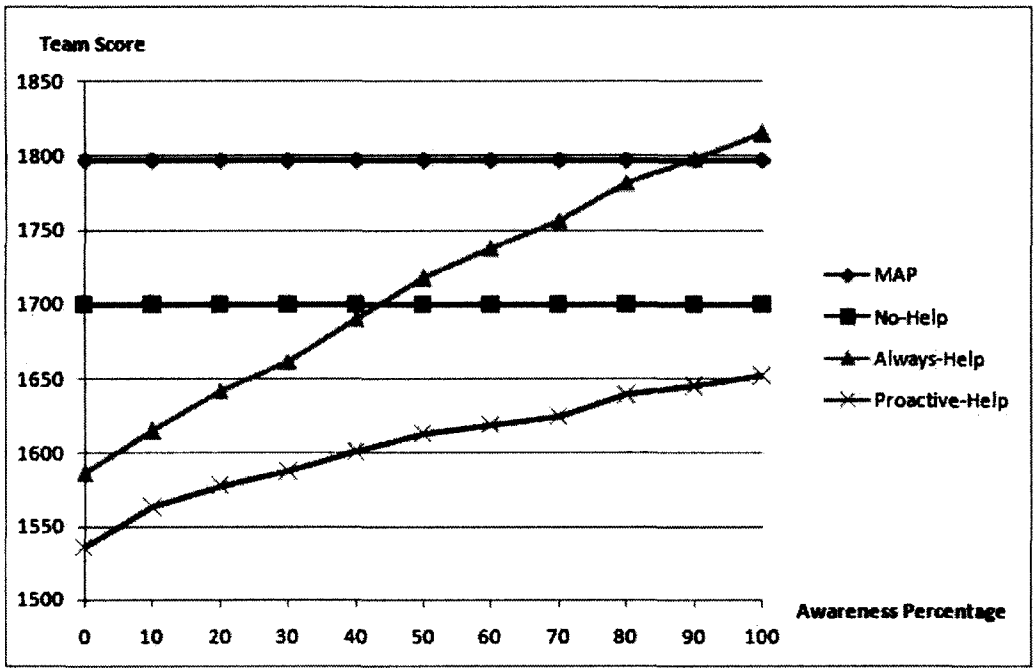

Figure 5.4: The Impact of Mutual Awareness on Team Score in the case of Low Disturbance, High Communication Cost, and High Computation Cost

In both latter cases, MAP continues to be dominant over all other approaches. Only when the mutual awareness percentage is 90 percent or 100 percent, the Always-Help method performs equal or slightly better results because of the fewer communication among the members.

The analogous experiments in case of the high disturbance in the environment are presented in Figures 5.5, 5.6, 5.7, 5.8. The experiments show that, despite the higher disturbance, along with the increase of the mutual awareness the compared approaches exhibit a behavior similar to the case of the low disturbance. Thus, the critical factors in the success of the compared approaches are the computation and communication costs among the team members.

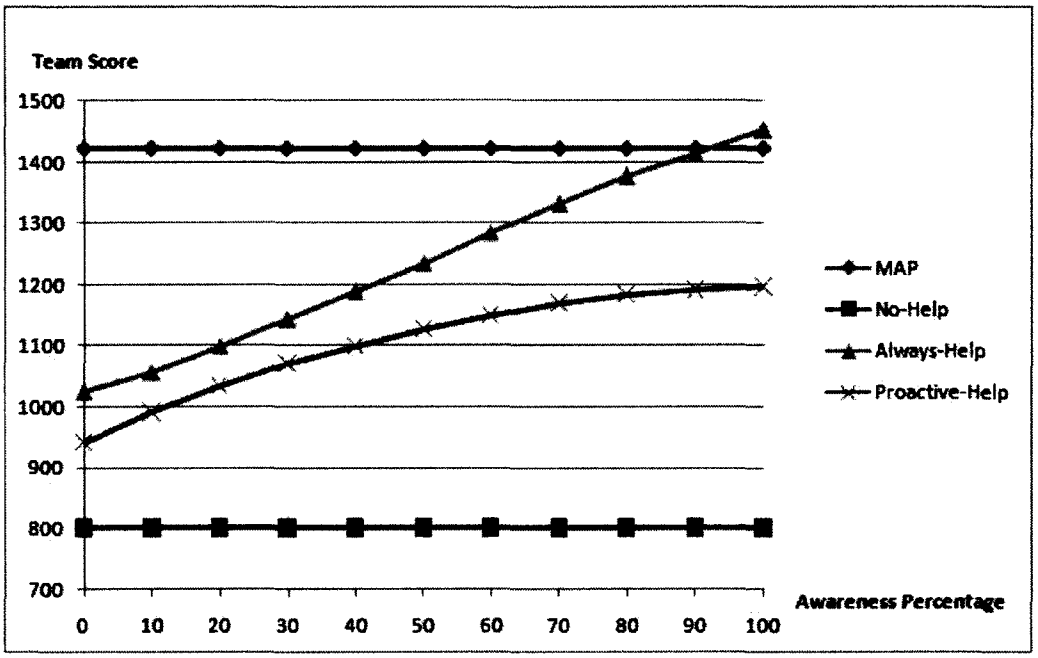

Figure 5.5: The Impact of Mutual Awareness on Team Score in the case of High Disturbance, Low Communication Cost, and Low Computation Cost

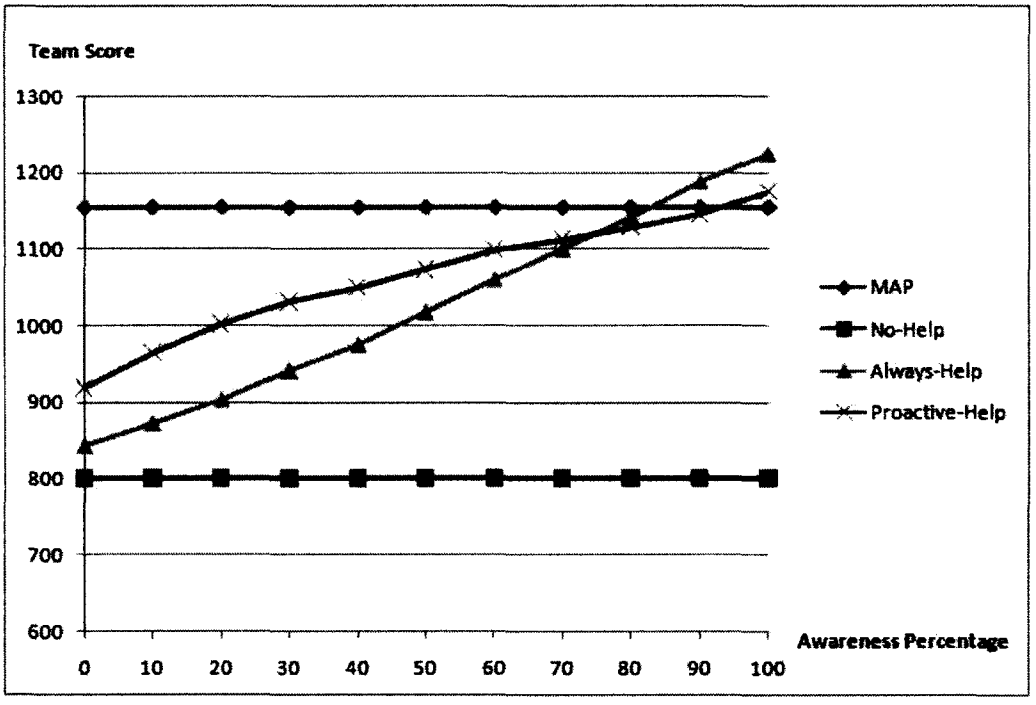

Figure 5.6: The Impact of Mutual Awareness on Team Score in the case of High Disturbance, High Communication Cost, and Low Computation Cost

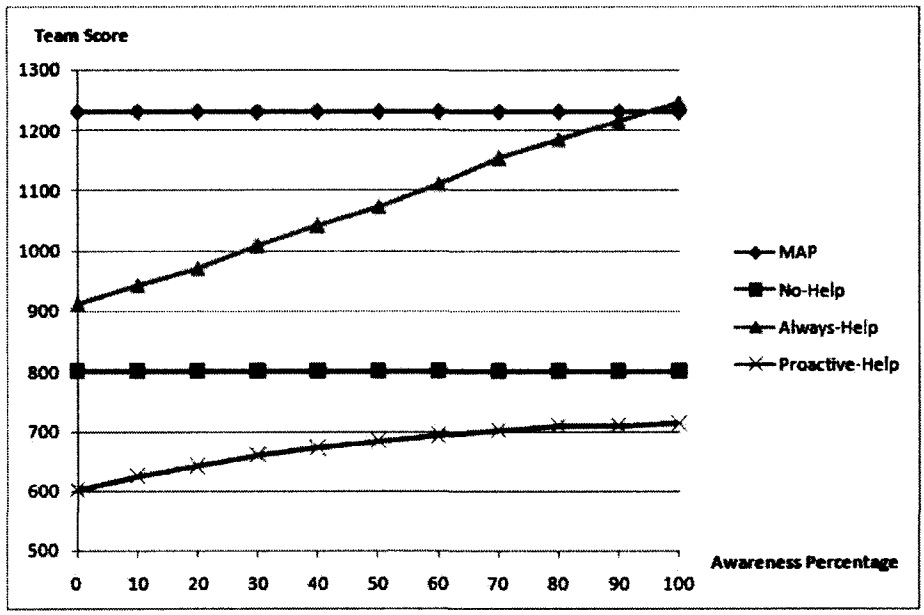

Figure 5.7: The Impact of Mutual Awareness on Team Score in the case of High Disturbance, Low Communication Cost, and High Computation Cost

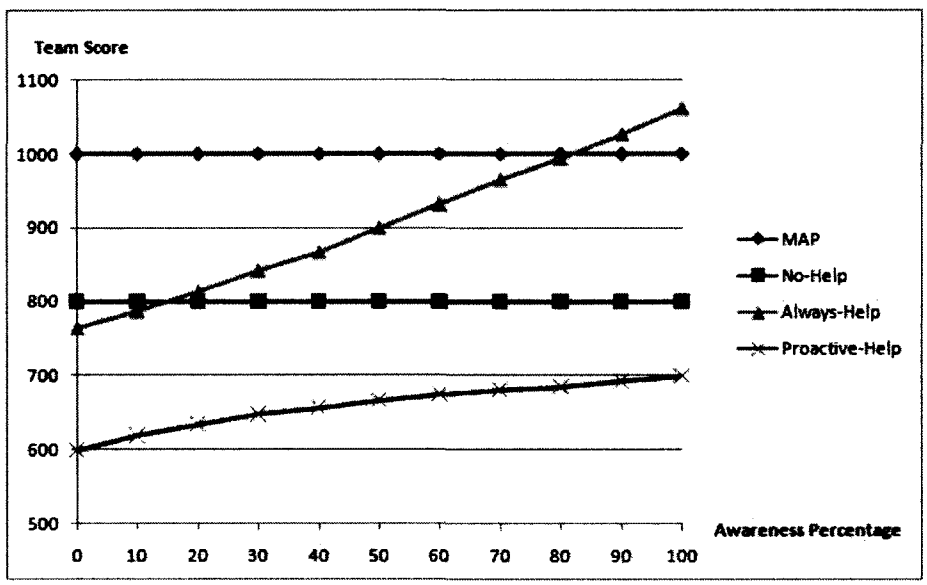

Figure 5.8: The Impact of Mutual Awareness on Team Score in the case of High Disturbance, High Communication Cost, and High Computation Cost

5.4.2 The Impact of Disturbance on Team Score

In this section, we compare the MAP protocol with No-Help, Always-Help, and Proactive-Help methods by varying the disturbance on the board. We design the same experiments with different values of mutual awareness percentages among the agents, communication costs, and computation costs. Specifically, for mutual awareness we use moderately low (30 percent) and moderately high (70 percent) values, for communication and computation costs we use low (0.3) and high (1) values. Note that the use of low value of 0.3 as opposed to the previously used low value 0.1 does not alter the behavior represented by the graphs, but provides their better separation.

Figure 5.9 presents the team scores of the compared methods depending on the disturbance in the environment, when the mutual awareness of the team members is moderately low, and the computation and communication costs are low. As seen in the figure, along with the increase of the disturbance, MAP performs better compared to all other approaches. The Always-Help method produces slightly better results than the Proactive-Help method, as along with the increase of the disturbance the Proactive-Help agents perform more computations for reasoning about whether to help. However, as the computation cost is not high, such additional computations by Proactive-Help agents still guarantee their better performance compared to the No-Help method.

Figure 5.10 presents the analogous performances of the methods with high communication and computation costs. Here, too, along with the increase of the disturbance, MAP outperforms all other methods. Because of the high computation and communication costs, Proactive-Help approach produces the worst results, whereas above certain disturbance percentage, the Always-Help approach outperforms the No-Help approach because of the more frequent inefficiency of the latter.

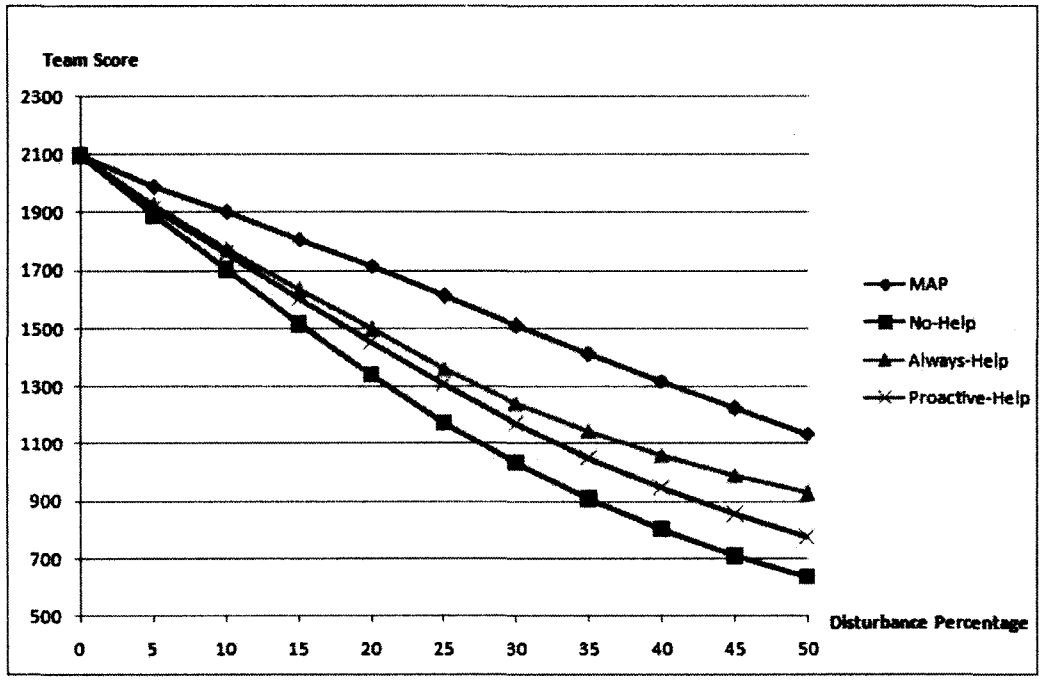

Figure 5.9: The Impact of Disturbance on Team Score in the case of Moderately Low Mutual Awareness, Low Communication cost, and Low Computation Cost

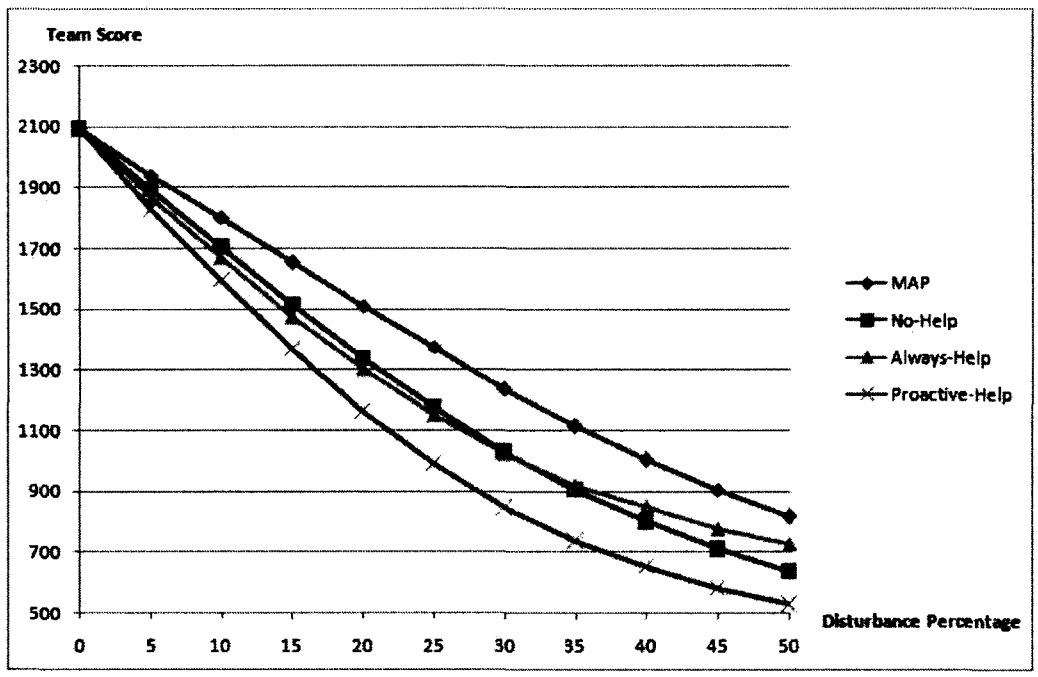

Figure 5.10: The Impact of Disturbance on Team Score in the case of Moderately Low Mutual Awareness, High Communication cost, and High Computation Cost

The simulation results in the case of high communication cost and low computation cost are similar to the results presented in Figure 5.9, whereas the simulation results in the case of low communication cost and high computation costs are similar to the results presented in Figure 5.10.

Figures 5.11 and 5.12 present the analogous team scores in which the mutual awareness of the team members is higher. The experiments show that despite the changes of the awareness level, the methods exhibit similar behavior when the disturbance in the environment changes.

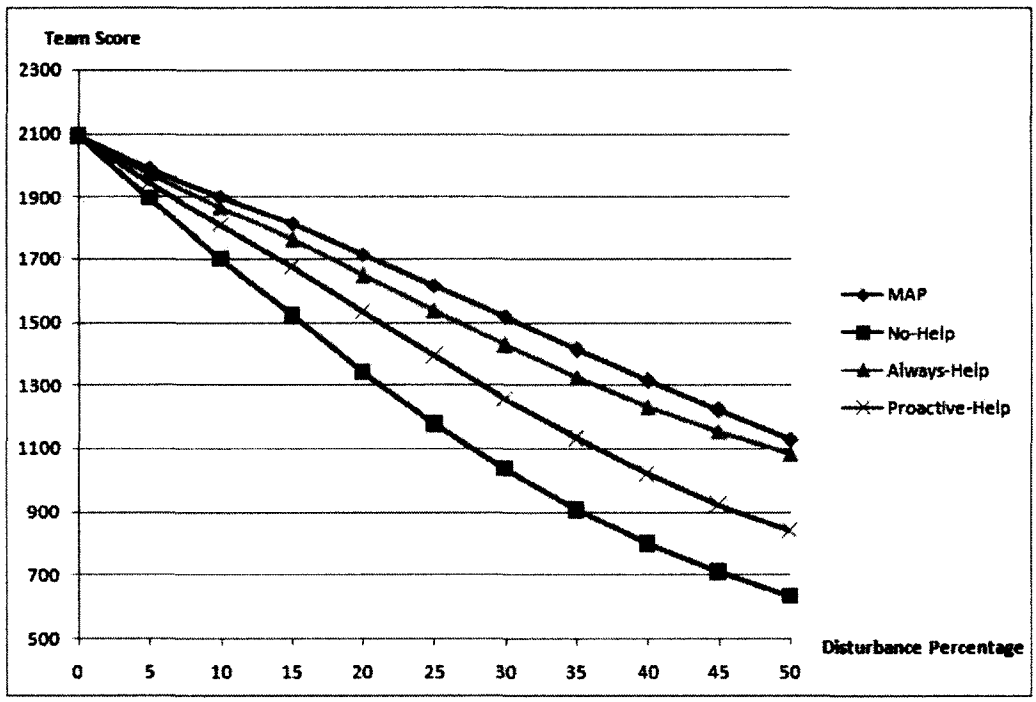

Figure 5.11: The Impact of Disturbance on Team Score in the case of Moderately High Mutual Awareness, Low Communication cost, and Low Computation Cost

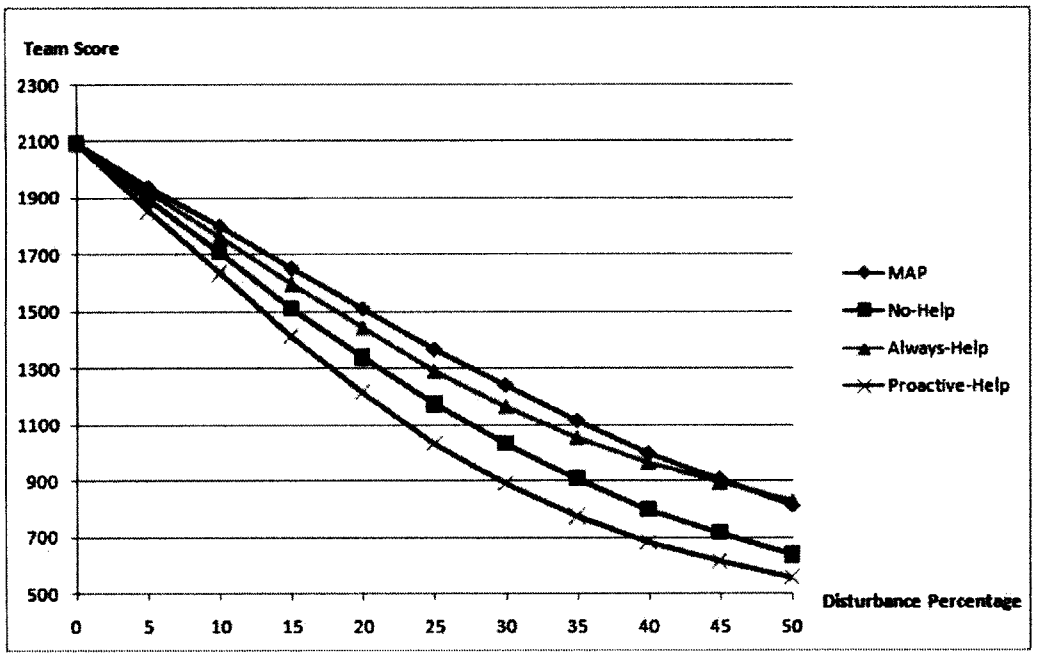

Figure 5.12: The Impact of Disturbance on Team Score in the case of Moderately High Mutual Awareness, High Communication cost, and High Computation Cost

The results shown in Figures 5.9 to 5.12 are influenced by the fact that in the simulation the mechanism for recognizing the need for help is triggered by the perceived changes in the environment. This explains why all four methods have the same team score at disturbance level zero.

The simulation results in the case of high communication cost and low computation cost are similar to the results presented in Figure 5.11, whereas the simulation results in the case of low communication cost and high computation costs are similar to the results presented in Figure 5.12.

The above-presented experiments confirm the advantage of MAP over other methods in cases of moderately low and moderately high awareness probabilities, regardless of the disturbance level in the environment (assuming that the changes in the environment are moderate and are not such that there might be a need for team reorganization).

5.4.3 The Impact of Communication Cost on Team Score

In this section, we compare the MAP protocol with No-Help, Always-Help, and Proactive-Help methods by varying the value of the communication cost among agents. We design the same experiments with different values of mutual awareness among the agents, and different values of computation costs. Specifically, for mutual awareness we use moderately low (30 percent) and moderately high (70 percent) values, for computation cost we use low (0.3) and high (1) values. We perform the experiments in environments with moderate disturbances (30 percent).

Figure 5.13 presents the team scores of the compared methods depending on the communication cost among the team members, when the awareness of the team members about each-other is moderately low, and the computation cost is moderately low. As the figure shows, the increase of the communication cost in teams results in degraded performance of MAP, Always-Help, and Proactive-Help methods. However, as the Proactive-Help method uses less communication than MAP and Always-Help methods, its results are not affected significantly. MAP agents communicate slightly more than the Always-Help agents, but the more optimal decisions of the MAP team compared to the Always-Help team compensate the overhead of communication, resulting in approximately equal amount of team points loss compared to the Always-Help team. When the communication cost increases significantly, the performance of Always-Help method becomes worse than the performance of the Proactive-Help method. The reason for such a behavior is that the Always-Help team members perform more communications associated with high cost, than the Proactive-Help members.

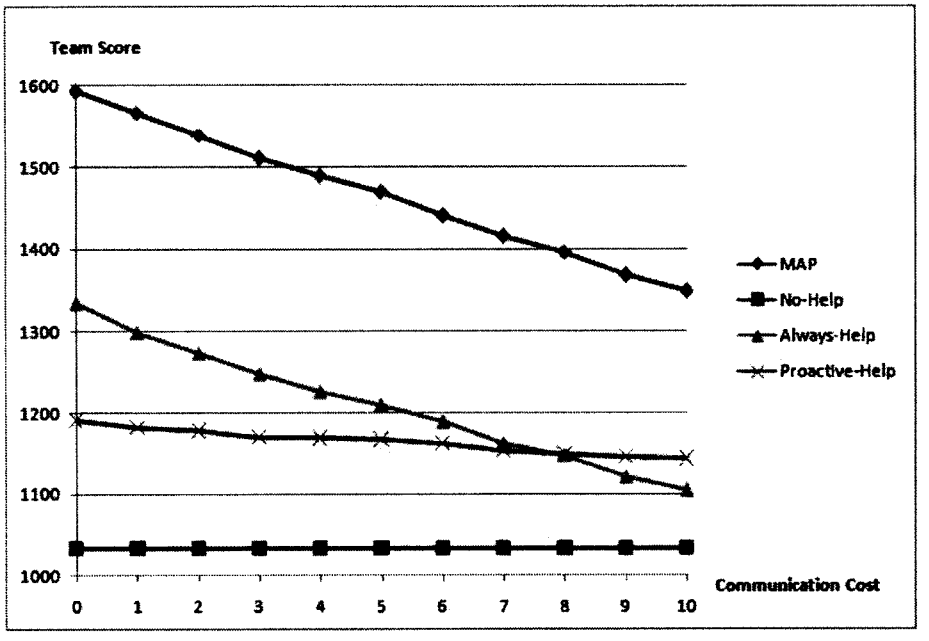

Figure 5.13: The Impact of Communication Cost on Team Score in the case of Moderately Low Mutual Awareness and Low Computation Cost

Figure 5.14 shows the team scores of analogous experiments in which the computation cost among the team members is high. Although the behaviors of the compared methods are the same as in the case of low computation costs, in this case the Proactive-Help method performs the worst. In addition, the No-Help method outperforms the Always-Help method when the communication costs become very high. The rationale for such results is explained in the high cost of the computation and communication among the team members. In such cases, it may be preferred for the team to operate without any probabilistic guesses and helpful behaviors, as the questionable outcome of such a help may not justify the spent resources for performing such a help. As the decisions of the MAP team are accurate, the help using MAP method is guaranteed to be beneficial for the team, unless the computation and communication costs are extremely high.

Figures 5.15 and 5.16 present the performances of the methods in the analogous experiments when the mutual awareness of the team members is higher. Because of

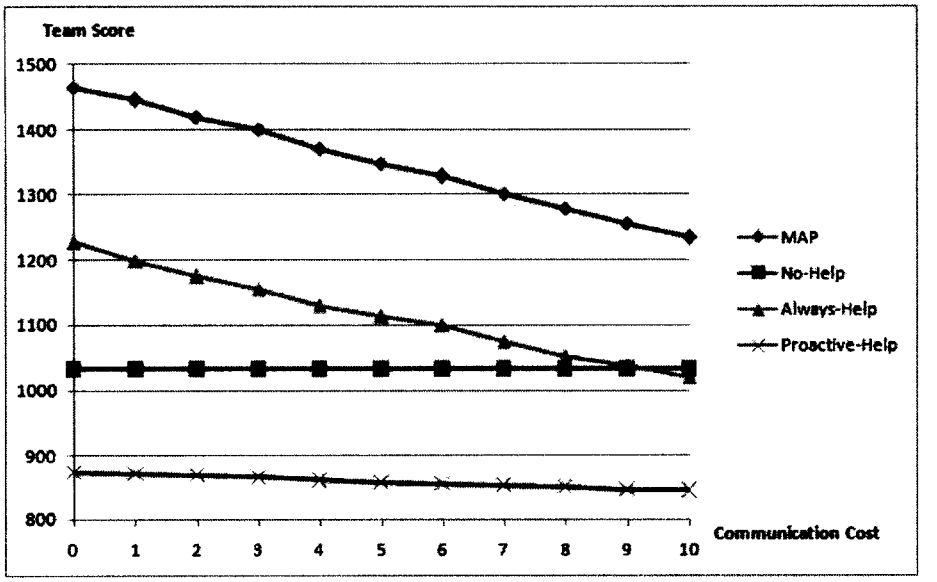

Figure 5.14: The Impact of Communication Cost on Team Score in the case of Moderately Low Mutual Awareness and High Computation Cost

the high mutual awareness, when the communication cost increases, and the computation cost is low (Figure 5.15) both Always-Help and Proactive-Help methods perform better than the No-Help approach. However, when the communication cost increases with the computation cost being high (Figure 5.16), the Proactive-Help method performs worse than the No-Help method, despite the higher level of awareness.

5.4.4 The Impact of Computation Cost on Team Score

In this section, we compare the MAP protocol with No-Help, Always-Help, and Proactive-Help methods by varying the value of the computation cost among agents. We design the same experiments with different values of mutual awareness among the agents, and different values of communication costs. Specifically, for mutual awareness we use moderately low (30 percent) and moderately high (70 percent) values, for communication cost we use low (0.3) and high (1) values. We perform the experiments in environments with moderate disturbances (30 percent).

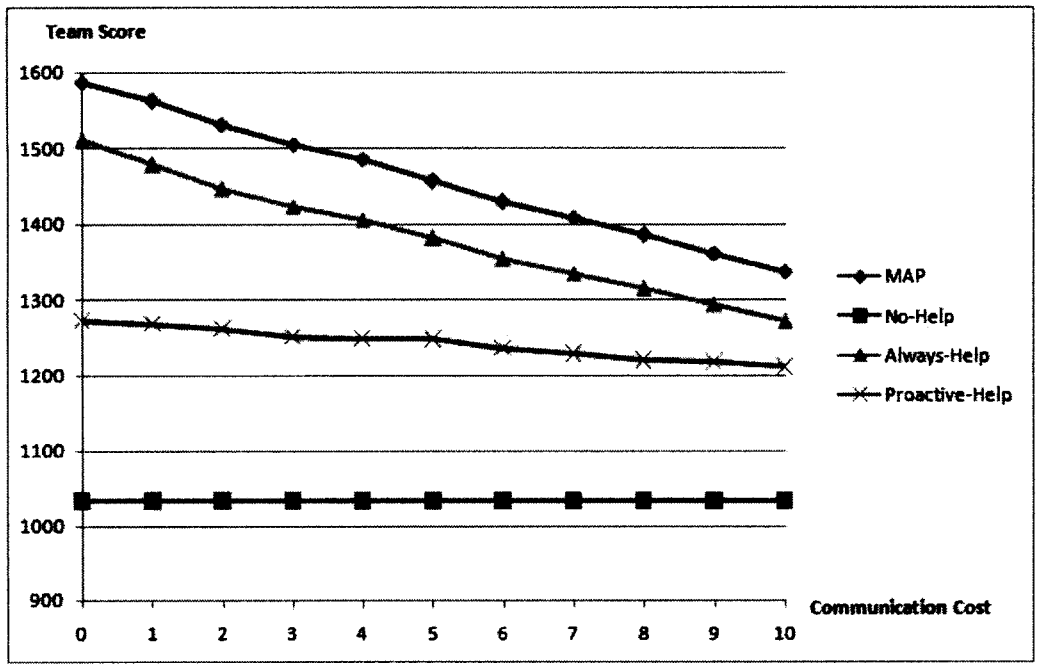

Figure 5.15: The Impact of Communication Cost on Team Score in the case of Moderately High Mutual Awareness and Low Computation Cost

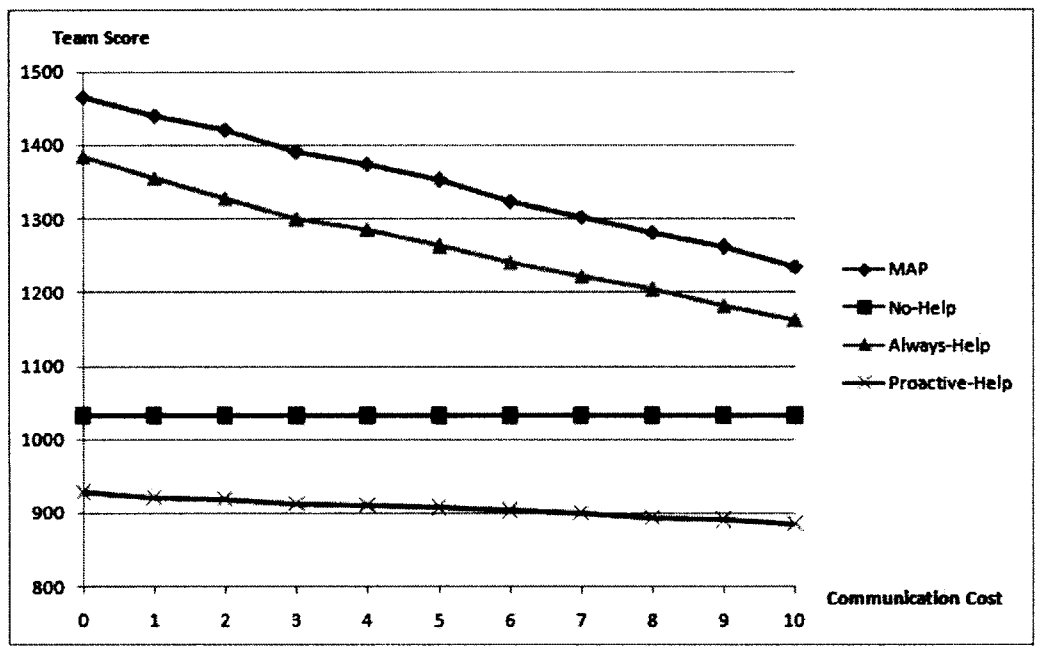

Figure 5.16: The Impact of Communication Cost on Team Score in the case of Moderately High Mutual Awareness and High Computation Cost

Figure 5.17 exposes the performance of the compared methods depending on the computation costs of the team members when reasoning about the help needs of each-other and team benefits. The figure presents the team scores in the case when the mutual awareness of the team members moderately low, and the communication cost is low. As the figure shows, the increase of the computation cost in teams decreases the performance of MAP, Always-Help, and Proactive-Help methods. However, as the Proactive-Help method uses more computations when checking whether anyone needs help, and when computing the team impact of potentially every team member for performing its next move, along with the increase of the computation costs the results of the Proactive-Help method are affected dramatically, at some point even producing worse results than the No-Help method. As MAP and Always-Help methods perform the same amount of computations - much less than the Proactive-Help method, their results are not degraded significantly.

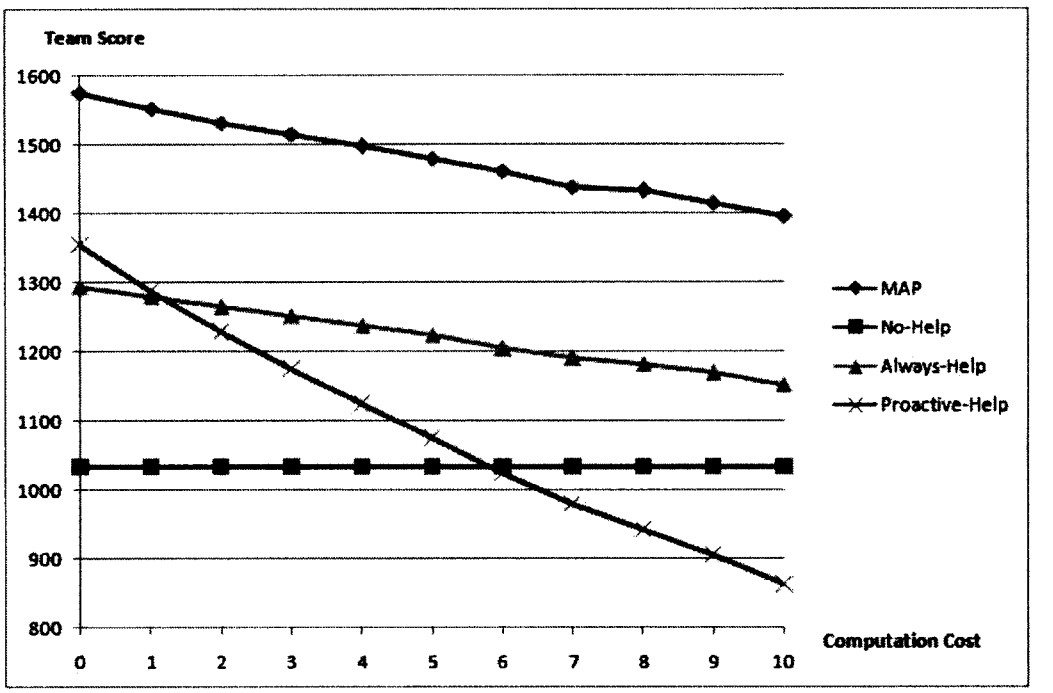

Figure 5.17: The Impact of Computation Cost on Team Score in the case of Moderately Low Awareness and Low Communication Cost

Figure 5.18 presents the team scores of the compared methods when the communication cost among the team members is high. MAP outperforms the other methods in all cases. The Proactive-Help method is more preferable than the Always-Help and No-Help approaches in cases of low computation and high communication costs, but quickly becomes worse than both of them, as the computation cost increases.

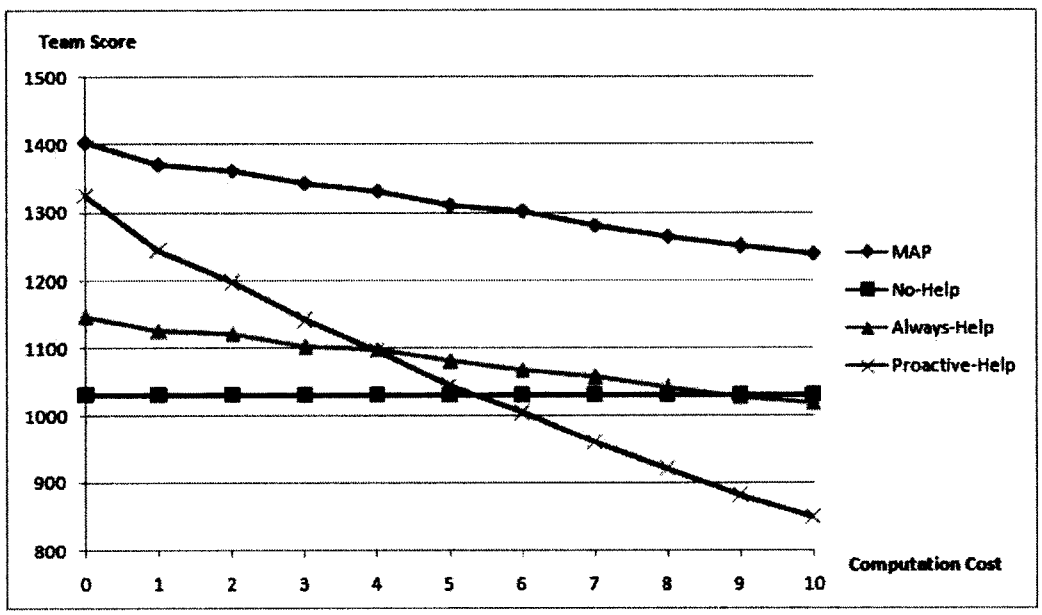

Figure 5.18: The Impact of Computation Cost on Team Score in the case of Moderately Low Awareness and High Communication Cost

Figures 5.19 and 5.20 show the team scores of analogous experiment results when the mutual awareness among the team members is higher. Despite the change of the awareness level, all methods behave the same way as in lower awareness case.

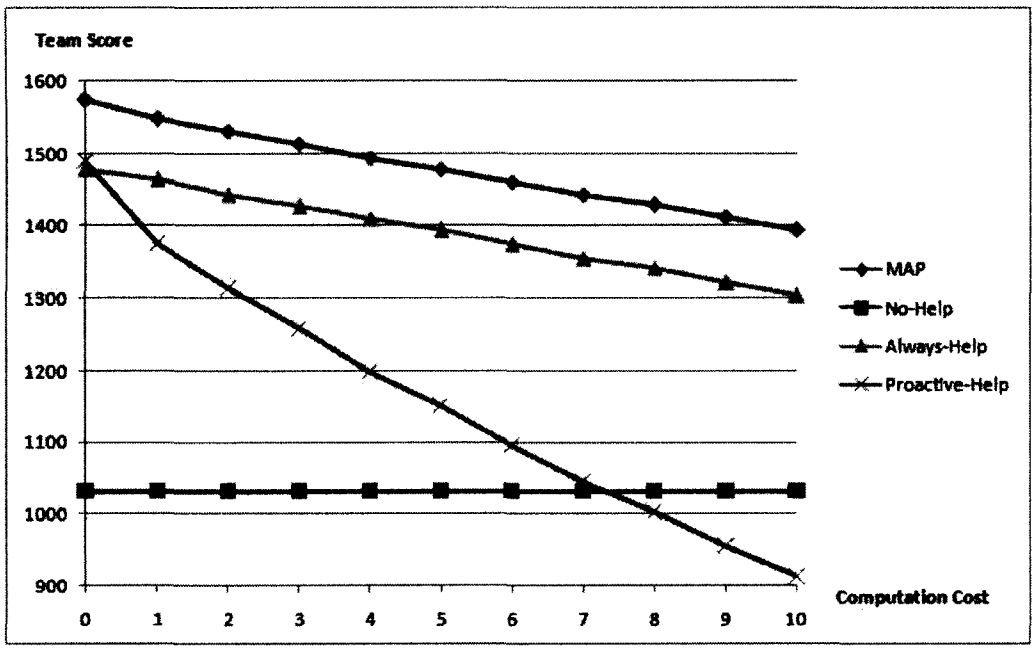

Figure 5.19: The Impact of Computation Cost on Team Score in the case of Moderately High Awareness and Low Communication Cost

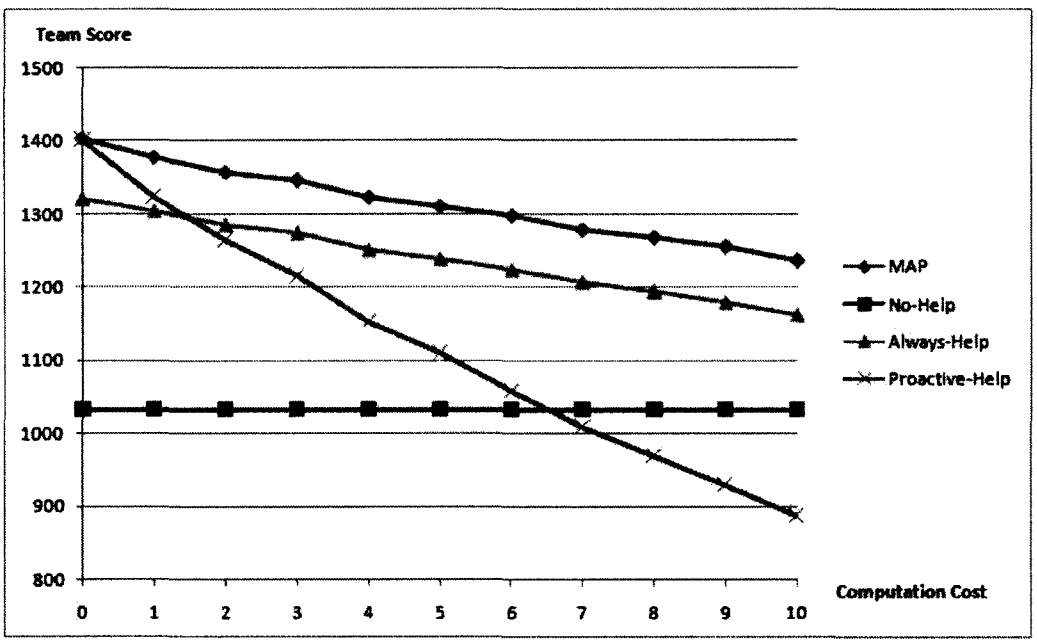

Figure 5.20: The Impact of Computation Cost on Team Score in the case of Moderately High Awareness and High Communication Cost

5.4.5 Summary of the Evaluation Results

The presented experiments show the dominance of MAP over the other compared methods when the mutual awareness among the team members is not close to perfect. In the case of perfect awareness among the team members, the Always-Help approach performs slightly better than MAP because of its fewer usage of communication. For the Proactive-Help approach, even in the case of perfect awareness, the relative advantage over MAP highly depends on the computation and communication costs among the team members, as the Proactive-Help method uses less communication than MAP, but performs a large number of computations and checks at each move. However, it is worth noting that in real teams having perfect awareness about each other during the team operation entails additional communication, which is not modeled in our experiments. This observation leads to believe that in real teams with high communication costs, the better performances of Always-Help or Proactive-Help approaches compared to MAP may be unrealistic, as the cost of maintaining a near-perfect awareness about each other would likely be prohibitive.

For the success of the Proactive-Help method, a critical factor is the value of computation cost among the team members. While in the low values of computation cost the Proactive-Help method can produce equal or slightly worse results compared to MAP, with the increase of the computation cost its performance degrades significantly, sometimes even performing worse than the No-Help method.

The Always-Help method improves its performance dramatically when the mutual awareness among the team members increases. Its performance also highly depends on the value of the communication cost, whereas the change of the computation cost has less impact on the Always-Help method.

The No-Help method never outperforms MAP in any of the experimental settings

(unless the communication and computation costs are extremely high). However, depending on the awareness level among the team members, as well as the communication and computation costs, No-Help method may produce better results than Always-Help and Proactive-Help methods, as the latter ones may spend additional resources on computing and communicating, while not arriving at optimal decisions.

Chapter 6

Conclusions and Future Work

This thesis proposes a novel protocol, called the Mutual Assistance Protocol (MAP), for incorporating helpful behavior into multiagent teamwork. Initial research has included a study of literature in several areas of multiagent systems (MAS), especially agent teamwork, agent protocols, and helpful behavior in agent teams. The study led to an observation that, despite the growing use of protocols in MAS, there is a shortage of protocols designed for MAS teamwork, and particularly for incorporating helpful behavior into MAS teamwork. Another observation was that some of the existing approaches to helpful behavior in teamwork enable agents to provide help based on unilateral probabilistic beliefs of a single agent, which in many realistic teamwork environments may be inaccurate. These observations motivated the design of a new protocol for helpful behavior in teamwork, with a particular attention to the choice of individual agents' beliefs involved in the help decision.

In MAP, helpful behavior occurs when an agent uses its own abilities and resources to advance a subtask assigned to another agent. Similar to the bidding sequence of the Contract Net Protocol, the agent that needs help broadcasts a request, receives offers from teammates willing to help, and chooses the most suitable offer. The helpful act is performed only when the two agents, based on their own beliefs, determine that

it is in the interest of the team. When pondering possible help, each agent assesses the team impact of changing its current local plan to a new plan that includes the helpful act. The underlying design philosophy is that each agent, in its mainstream behavior, regularly assesses the team impact of its alternative local plans; thus, insofar as its individual beliefs can effectively support its mainstream behavior, they can also effectively support its helpful behavior. As the helpful act may consist of either performing actions or granting additional resources, two MAP versions, called the Action Map and Resource Map, have been developed to address these two aspects separately.

MAP was then analyzed in terms of the complexity of the resource costs during the helpful act transaction. In addition, Action MAP was submitted to a test of how well it performs compared to approaches with no helpful behavior or with probabilistic unilateral decision mechanisms, using an implemented simulation game. The advantages of MAP over protocols that use unilateral help decisions were demonstrated through simulation experiments, using varying levels of mutual awareness in the team, dynamic disturbance in the environment, communication costs, and computation costs.

The analysis and experiments suggest that MAP indeed increases the effectiveness of teamwork, and is superior compared to unilateral decision mechanisms for helpful behavior, especially in cases when the beliefs of the team members about each others' abilities and activities may not be accurate.

The thesis includes two variations of MAP that need to be further explored. In one of them, the agent requesting help can be involved in multiple simultaneous (related or independent) MAP transactions. The other is the Helper-Initiated MAP, that starts with a broadcast by an agent willing to offer help.

In MAP, an agent deliberating about help relies on its local beliefs, acquired through perception, as well as its context beliefs, acquired through communication with the rest of the team. While this thesis did not explore the formation and maintenance of context beliefs, this topic is relevant in MAP implementation and leads to interesting architectural questions that merit further study.

Bibliography

Huib Aldewereld, Wiebe van der Hoek, and John jules Meyer. Rational Teams: Logical Aspects of Multi-Agent Systems. Technical report, Utrecht University, 2004.

Elisabeth Ball and Michael Butler. Using Decomposition to Model Multi-agent Interaction Protocols in Event-B. In *FM'06 Doctoral Symposium*. Springer, 2006.

Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal logic*. Cambridge University Press, New York, NY, USA, 2001.

Michael Bratman. *Intention, Plans, and Practical Reason*. Harvard University Press, 1987.

Jacek Brzeziński, Piotr Dunin-Kęplicz, and Barbara Dunin-Kęplicz. Collectively Cognitive Agents in Cooperative Teams. In *Engineering Societies in the Agents World V*, pages 191–208. Springer, 2005.

Jannis A Cannon-Bowers, Eduardo Salas, and Sharolyn Converse. Shared Mental Models in Expert Team Decision Making. In *Individual and Group Decision Making: Current issues*, pages 221–246. Erlbaum, 1993.

Sen Cao, Richard A. Volz, Thomas R. Ioerger, and Michael S. Miller. On Proactive Helping Behaviors in Teamwork. In *Proceedings of the International Conference on Artificial Intelligence*, 2005.

Kay-Yut Chen, Tad Hogg, and Bernardo Huberman. Behavior of Multi-Agent Protocols using Quantum Entanglement. In *Proceedings of AAAI-2007 Spring Symposium on Quantum Interaction*, 2007.

Philip R. Cohen and Hector J. Levesque. Intention is Choice with Commitment. *Artificial Intelligence*, pages 213–261, 1990.

Philip R. Cohen and Hector J. Levesque. Teamwork. *Special Issue on Cognitive Science and Artificial Intelligence*, pages 487–512, 1991.

Nancy J. Cooke, Eduardo Salas, Janis A. Cannon-Bowers, and Rene'e Stout. Measuring Team Knowledge. *Human Factors*, pages 153–173, 2000.

Daniel Corkill. Blackboard Systems. *AI Expert*, 6(9):40–47, 1991.

Eric M. Dashofy, André van der Hoek, and Richard N. Taylor. Towards Architecture-Based Self-healing Systems. In *Proceedings of the First Workshop on Self-healing Systems*, WOSS '02, pages 21–26, New York, NY, USA, 2002. ACM.

Chrysanthos Dellarocas and Mark Klein. Designing Robust, Open Electronic Marketplaces Of Contract Net Agents. In *Proceedings of the 20th International Conference on Information Systems*, 1999.

Thomas E. Downing, Scott Moss, and Claudia Pahl-Wostl. Understanding Climate Policy Using Participatory Agent-Based Social Simulation. In *Proceedings of the Second International Workshop on Multi-Agent-Based Simulation*, pages 198–213, London, UK, 2001. Springer-Verlag.

Barbara Maria Dunin-Keplicz and Rineke Verbrugge. *Teamwork in Multi-Agent Systems: A Formal Approach.* Wiley, 2010.

Hywel Dunn-Davies, Jim Cunningham, and Shamimabi Paurobally. Propositional Statecharts for Agent Interaction Protocols. *Electronic Notes in Theoretical Computer Science*, 134:55–75, 2005.

Edmund Durfee, Victor Lesser, and Daniel Corkill. Trends in Cooperative Distributed Problem Solving. *IEEE Transactions on Knowledge and Data Engineering*, 1(1): 63–83, 1989.

Edmund H. Durfee. Distributed problem solving and planning. In Gerhard Weiss, editor, *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence*, chapter 3, pages 121–164. The MIT Press, 1999.

Xiaocong Fan, John Yen, and Richard A. Volz. A Theoretical Framework on Proactive Information Exchange in Agent Teamwork. *Artificial Intelligence*, 169:23–97, 2005.

Amos Fiat, Yishay Mansour, and Uri Nadav. Efficient Contention Resolution Protocols for Selfish Agents. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 179–188. Society for Industrial and Applied Mathematics, 2007.

Foundation of Intelligent Physical Agents. FIPA Dutch Auction Interaction Protocol Specification. *FIPA TC Communication 00032*, 2000.

Foundation of Intelligent Physical Agents. FIPA English Auction Interaction Protocol Specification. *FIPA TC Communication 00031*, 2001a.

Foundation of Intelligent Physical Agents. *FIPA Specification Part 2 - Agent Communication Language*, 1997.

Foundation of Intelligent Physical Agents. FIPA contract net interaction protocol specification. *FIPA TC Communication 00029*, 2001b.

Ya'akov Gal, Barbara Grosz, Sarit Kraus, Avi Pfeffer, and Stuart Shieber. Agent decision-making in open mixed networks. *Artificial Intelligence*, 174(18):1460 – 1480, 2010.

Michael Georgeff and Amy Lansky. Reactive Reasoning and Planning. In *Proceedings of the Ninth National Conference an Artificial Intelligence*, pages 677–682, 1987.

Mark Greaves, Heather Holmback, and Jeffrey Bradshaw. What Is a Conversation Policy? *Issues in Agent Communication*, pages 118–131, 2000.

Barbara Grosz, Sarit Kraus, and Shavit Talman. The Influence of Social Dependencies on Decision-Making: Initial Investigations with a New Game. In *Autonomous Agents and Multiagent Systems*, 2004.

Barbara J. Grosz and Sarit Kraus. Collaborative Plans for Complex Group Action. *Artificial Intelligence*, 86:269–357, 1996.

Aaron Helsinger and Todd Wright. Cougaar: A Robust Configurable Multi Agent Platform. In *Aerospace, 2005 IEEE Conference*, pages 1–10, 2005.

Randall W. Hill, Jonathan Gratch, Stacy Marsella, Jeff Rickel, William R. Swartout, and David R. Traum. Virtual Humans in the Mission Rehearsal Exercise System. *KI Embodied Conversational Agents*, 2003.

Hoffman, Shadbolt, Burton A. Mike, and Klein Gary. Eliciting Knowledge from Experts: A Methodological Analysis. *Organizational Behavior and Human Decision Processes*, 62(2):129–158, 1995.

George Hughes and Maxwell John Cresswell. *A New Introduction to Modal Logic.* Routledge, 1996.

Goichi Itabashi, Yoshiaki Haramoto, Yasushi Kato, Kaoru Takahashi, and Norio Shiratorii. Specification and Analysis of the Contract Net Protocol Based on State Machine Model. In *Special Section on Concurrent System Technology and its Application to Multiple Agent Systems*, 2002.

Takayuki Ito, Hiromitsu Hattori, and Mark Klein. Multi-Issue Negotiation Protocol for Agents: Exploring Nonlinear Utility Spaces. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 1347–1352. Morgan Kaufmann Publishers Inc., 2007.

Hideshi Itoh. Incentives to Help in Multi-agent Situations. *Econometrica*, 59(3), 1991.

Java Agent Development Framework. http://jade.tilab.com/doc/api/jade/proto/package-summary.html. *FIPA Standard Protocols in JADE*, 2004.

Nicholas Jennings. The ARCHON System and its Applications. In *2nd International Conference on Cooperating Knowledge Based Systems*, pages 13–29, 1994.

Nicholas. R. Jennings, Ebrahim Mamdani, Inaki Laresgoiti, J. Perez, and J. Corera. Grate: A general framework for cooperative problem solving. *IEE-BCS Journal of Intelligent Systems Engineering*, 1(2):102–114, 1992.

Ece Kamar, Ya'akov Gal, and Barbara J. Grosz. Incorporating Helpful Behavior into Collaborative Planning. In *Autonomous Agents and Multiagent Systems/Agent Theories, Architectures, and Languages*, pages 875–882, 2009.

Gal Kaminka, Ari Yakir, Dan Erusalimchik, and Nirom Cohen-Nov. Towards Collaborative Task and Team Maintenance. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems - AAMAS '07*. ACM Press, 2007.

Jeffrey A. LePine, Mary Ann Hanson, Walter C. Borman, and Stephan J. Motowidlo. Contextual performance and teamwork: Implications for staffing. *Research in Personnel and Human Resources Management*, 19:53–90, 2000.

Hector Levesque, Philip Cohen, and Jose Nunes. On Acting Together. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 94–99, 1990.

Magnus Ljungberg and Andrew Lucas. The OASIS Air-Traffic Management System. In *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, Seoul, Korea, 1992.

Janusz Marecki, Nathan Schurr, and Milind Tambe. *Agent-Based Simulations for Disaster Rescue Using the DEFACTO Coordination System*, pages 281–297. John Wiley and Sons, Inc., 2005.

James Mayfield, Yannis K Labrou, and Tim Finin. Evaluation of KQML as an Agent Communication Language. In *Intelligent Agents II*, volume 1037, pages 347–360. Springer-Verlag, 1996.

Maria Miceli, Amedeo Cesta, and Paola Rizzo. Autonomous Help in Distributed Work Environments. In *Proceedings of the Seventh European Conference on Cognitive Ergonomics*, pages 367–377, 1994.

Tim Miller and Peter McBurney. On Illegal Composition of First-Class Agent Interaction Protocols. In *Proceedings of the thirty-first Australasian conference on Computer science - Volume 74*, ACSC '08, pages 127–136. Australian Computer Society, Inc., 2008.

Susan Mohammed and Brad C. Dumville. Team Mental Models in a Team Knowledge Framework: Expanding Theory and Measurement Across Disciplinary Boundaries. *Journal of Organizational Behavior*, 22:89–106, 2001.

Daniel Mountjoy and Bala Ram. Agent-Based Planning Team Training Platform. Technical report, United States Air Force Research Laboratory, 2003.

Shamimabi Paurobally and Jim Cunningham. Verification of Protocols for Automated Negotiation. In *Proceedings of the 15th Eureopean Conference on Artificial Intelligence, ECAI'2002.* IOS Press, 2002.

Shamimabi Paurobally, Jim Cunningham, and Nicholas R. Jennings. Developing agent interaction protocols using graphical and logical methodologies. In *Programming Multi-Agent Systems*, volume 3067, pages 149–168. Springer, 2003.

Shamimabi Paurobally, J. Cunningham, and Nicholas R. Jennings. Verifying the contract net protocol: A case study in interaction protocol and agent communication semantics. In *2nd International Workshop on Logic and Communication in Multi-Agent Systems*, pages 98–117, 2004.

Terry R. Payne, Terri L. Lenox, Susan Hahn, Michael Lewis, and Katia Sycara. Agent-Based Team Aiding in a Time Critical Task. In *In Proceedings of Hawaii International Conference on System Sciences*, 2000.

Jeremy Pitt and E. H. Mamdani. Communication Protocols in Multi-agent Systems: A Development Method and Reference Architecture. In *Issues in Agent Communication*, pages 160–177, 2000.

Desanka Polajnar, Jernej Polajnar, and L. Lukic. Metamodel Abstractions of Agent Roles in Cooperative Process Planning. In *Proceedings of the 2008 IEEE SMC International Conference on Distributed Human-Machine Systems*, pages 77–82, Athens, Greece, 2008.

Jernej Polajnar, Behrooz Dalvandi, and Desanka Polajnar. Does Empathy between Artificial Agents Improve Agent Teamwork? In *Proceedings of the10th IEEE International Conference on Cognitive Informatics and Cognitive Computing*, Banff, Alberta, Canada, 2011. IEEE Computer Society.

Anand Rao and Michael Georgeff. BDI-Agents: From Theory to Practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, 1995.

Tuomas W. Sandholm. An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations. *Eleventh National Conference on Artificial Intelligence*, pages 256–262, 1993.

Tuomas W. Sandholm. *Distributed Rational Decision Making*, pages 201–258. MIT Press, Cambridge, MA, USA, 1999.

John Rogers Searle. Collective Intentions and Actions. In *Intentions in Communication*, pages 401–416. The MIT Press, Cambridge, MA, 1990.

Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-theoretic, and Logical Foundations.* Cambridge University Press, 2009.

Maarten Sierhuis, Jeffrey M. Bradshaw, Alessandro Acquisti, Ron van Hoof, Renia Jeffers, and Andrzej Uszok. Human-Agent Teamwork and Adjustable Autonomy in Practice. In *Proceedings of the Seventh International Symptosium on Artificial Intelligence, Robotics and Automation in Space*, 2003.

Munindar Singh. Group Ability and Structure. *Decentralized Artificial Intelligence*, 2, 1991a.

Munindar P. Singh. A Logic of Situated Know-How. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 343–348. AAAI Press, 1991b.

Munindar P. Singh. Know-How. In *Foundations of Rational Agency*, Applied Logic Series, pages 105–132. Kluwer, 1999.

Ira Smith and Philip R. Cohen. Toward a Semantics for an Agent Communication Language Based on Speech-acts. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 24–31, 1995.

Reid G. Smith. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, 29:1104–1113, 1980.

Robert Fletcher Sproull and Dan Cohen. High-Level Protocols. *Proceedings of the IEEE*, 66(11):1371– 1386, 1978.

Katia. Sycara and Michael. Lewis. Integrating Intelligent Agents into Human Teams. In *Team Cognition: Understanding the Factors that Drive Process and Performance.* American Psychological Association, 2004.

Milind Tambe. Agent Architectures for Flexible, Practical Teamwork. In *Proceedings of the National Conference on Artificial Intelligence*, 1997.

Wiebe van ver Hoek and Michael Wooldridge. Cooperation, Knowledge, and Time: Alternating-time Temporal Epistemic Logic and its Applications. *The Dynamics of Knowledge*, 75:125–157, 2003.

Gerhard Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* The MIT Press, 2001.

Michael Wooldridge. *An Introduction to MultiAgent Systems.* Wiley, second edition, 2009.

Michael Wooldridge and Nicholas Jennings. Formalizing the Cooperative Problem Solving Process. In *Proceedings of the Thirteenth International Workshop on Distributed Artificial Intelligence*, pages 403–417, 1994.

Michael Wooldridge and Nicholas R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

Michael Wooldridge and Nicholas R. Jennings. The Cooperative Problem Solving Process. *Logic and Computation*, 9(4):563–592, 1999.

John Yen, Jianwen Yin, Thomas R. Ioerger, Michael S. Miller, Dianxiang Xu, and Richard A. Volz. CAST: Collaborative Agents for Simulating Teamwork. In *17th International Joint Conference on Artificial Intelligence*, pages 1135–1144, 2001.

John Yen, Xiaocong Fan, and Richard A. Volz. Information Needs in Agent Teamwork. *Web Intelligence and Agent Systems*, 2:231–247, 2004.