AN INTERACTION PROTOCOL FOR BIDIRECTIONAL DELIBERATION ON DIRECT HELP IN AGENT TEAMWORK

by

MOJTABA MALEK AKHLAGH

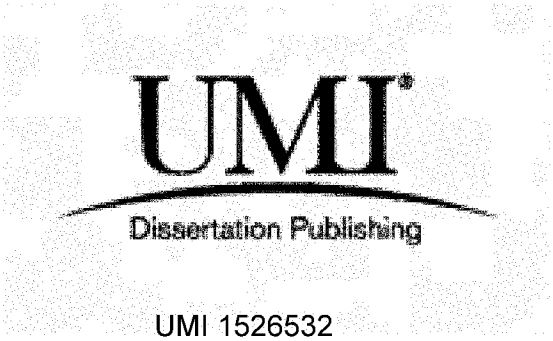
B.Sc., Physics,
University of Guilan, Iran, 2007

Master of Information Technology,
Multimedia University, Malaysia, 2010

THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER SCIENCE

UNIVERSITY OF NORTHERN BRITISH COLUMBIA

April 2015



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

This thesis proposes a new interaction protocol for direct help in agent teamwork. It addresses design questions that may arise in practical systems development, and achieves higher teamwork performance impact than previous versions of the Mutual Assistance Protocol (MAP). Direct help, such as performing an action on teammate's behalf, is deliberated by team members as need arises, rather than imposed by team organization or centralized mechanisms. The deliberation can start with a request for help, or with an offer of help; the two design principles have been embodied in two distinct versions of MAP. Based on their observed complementarity, we refine and combine them into a single protocol that leverages their individual advantages. Its novel features let an agent initiate help deliberation with request or offer, and also simultaneously provide and receive help. Simulation experiments demonstrate its team performance gains while varying the environment dynamism, agent resources, and communication costs.

Contents

List of Figures

Previously Published Work

Portions of this thesis appear in the following paper:

Mojtaba Malek Akhlagh and Jernej Polajnar. Distributed deliberation on direct help in agent teamwork. In Proceedings of the 12th European Conference on Multi-Agent Systems (EUMAS 2014), Prague, Czech Republic, December 2014. To appear as LNAI 8953, Springer, 2015.

Acknowledgements

I would like to sincerely express my gratitude to my supervisor, Dr. Jernej Polajnar, for his invaluable support throughout all stages of this thesis. Such a study would not happen today without his experience, advice, encouragement, and patience.

I am grateful to my committee member, Dr. Desanka Polajnar, for her support, advice, and encouragement.

I am thankful to Dr. Iliya Bluskov for his support and interest in serving on my supervisory committee.

I would like to thank my colleague, Denish Mumbaiwala, for his help and input on the simulation experiments. I am also thankful to Narek Nalbandyan and Omid Alemi for sharing their experience and insight on the subject.

This Thesis is dedicated to my family

Nahid Mohseni, Esmaeil Malek Akhlagh,
and Moein Malek Akhlagh

for their endless love, support, and encouragement.

Chapter 1

Introduction

Teamwork in multiagent systems is a rapidly evolving field of study with rising research motivations in the last two decades. Agents are autonomous intelligent entities, situated in an environment, and performing tasks that are delegated to them. They can react to changes in the environment, formulate and proactively pursue their own goals, and interact socially with other agents and humans [Wooldridge, 2009]. In particular, intelligent agents can exhibit collaborative behavior in a team in order to achieve a joint objective. The groundwork for studies of agent teamwork has been provided in the fundamental papers by Levesque et al. [1990], Cohen and Levesque [1991], Wooldridge and Jennings [1994], Rao and Georgeff [1995], Grosz and Kraus [1996], and Dunin-Keplicz and Verbrugge [2010]. Early work also included development of multiagent software platforms [Georgeff and Lansky, 1987] and applications of agent teamwork [Ljungberg and Lucas, 1992]. More recently, technological advances in intelligent distributed systems increasingly motivate research on agent teamwork from an engineering perspective.

The collaboration in an agent team can incorporate helpful behavior among the teammates, inspired by effective human teamwork context. The potential benefits of helpful behavior to human team performance are recognized in management practice [LePine et al., 2000]. Helpful behavior in agent teamwork refers to a collaborative act performed by one agent in order to assist its teammate. Different mechanisms for assistance between teammates by executing actions or providing information have been proposed in the last two decades [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., 2012].

Helpful behavior among agents can be enforced by their global team organization or a centralized mechanism, or it can be specified as direct help, which is initiated by individual agents as need arises and bilaterally agreed upon by the agents involved in the help act. Direct help is specified as a possible component of team strategies for decentralized and reactive adjustment of team behavior to an unpredictable changing environment [Polajnar et al., 2014]. The growing practical importance of multiagent systems motivates the investigation of potential benefits in team performance from incorporation of direct help mechanisms.

In order to investigate the practical impact of direct help upon team performance from an engineering perspective, one needs to develop interaction mechanisms. Agent interaction protocols specify interactive behavior of individual agents dealing with each other. Nalbandyan [2011] and Polajnar et al. [2012] introduce the Mutual Assistance Protocol (MAP), with its subsequent elaboration in [Nalbandyan et al., 2013], which specifies a direct help mechanism for an agent team with heterogeneous skill profiles, situated in a dynamically changing environment. The mechanism is based on a bilateral distributed agreement in which both requester and helper agents jointly decide on performing a help act, in contrast with unilateral approaches as in [Ka-

mar et al., 2009], in which only one side of the help transaction makes the decision. MAP has two different generic types: Action MAP, that enables teammates to help each other by directly performing actions, and Resource MAP, that enables teammates to help each other by providing the needed resources. Simulation experiments in [Polajnar et al., 2012] suggest behavioral advantages of employing Action MAP over unilateral help protocols when communication cost is relatively low compared to agents' action costs.

Deliberation on possible direct help can be initiated in two different ways. In the requester-initiated approach, the opening message of the help transaction is a request for help; in the helper-initiated approach, the opening message is an offer of help. Simulation experiments in [Nalbandyan et al., 2013] show that the Requester and Helper-Initiated Action MAP (RIAMAP and HIAMAP) have complementary impacts on team performance with respect to varying levels of environment dynamism, agent resources, and communication cost. Where one dominates, the other is often weak, but jointly they maintain superiority over all other Action MAP versions across the parameter space. This observation has raised the research question of whether a possible combination of the two different approaches might exhibit a superior performance.

In this thesis, we approach this research question by designing new protocols, developing simulation models for them, exploring their impact on team performance through simulation experiments, and adjusting our model parameters for optimum performance. We study agent interaction protocols for direct help in the context of a board game microworld, inspired by the Colored Trails game [Gal et al., 2010], and used in previous studies of MAP [Polajnar et al., 2012, Nalbandyan et al., 2013]. For modeling and simulation, we use the Agent Interaction Modeling Simulator (AIMS)

framework introduced by Alemi et al. [2014], which allows concurrent simulation of multiple teams in identical dynamic environments. It facilitates representation of agent interaction protocols and modeling of environment dynamism, and provides an interactive simulation mode to gain direct insights into behavior comparisons between teams that employ different help protocols, but are otherwise identical and operate in identical dynamic environments.

In particular, we investigate an interaction protocol design that combines the requester and helper-initiated approaches for deliberation on direct help. We start with the observation that in the existing requester and helper-initiated protocols, RIAMAP and HIAMAP, an agent is only allowed to either provide or receive help at any given time. We refine the existing protocols to enable agents to provide and receive help simultaneously. The refined protocols, RIAMAP* and HIAMAP*, are then composed into a single new protocol that allows help deliberation to be initiated by requesters as well as potential helpers. The new protocol, called the Bidirectionally Initiated Action MAP (BIAMAP), allows an agent to initiate help deliberation by either a request or an offer, and to provide and receive help simultaneously.

The design objective of BIAMAP is to specify an interactive behavior model for direct help among agents that incorporates both proactive requesting and proactive offering behaviors and optimizes the balance between them in order to leverage the advantages of each across a space of parameters representing environment dynamism, agent resources, and communication cost.

We represent the new interaction protocols using a particular state-machine formalism, translate them to executable code in the AIMS framework, and investigate them through simulation experiments across the parameter space. We first evaluate the team performance impact of enabling individual agents to simultaneously provide

and receive help, and then the team performance impact of bidirectional deliberation on direct help. The simulation results show that RIAMAP* and HIAMAP* outperform RIAMAP and HIAMAP, respectively, in most situations. They also show the superiority of BIAMAP over both RIAMAP* and HIAMAP* in most of the parameter space.

In summary, Nalbandyan [2011] and Polajnar et al. [2012] introduce MAP as a better performing interaction protocol in comparison to the unilateral protocols in situations when the communication costs are relatively low compared to the actions costs. RIAMAP and HIAMAP are introduced in [Nalbandyan et al., 2013] as two advanced variations of MAP, which outperform the basic MAP version and demonstrate complementary behaviors over the parameter space. In this thesis, first we introduce RIAMAP* and HIAMAP*, their refined versions demonstrating performance gains, and finally introduce BIAMAP, a composition of RIAMAP* and HIAMAP*, being the best-performing variation of the MAP family using the same parameter settings.

The rest of this thesis is organized as follows. Chapter 2 covers the background and related work, Chapter 3 describes the research objectives and presents refinements of Action MAP, Chapter 4 presents the Bidirectionally Initiated Action MAP (BIAMAP), Chapter 5 presents an evaluation of the new protocols in simulation experiments, and Chapter 6 presents the conclusions and future work.

Chapter 2

Background and Related Work

In this chapter, we overview some groundwork together with related recent studies in multiagent systems, agent teamwork, helpful behavior in agent teams, agent intraction protocols, and the Mutual Assistance Protocol (MAP) family, with the focus relevant to this thesis.

2.1 Multiagent Systems

According to [Wooldridge, 2009], an agent is a computer system situated in some environment, holding some level of autonomy in order to satisfy its design objectives; and a multi-agent system is a system composed of multiple interacting agents within an environment. 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 and Jennings [1995] suggest that in order to consider an agent to be intelligent, the agent is expected to hold the following capabilities. An agent needs to be *reactive*, i.e., able to perceive the environment and respond to occurring changes in a timely manner; *proactive*, i.e., able to take initiatives and perform goal-directed behavior; and *social*, i.e., able to interact with agents and possibly humans.

The environments that agents might be situated in can have some general properties classified by Russell and Norvig [1995]. For instance, an environment can be deterministic or non-deterministic. In a deterministic environment, each agent's action has a single guaranteed effect, and there is no uncertainty regarding the resulting environment state due to the action. Also, an environment can be static or dynamic. In a static case, the environment state only changes by the agent's actions; while in a dynamic case, there are other processes influencing the environment beyond the agent's control. In a basic model of agents interacting with their environments, the environment is initially in some state, the agent perceives the current state and performs an action in order to influence the environment and achieve a desirable state; the environment may respond to the action by transiting to another state which in advance may be unknown to the agent, due to the environment being non-deterministic or dynamic. Hence, the agent may fail and try to perform another action in order to achieve its goal; and so on. The states of the environment can be associated with real values represented by a utility function, specifying how good each state is, letting the agent to deliberate on what to do.

Practical reasoning [Bratman, 1987] is a particular model of decision-making in contrast with the traditional approach of *theoretical reasoning*, also known as *symbolic artificial intelligence*. While theoretical reasoning is directed towards beliefs, practical reasoning is directed toward actions. In theoretical reasoning, logical theorems are

used as models of decision making and agents act as theorem provers. In practical reasoning, the decision making process consists of two distinct activities. The first one, called *deliberation*, answers the question of *what* state of affairs to achieve, and the second one, called *means-ends reasoning*, discovers *how* to achieve these states of affairs. The output of deliberation is *intention* and the output of means-ends reasoning is *plan*. The practical reasoning agents are endowed with mental states similar to the ones in human mind, which are used in their decision making processes.

The most popular practical reasoning architecture has been the *Belief-Desire-Intention (BDI)* formalism. In this framework, the agent system is viewed as a rational agent with mental attitudes of belief, desire, and intention, representing the informative state, the motivational state, and the deliberative state of the rational agent, respectively. The behavior of the system is then determined by these mental attitudes and they are critical for desired performance when deliberation is subject to resource bounds [Bratman et al., 1988]. Modal logic, by which statements such as "necessarily true" or "possibly true" can be expressed [Hughes and Cresswell, 1996, Blackburn et al., 2001], has been applied for modelling and axiomatization of BDI agents [Rao and Georgeff, 1991, 1993, Georgeff and Rao, 1995].

The Procedural Reasoning System (PRS), developed by Georgeff and Lansky [1987], was one of the first and best known agent architectures to explicitly embody the BDI paradigm. It has been applied in several practical multiagent applications that are successfully used in the real world, such as the OASIS air traffic management system [Ljungberg and Lucas, 1992], and the SPOC business process management system [Georgeff and Rao, 1996].

Rao and Georgeff [1995] address two main criticisms against formalization of BDI agents: the necessity or adequacy of the three mental attitudes, and the utility of

applying modal logic that does not have complete axiomatizations and cannot be efficiently computed. They argue the necessity of all three mental attitudes in application domains where real-time performance is required. They also show that, with certain simplifying assumptions, it is possible to build practical BDI systems.

2.2 Agent Teamwork

Agent teamwork refers to the collaboration of individual agents who are committed to accomplishing of a particular task. The research on agent teamwork is an active field of study, and there have been different approaches to formalization of the mental states and semantics needed for agents' collaborative behavior.

The agent teamwork research started with the BDI framework [Cohen and Levesque, 1990, Levesque et al., 1990]. Cohen and Levesque [1991] formalize the notions of joint action, joint commitment, and joint intention. They argue that a joint action involves more than just the union of simultaneous individual actions even when the actions are coordinated, and it is done by individuals sharing certain specific mental properties. In their work, they investigate joint commitment and joint intention specifications based on individual cases such that a team of agents acts as an aggregate agent. The individual agents are situated in a dynamic environment in which there is potential divergence of mental state. This potential divergence makes the design specifications of teamwork complicated since there is tension in acting as an aggregate agent. For an agent team with a common goal, in order to hold the team together while letting agents have their own private beliefs, each agent treats the common goal as a weak achievement goal, i.e., the agent considers the possibility that any agent in the team may have privately come to believe that the goal is either achieved, unachievable, or

irrelevant, and is committed to make that belief mutually known to all other agents. Therefore, if any agent comes to believe that the goal is accomplished, not achievable, or irrelevant, it drops the common goal, but has a goal that the new status be mutually known by all the team members.

The Distributed Constraint Optimization Problem (DCOP) [Mailler and Lesser, 2004, Modi et al., 2005] is another framework for modeling teamwork in multiagent systems. In this framework, the agents cooperate as a team and share a common reward function. This framework is useful for modelling distributed reasoning in multi-agent domains for being able to perform optimization over a set of distributed constraints and having agents' information private. Formally, for each agent in the team, there is a variable from a set $V = \{x_1, x_2, ..., x_n\}$, over whose value the agent has control. Variable x_i can take on any value from a discrete finite domain D_i. The goal of optimization is to change the variable values so that the sum over a set of binary constraints and associated payoff or reward functions is maximized. DCOPs have been applied to meeting scheduling problems, allocating tasks, and coordinating teams of agents [Taylor et al., 2011].

The Partially Observable Markov Decision Process (POMDP) [Kaelbling et al., 1998] provides a mathematical framework to model sequential decision-making with uncertainty. POMDPs consider uncertainty in both agent's observations and actions. An agent in this model keeps a probability distribution over a set of possible states. They have been applied to real world domains including robot navigation and machine maintenance [Taylor et al., 2011]. Decentralized POMDPs (DEC-POMDPs) are frameworks to model this behavior in multi-agents systems. In general, finding an optimal joint policy for general DEC-POMDPs is NEXTP-complete [Bernstein et al., 2002]. There are two general approaches to this problem. The first one tries to find

approximate solutions using efficient algorithms [Nair et al., 2003, Bernstein et al., 2005], which gives no guarantee on the solution quality. The second category tries to find the global optimal solution by applying useful DEC-POMDPs' subclasses [Becker et al., 2004, Nair et al., 2005], which lacks expressiveness [Taylor et al., 2011].

While the BDI framework is focused on execution-time reasoning, the DCOP and POMDP frameworks focus more on planning-time reasoning and they often fail to scale up to large numbers of agents because of high computational cost for a high-quality joint policy and also to deal with the model uncertainty. A recent work by Taylor et al. [2011] argues that execution-time reasoning is a critical component to multi-agent systems and shows that robustly combining execution-time and planning-time reasoning in DCOP and POMDP frameworks results in better performance in terms of achieved award, runtime and scalability. It presents new algorithms that integrate execution-time and planning-time reasoning and suggests that multi-agent community should focus on incorporating more execution-time reasoning.

2.3 Helpful Behavior in Agent Teams

In the last two decades, there has been a research interest in helpful behavior in agent teamwork, which specifies mechanisms for assistance between teammates by executing actions or providing information [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., 2012].

Kamar et al. [2009] provide a decision-theoretic mechanism for incorporating helpful behavior in agent teams working on a collaborative activity. The authors argue that helpful actions incur some costs to the interacting agents, which may be due

to communication costs, missed opportunities while helping, adaptations needed for receiving the provided help, or interruption costs. They formally integrate decision-theoretic and BDI models by incorporating costs and uncertainty into deliberation, reasoning based on local probabilistic beliefs of individual agents, and formalizing helpful behavior based on the commitments and intentions of the participating agents. They formulate two distinct types of helpful behavior including performing actions and providing information, which are based on a unilateral decision making mechanism; i.e. the decision on performing helpful behavior is made by only one participant in the collaborative activity. In another work, Kamar and Grosz [2007] investigate effective interruption management in collaborative human-computer activities. They formalize interruptions as multi-agent decision making in a human-computer setting.

Polajnar et al. [2011] argue that when an agent team is situated in a realistic dynamic environment which is subject to unpredictable changes, helpful behavior among teammates can have a positive impact on the team performance. The authors point out that for an agent acting in such an environment, there might be situations caused by unexpected events which require specialized or collective behavior. As it might be impractical to model agents with all potentially required capabilities at design time, it is beneficial to incorporate mechanisms for helpful behavior at execution time. On the other hand, there is an advantage in modeling agents with certain standard capabilities for their reusability in similar domains. In that case, an agent may have abilities beyond its immediate role in a particular application, which can potentially be incorporated in a help mechanism in order to advance the team performance.

Nalbandyan [2011] and Polajnar et al. [2012] present an interaction protocol for helpful behavior, called the Mutual Assistance Protocol (MAP), with its subsequent

elaboration in [Nalbandyan et al., 2013]. MAP is based on a distributed decision making mechanism, called bilateral distributed agreement (BDA), in which the decision on performing a help act is made jointly by both requester and helper agents, in contrast with unilateral approaches presented in [Kamar et al., 2009].

In a different approach, Dalvandi [2012] investigates the need for empathy among artificial agents and provides simulation results suggesting the behavioral advantages of incorporating the empathy-driven deliberation on helpful behavior in certain dynamic environments where rational deliberation is costly. This approach is inspired by the way living systems deliberate on helpful behavior in certain situations based on the notion of empathy.

In another work, Polajnar et al. [2014] formulate adaptation strategies for agent teamwork organization situated in a dynamic environment with unpredictable changes. The authors formulate a decentralized reactive approach based on flexible, dynamic combinations of complementary adjustment strategies. The specific strategies include: individual replanning, direct help among teammates, and subtask swapping among the teammates. The simulation results in a microworld setting suggest that helpful behavior is generally the most beneficial contributor to the combined strategies; however, the most inclusive combinations tend to dominate.

2.4 Agent Interaction Protocols

Agent protocols are generally of two types: communication protocols and interaction protocols. Agent communication protocols operate at a low level regulating the means of sending and receiving information, typically by message passing, among the indi-

vidual agents. They provide speech act classification and semantics, and the domain ontology the agents use to understand each other [Wooldridge, 2009]. KQML/KIF [Mayfield et al., 1996] and FIPA-ACL [Foundation of Intelligent Physical Agents, 1997] are two best known standard agent communication protocols. Agent interaction protocols are high-level protocols and provide sets of rules specifying interactive behavior of individual agents. They model agents' social activities such as coordination, negotiation, and collaboration. One of the most widely used agent interaction protocols has been the Contract Net Protocol (CNP), originally designed by Smith [1980] for cooperative problem solving between nodes in distributed systems, inspired by the way companies put contracts out to tender [Wooldridge, 2009]. Paurobally et al. [2004] discuss CNP interactions for self-interested agents, based on competitive bidding between contracting agents.

One may consider designing a universal agent interaction protocol which can be employed in all the possible circumstances. Dunn-Davies et al. [2005] point out that such a global protocol is not realistic as agents' social behavior may be realized in a variety of approaches. Hence, each specific agent interactive behavior in a particular domain requires designing a specific interaction protocol.

Wajid and Mehandjiev [2013] classify agent interaction approaches into two general groups: protocol-based approaches, which specify agents' behavior using interaction protocols, and approaches without protocols, which constrain agents' behavior without using protocols. They point out that in the protocol-based approaches, some level of predictive behavior is ensured in agent interactions, however less flexibility is enforced in agents' behavior. On the other hand, approaches without protocols increase the flexibility of agents' behavior, but decrease the level of predictive outcome. The authors further classify protocol-based approaches by considering whether

the protocols are created at design time or run time. Obviously, approaches which create their protocols at run-time are more flexible, but less predictable; and those which use design-time created protocols have less flexibility, but more predictability. Finally, approaches which create their protocols at design-time are divided into two subclasses based on the time of loading protocols: approaches which load the protocols at run-time and those which load at design-time; again, the latter being less flexible, but more predictable.

2.5 The Mutual Assistance Protocol (MAP)

The Mutual Assistance Protocol (MAP) [Nalbandyan, 2011, Polajnar et al., 2012] has been specifically designed for helpful behavior in agent teamwork, distinguished by its distributed deliberation approach. MAP provides a mechanism for helpful behavior by enabling agents in a team to directly assist each other. Its bidding sequence is similar to the one in the Contract Net Protocol [Smith, 1980]. The help deliberation process is jointly determined by two agents through a bilateral distributed agreement. In this section, we first describe its team model, principles, and variations, and then review some design features, introduced in [Nalbandyan et al., 2013], which formulate criteria for deliberation on whether to engage in helpful behavior.

2.5.1 MAP Characterization and Principles

1. The Agent Team Model

A team A consists of agents A_i, $i \in I = \{1, \ldots, n\}$, $n > 1$, situated in an environment E. The agents are able to perform actions from a domain $Act = \{\alpha_1, \ldots, \alpha_m\}$, $m > 1$. Each agent A_i has its individual skill profile, which defines A_i's efficiency for the domain Act in the context of environment state. A_i knows its own skill profile precisely; but its knowledge of teammates' profiles may be limited to a varying degree. The team is assigned a task T, with each agent A_i addressing a subtask T_i that has a resource budget R_i. The subtask assignment process can be random or optimal [Polajnar et al., 2014]. The environment E can change dynamically by events other than the actions of any agent in the team. Agent A_i acts rationally with the objective to advance the team performance.

2. Local Planning Autonomy (LPA)

Agents in a team are equipped with a property called *local planning autonomy* (LPA), which enables each agent to generate its own local plan, π_i, for its assigned subtask, T_i. An agent A_i rationally selects the best plan among its candidate plans using its own beliefs, B_i, based on its individual view of team interest rather than self-interest, represented by A_i's *team utility function*

$$u_i : Plans \times BeliefSets \longrightarrow \mathbb{R}^+$$

where $\mathbb{R}^+$ is the set of positive real numbers. Thus, each individual agent can autonomously assess how to best contribute to the team. Agents with LPA can engage in helpful behavior when they jointly determine through a bilateral distributed agreement, that the help act is in the interest of the team.

3. Bilateral Distributed Agreement (BDA)

The fundamental design concept underlying the MAP protocol is that the deliberation on whether to perform a helpful behavior is based on a bilateral distributed approach in contrast to unilateral approaches as in [Kamar et al., 2009]. In a *bilateral distributed agreement* (BDA), the two parties deliberating about a help transaction jointly decide on performing the helpful behavior based on their individual assessments of team benefit and team loss. The agent A_i that considers receiving help, calculates the *team benefit*, $\Delta_i^+ = u_i(\pi_i', B_i) - u_i(\pi_i, B_i)$, where π_i is its original plan and π_i' its new plan affected by the possible received help. Similarly, the agent A_j that considers providing help calculates the *team loss*, $\Delta_j^- = u_j(\pi_j, B_j) - u_j(\pi_j'', B_j)$, where π_j is its original plan and π_j'' its new plan that includes the help act. The help transaction may occur only if the difference $\Delta_{ij} = \Delta_i^+ - \Delta_j^-$, called the *net team impact (NTI)*, is positive. Note that agent A_i calculates Δ_i^+ using its individual belief set B_i, while agent A_j calculates Δ_j^- using its own beliefs B_j. (In particular, note that agent's individual beliefs include its cost vector.) The team benefit and team loss calculations are done from the receiver and provider's perspectives, respectively, and therefore the functions u_i and u_j must be properly mutually scaled to allow meaningful comparisons. The agreement is reached through a bidding sequence similar to the one in the Contract Net Protocol [Smith, 1980]. The behavioral advantages of the bilateral distributed approach are presented in [Polajnar et al., 2012] by modeling and simulation of the

MAP protocol and two unilateral protocols.

4. Variations of the MAP

There are two generic versions of the MAP protocol: Action MAP and Resource MAP. In Action MAP, an agent can perform an action on behalf of a teammate. In Resource MAP, an agent can provide a part of resources needed for an action to be performed by a teammate. Action help is performed by a single helper agent, while multiple helper teammates can cooperate in resource MAP.

Furthermore, helpful behavior can be initiated either by a requester agent who asks for help or a helper agent who offers help. The corresponding interaction protocols, called the Requester-Initiated Action MAP (RIAMAP) and the Helper-Initiated Action MAP (HIAMAP), are introduced in [Nalbandyan et al., 2013].

2.5.2 Metrics for Help Deliberation

1. Estimating the Cost of a Plan

Each agent A_i is able to generate candidate plans to accomplish its assigned subtask T_i. The agent selects the least-cost plan among the generated plans and remains committed to it. However, the environment can change dynamically and the initial plan cost can change as the agent proceeds. If the agent knows the model of environment dynamism, it can estimate the expected cost of its initial plan, or its remainder [Nalbandyan et al., 2013].

2. Agent's Individual Wellbeing

The *individual wellbeing* metric, introduced in [Nalbandyan et al., 2013], expresses how well an agent A_i is accomplishing its subtask T_i. It is defined as:

$$W_i = \frac{R_i - Ecost_i(P_i)}{(\ell + 1)\bar{c}_i} \tag{2.1}$$

where $Ecost_i(P_i)$ is the estimated cost of the remaining plan P_i for agent A_i, ℓ is the number of actions in P_i, R_i is the remaining resources, and $\bar{c}_i$ is the average expected cost of an action for A_i (i.e., $\bar{c}_i = (1/m)\Sigma_{k=1}^{m} c_{ik}$). The wellbeing value changes as A_i performs actions, gets involved in a help act, or as the environment state changes. An agent with positive wellbeing expects to accomplish its subtask with its own resource budget and have some excess of resources; while a negative wellbeing indicates shortage of resources and that help may be needed. Agents apply wellbeing thresholds called *watermarks* in order to deliberate on helpful behavior.

3. Proximity to Accomplishing a Subtask

Agents incorporate the notion of proximity to accomplishment into their deliberation on help act in order to favor the teammates who are in proximity of accomplishing their subtasks and are more likely to score for the team [Nalbandyan et al., 2013]. An agent A_i applies a *proximitiy bias* function, *pbias*, which maps its remaining plan length, ℓ, into a positive priority value, to its assessments of the team benefit and team loss.

2.6 The Requester-Initiated Action MAP

In the Requester-Initiated Action MAP (RIAMAP), agents can proactively request action help, but they do not offer help [Nalbandyan et al., 2013]. Agents can bid to the requests, but if an agent has sent a request, it does not bid to other requests. The protocol comprises three interaction phases as follows. The corresponding sequence diagram is given in Fig. 2.1.

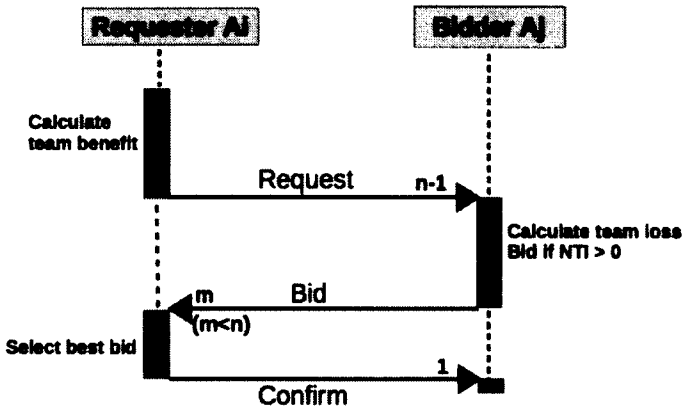


Figure 2.1: The RIAMAP bidding sequence.

1) Help request generation:

At the start of every round, agent A_i with next action α_k at the cost of $cost_{ik}$ and the current calculated wellbeing of W_i broadcasts a help request message containing α_k and the calculated team benefit, Δ_i^+, if any of the following three conditions holds:

(i) $R_i < cost_{ik}$;

(ii) $W_i < W^{LL}$ and $cost_{ik} > LowCostThreshold$;

(iii) $cost_{ik} > RequestThreshold$;

where *LowCostThreshold* is the upper limit of the "cheap" action range, *Request-Threshold* is the lower limit of the "expensive" action range, and W^{LL} is a fixed *low watermark* value for wellbeing. Condition (i) means that the agent's remaining resources R_i are insufficient for A_i to perform α_k; only help can unblock the agent. Condition (ii) indicates that the agent's wellbeing is low, and that α_k is not cheap in its skill profile, which will further reduce the wellbeing; help will improve the agent's chances of reaching the goal. Condition (iii) indicates that α_k is expensive for agent A_i; help may benefit the team regardless of A_i's wellbeing.

2) Bidding to a request:

Each agent receives the request and deliberates on bidding to it, while an agent who has sent a request does not. Agent A_j calculates the team loss, Δ_j^-, for performing the requested action α_k and NTI using the received team benefit, Δ_i^+. If NTI is positive, A_j sends a bid containing α_k and the associated NTI value to the requester A_i. In case of having multiple qualified requests, A_j bids to the one with highest NTI.

3) Confirming the chosen bid:

Agent A_i receives the bids, selects the one with highest NTI, and sends a confirmation to the bidder agent A_j.

An Improved Version of RIAMAP

Polajnar et al. [2014] introduce a "frugal" version of RIAMAP, called RIAMAP2, in which the help transaction may also occur when NTI is equal to zero. The original criterion, NTI > 0, allows a help act only if it is expected to increase the team score. However, the calculation of the expected team score does not take into account the potential impact of help. RIAMAP2 recognizes the fact that if help saves the total resources at team level, the savings can later be used for help acts that may eventually increase the team score. When agent A_i requests help from agent A_j for an action α_k, and NTI $= 0$, savings occur if A_j performs α_k at lower cost than A_i. The cost comparison accounts for the fact that agent A_j, when performing α_k on behalf of a teammate (rather than within its own local plan), incurs additional overhead cost, represented here by a constant *help overhead*, h. In RIAMAP2, the help act is allowed to occur if either NTI > 0, or else NTI $= 0$ and $cost_{ik} - cost_{jk} > h$.

2.7 The Helper-Initiated Action MAP

In the Helper-Initiated Action MAP (HIAMAP), agents can proactively offer action help, but they do not otherwise request help [Nalbandyan et al., 2013]. Agents can bid to the offers, but if an agent has sent an offer, it does not bid to other offers. The protocol comprises three interaction phases as follows. The corresponding sequence diagram is given in Fig. 2.2.

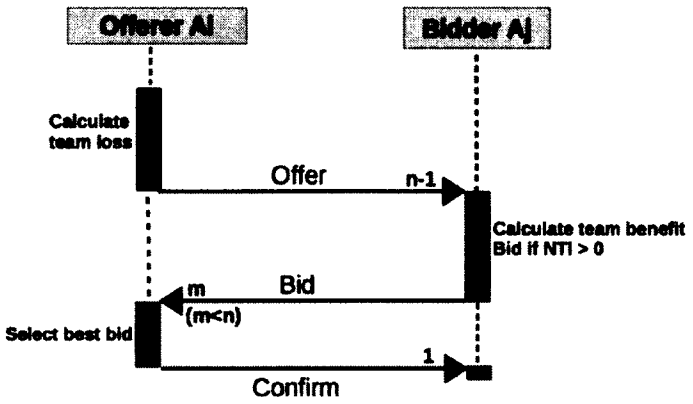


Figure 2.2: The HIAMAP bidding sequence.

1) Help offer generation:

At the start of every round, agent A_i calculates its individual wellbeing W_i. If W_i is above a fixed *high watermark* W^{HH}, it broadcasts an offer message containing pairs $[\alpha_k, \Delta_i^{(k)-}]$ for each action α_k for which A_i's cost is below the *OfferThreshold* parameter, where $\Delta_i^{(k)-}$ denotes the team loss associated with α_k.

2) Bidding to an offer:

Each agent receives the offer and deliberates on bidding to it, except an agent who itself has sent an offer. If an agent A_j finds that its next action α_k is included in the offer, it calculates the team benefit for not performing the offered action, $\Delta_j^{(k)+}$, and uses the team loss, $\Delta_i^{(k)-}$, received in the offer, to calculate the value of NTI. If NTI is positive, A_j sends a bid containing α_k and the associated NTI value to A_i. In case of having multiple qualified offers, A_j bids to the one with highest NTI.

3) Confirming the chosen bid:

Agent A_i receives the bids, selects the one with highest NTI, and sends a confirmation to the bidder agent A_j.

2.8 The Agent Interaction Modeling Simulator

The Agent Interaction Modeling Simulator (AIMS) framework introduced in [Alemi et al., 2014] is developed for design-oriented simulation studies of interaction models used in agent teamwork. It allows concurrent simulation of multiple teams in identical dynamic environments. It facilitates representation of agent interaction protocols and modeling of environment dynamism.

The AIMS framework has been specifically developed to support incremental improvement of interaction protocol design by performance optimization of the protocol through iterative adjustment of its parameter values in a series of simulation experiments. Its interactive simulation mode provides direct insights into behavior comparisons between teams that employ different help protocols, which can be used in each iteration of the protocol design.

In the AIMS framework, protocols are represented as state machines of a particular type, with synchronous communication and alternating states for sending and receiving messages. Such representations are easily translatable to executable code in AIMS.

Chapter 3

Refinements of Action MAP

This chapter presents the motivation for designing a new combined protocol together with its design objectives. It includes the refined versions of the requester and helper-initiated protocols which are required for composition of the new combined protocol. Section 3.1 describes the research motivation, Section 3.2 formalizes a refined team model which explicitly represents the impact of the environment state, Sections 3.3 and 3.4 present refined models of the two one-sided protocols which allow simultaneous providing and receiving help, and Section 3.5 provides the state-machine representations of the refined protocols.

3.1 The Motivation and Research Objectives

The motivation for incorporation of helpful behavior in agent teamwork emanates from its potential benefits in practical applications. In this study, we are interested

in investigating the practical impact of direct help upon team performance from an engineering perspective. Our research objectives involve developing a comprehensive mechanism for the deliberation process in help interactions in order to advance the team performance. The Mutual Assistance Protocol (MAP) family provides a direct help mechanism by enabling agents in a team to directly assist each other. Distributed deliberation on direct help is the underlying principle of MAP. In a decentralized manner, the members of an agent team use their own individual beliefs to jointly deliberate on performing a help act. Each direct help act results from a bilateral distributed agreement that is in the interest of the team.

The deliberation on getting involved in helpful behavior, whether to provide or receive direct help, can be initiated from two different sides. One possibility is for an agent to make a request for help to its potential helper teammates. Alternatively, it is possible for an agent to make an offer of help to its potential requester teammates. The former approach is realized in the Requester-Initiated Action MAP (RIAMAP), and the latter in the Helper-Initiated Action MAP (HIAMAP). The practical impact of the two different approaches on team performance has been investigated through simulation studies. It was noted in [Nalbandyan et al., 2013] that the performance profiles of RIAMAP and HIAMAP are complementary. In certain situations, where one is performing weakly, the other is often dominating. While neither of them generally outperforms the other, together they maintain superiority across the parameter space, representing environment dynamism, agent resources, and communication cost. This has given rise to the research question of whether a possible combination of the two one-sided protocols might show an even better performance profile. This combination can be realized by letting agents to initiate help deliberation by both requests and offers in a single combined protocol.

In this study, we compose the requester and helper-initiated protocols in order to specify a single interaction protocol that integrates both proactive requesting and proactive offering of direct help. The new protocol specification is intended to leverage the advantages of the two one-sided protocols. It is expected to exhibit better overall performance by being robust and superior in most of the parameter space. The impact of the new combined protocol on team performance are explored through simulation experiments in Chapter 5.

While investigating possible combinations of the two one-sided protocols, we have realized a limitation in the existing MAP protocols: an agent is not allowed to both provide and receive help at the same time. In our new combined protocol, simultaneous providing and receiving help act can potentially happen as there are both offers and requests made at the same time and an agent can potentially be involved in two different roles. Moreover, the practical impact of simultaneous providing and receiving direct help is of interest from an engineering perspective, as it enables the agents with specialized roles to more flexibly share their knowledge and expertise. An agent may be expert in a particular situation for which it can help one of its teammates who has difficulty in performance, but at the same time, it is also beneficial for the team if the agent dealing with an unexpected situation can receive help from another teammate who is expert in that situation. Hence, as a prerequisite step, we first refine the existing requester and helper-initiated protocols in order to enable the agents to provide and receive help simultaneously. The immediate impacts of refined protocols on team performance are examined through simulation experiments in Chapter 5.

3.2 The Refined Agent Team Model

In the previous studies of Action MAP (such as [Polajnar et al., 2012, Nalbandyan et al., 2013]), the cost of an action performed by an agent only depends on the particular type of action and the agent's skill profile, but not on the nature of the environment in which the agent is situated. Malek Akhlagh and Polajnar [2014] formalize the following refined agent team model, in which the cost of performing an action also depends on a component of the environment state that impacts the execution of the action. This refinement explicitly represents the impact of the environment state in the agent's action cost model. Using this refined model, the criteria for help deliberation such as estimating the cost of a plan or agent's individual wellbeing are not restricted to a particular microworld and can potentially be applied to different practical application domains.

A team consists of agents $A_1, A_2, \ldots, A_n$, $n > 1$, that operate in an environment E by performing actions from a domain Act. The team is assigned a task T, and each A_i is given an individual *subtask* T_i with a *budget* R_i. Each instance of action $a \in Act$ performed towards T_i has a cost that is charged to R_i. Let $Act^E = \{\alpha_1, \ldots, \alpha_m\}$, $m > 1$, be the set of all *augmented actions* of the form $< a, e >$, where e is the component of environment state that impacts the cost of performing a. Then the agent A_i performs α_k at a cost represented as a positive integer constant $cost_{ik}$. The vector $cost_i$ represents the A_i's *skill profile* with respect to the augmented actions, and the $n \times m$ matrix *cost* represents the individual abilities of all agents.

The environment is dynamic in the sense that its state can be changed by events other than agents' actions. Each agent maintains its own belief base through perception and communication, and acts rationally in the interest of the team. To model

agent interaction protocols without explicit synchronization details, it is assumed that agents perform actions in synchronous *rounds* and communicate only at the start of each round, in a sequence of synchronous *phases*.

3.3 The Refined Requester-Initiated Action MAP

In the requester-initiated approach, agents can proactively request, but they do not offer help. In the previous model, RIAMAP [Nalbandyan et al., 2013], agents that consider providing help can bid to requests, but if an agent has sent a request, it does not concurrently bid to other requests in the same round, resulting in only providing or receiving help in one round. In order to allow simultaneous providing and receiving of help, we present the refined Requester-Initiated Action MAP (RIAMAP*) that removes this restriction. In the refined model, an agent is allowed to concurrently request help in one protocol session, and bid to a request in another session. In other words, the agent is able to take the roles of bidder and requester at the same time. The potential advantage of the refined model becomes manifest when an agent has sent its request for having an expensive action performed by a teammate, and simultaneously bids to a request from another agent to perform a cheaper action in its skill profile. A protocol session comprises three interaction phases as follows. Its interaction sequence is illustrated in Fig. 3.1.

1) Help request generation:

This phase of RIAMAP* is identical to the request generation of RIAMAP (Section 2.6); we restate the formal conditions here to make the description of RIAMAP*

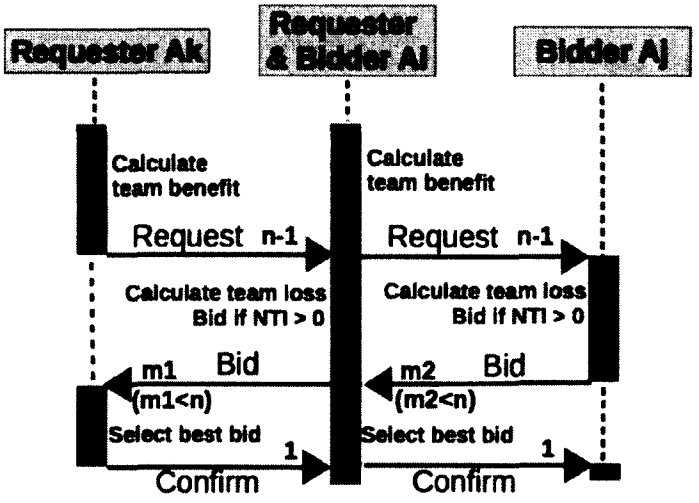


Figure 3.1: The RIAMAP* bidding sequence.

self-contained.

At the start of every round, agent A_i with next augmented action α_k at the cost of $cost_{ik}$ and the current calculated wellbeing of W_i broadcasts a help request message containing α_k and the calculated team benefit, Δ_i^+, if any of the following three conditions holds:

(i) $R_i < cost_{ik}$;

(ii) $W_i < W^{LL}$ and $cost_{ik} > LowCostThreshold$;

(iii) $cost_{ik} > RequestThreshold$;

where *LowCostThreshold* is the upper limit of the "cheap" action range, *Request-Threshold* is the lower limit of the "expensive" action range, and W^{LL} is a fixed *low watermark* value for wellbeing.

2) Bidding to a request:

All agents *including the ones who have sent requests*, receive the request from A_i and deliberate on bidding to it. The rest is identical as in HIAMAP (Section 2.7), restated here for completeness. An agent A_j calculates the team loss Δ_j^- for performing the requested action α_k and NTI using the received team benefit, Δ_i^+. If NTI is positive, A_j sends a bid containing α_k and the associated NTI value to the requester A_i. In case of having multiple qualified requests, A_j bids to the one with highest NTI. The potential advantage of the refined model becomes manifest when an agent has sent its request for having an expensive action; and it also bids to requests from other agents for cheaper actions in its skill profile.

3) Confirming the chosen bid:

Agent A_i receives the bids, selects the one with highest NTI, and sends a confirmation to the selected bidder agent A_j.

3.4 The Refined Helper-Initiated Action MAP

In the helper-initiated approach, agents can proactively offer, but they do not otherwise request help. In the previous model, HIAMAP [Nalbandyan et al., 2013], agents that consider receiving help can bid to offers, but if an agent has sent an offer, it does not concurrently bid to other offers in the same round, precluding the simultaneous providing and receiving of help. In order to remove this restriction, we present the refined Helper-Initiated Action MAP (HIAMAP*), which allows an agent who has

sent a help offer to concurrently bid to offers from other agents in the same round, taking the roles of offerer and bidder at the same time, and may as a result provide and receive help simultaneously. The potential advantage of the new model becomes manifest when the agent has offered help for some low cost actions in its skill profile, and simultaneously bids to an offer from another agent to receive help for its next expensive action. A protocol session comprises three interaction phases as follows. Its interaction sequence is illustrated in Fig. 3.2.

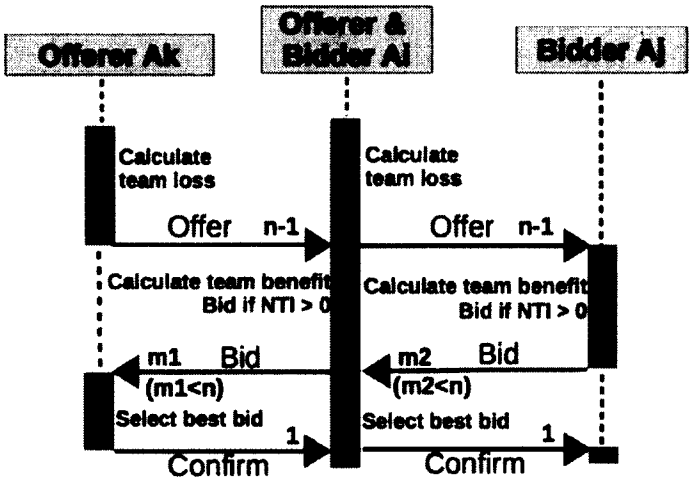


Figure 3.2: The HIAMAP* bidding sequence.

1) Help offer generation:

This phase of HIAMAP* is identical to the offer generation of HIAMAP (Section 2.7); we restate the formal conditions here to make the description of HIAMAP* self-contained.

At the start of every round, agent A_i calculates its individual wellbeing W_i. If W_i is above a fixed *high watermark* W^{HH}, it broadcasts an offer message containing pairs $[\alpha_k, \Delta_i^{(k)-}]$ for each augmented action α_k for which A_i's cost is below *OfferThreshold* parameter, where $\Delta_i^{(k)-}$ denotes the team loss associated with α_k.

2) Bidding to an offer:

All agents *including the ones who have sent offers*, receive the offer from A_i and deliberate on bidding to it. The rest is identical as in HIAMAP (Section 2.7), restated here for completeness. If an agent A_j finds that its next action α_k is included in the offer, it calculates the team benefit for not performing the offered action, $\Delta_j^{(k)+}$, and uses the team loss, $\Delta_i^{(k)-}$, received in the offer, to calculate the value of NTI. If NTI is positive, A_j sends a bid containing α_k and the associated NTI value to A_i. In case of having multiple qualified offers, A_j bids to the one with highest NTI. The potential advantage of the new model becomes manifest when an agent has offered help for actions that have low costs in its skill profile, while simultaneously bidding to offers from other agents for its own next action, which has a high cost in its skill profile.

3) Confirming the chosen bid:

Agent A_i receives the bids, selects the one with highest NTI, and sends a confirmation to the selected bidder agent A_j.

3.5 The State-Machines of the Refined Protocols

In the simulation models, we represent agent interaction protocols as finite state machines of a particular type, with synchronous communication and alternating states for sending and receiving messages. The state-machine formalism is then supported by full specification of the agent's deliberation and interactive behavior in each specific state together with the respective transition rules. Such representations are easily translatable to executable code in the AIMS framework.

In the state-machine representation of the refined one-sided protocols, at the end of every protocol session, each agent A_i is in one of the following final states that determines its team-oriented behavior based on its interaction with the teammates and deliberation on receiving or providing help:

Blocked:	for shortage of resources and not receiving help from teammates;
OwnAct:	decided to perform its own action and not engage in a help act;
GetHelp:	committed to receive help from a teammate and have its next action performed at no cost;
HelpAct:	committed to provide help by performing a teammate's next action instead of its own;
Help&GetHelp:	committed to both receive and provide help simultaneously.

Fig. 3.3 and 3.4 illustrate the state-machine representations of RIAMAP* and HIAMAP*, respectively. A sample state specification is presented in Fig. 3.5 for RIAMAP* and in Fig. 3.6 for HIAMAP*.

As an illustration, consider the behavior of agent A_i in RIAMAP* (Fig. 3.1). Initially, the agent is in state *S-Request*. If it decides to broadcast a request, it transits

to state *R-Request2*, in which it receives requests from other agents. Then it transits to *S-Bid2*, where it deliberates on bidding to the requests. If the deliberation results in bidding to a request, the agent transits to *R-Bid2*, where it receives bids from other agents for its own request. If it has received any bids, it transits to *S-Confirm2*, where it sends a confirmation to the selected bidder. In the case that confirmation was sent, it transits to *R-Confirm3*, where it may receive confirmation for its own bid to another agent's request. If it does receive the confirmation, it transits to *S-Act4*. Finally, it transits to *R-Help&GetHelp*, which results in providing and receiving help simultaneously.

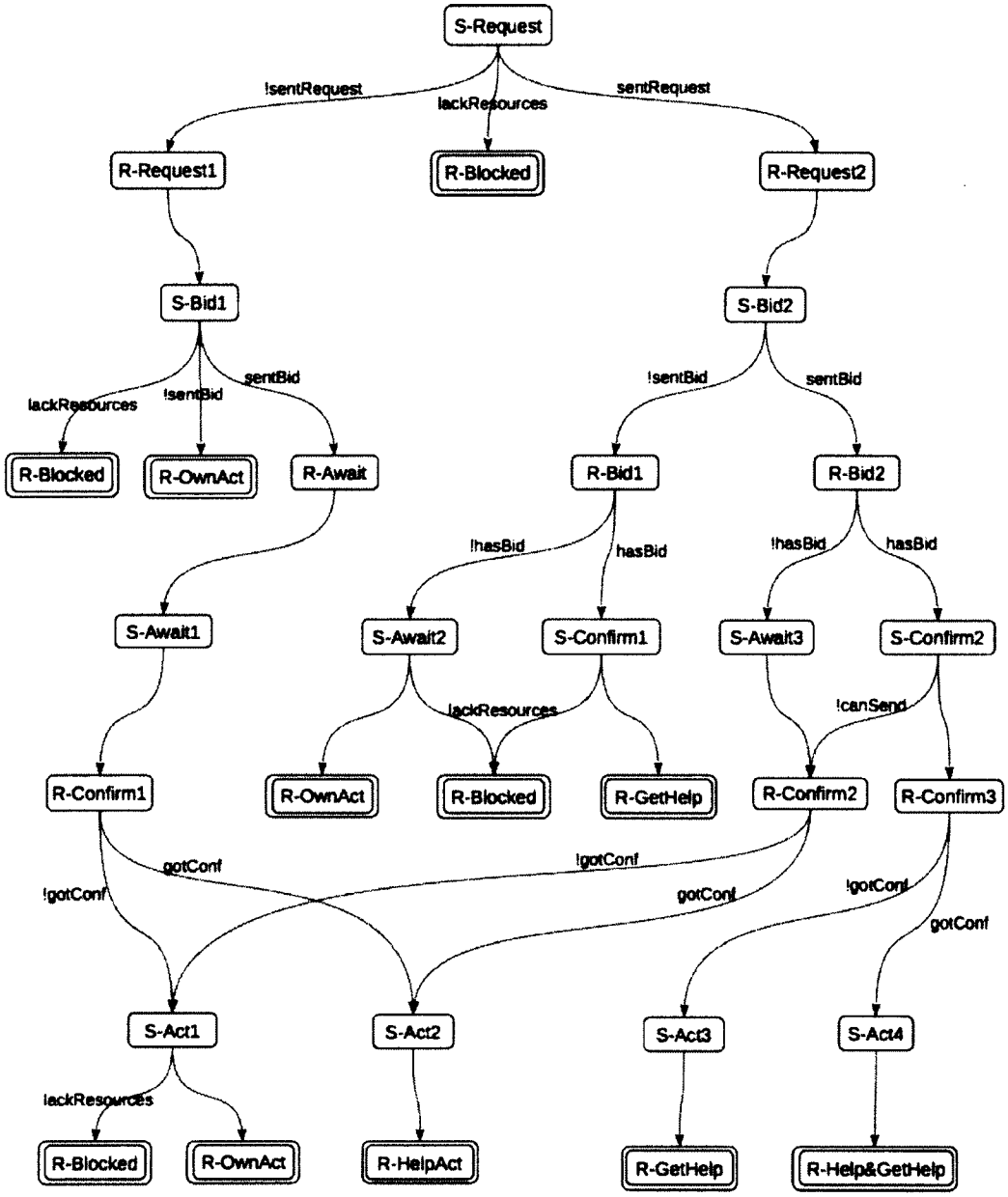


Figure 3.3: The RIAMAP* state-machine representation. The agents participating in the protocol execute interacting state machines. The execution strictly alternates between sending and receiving states (denoted S and R, respectively). Double-line boundaries indicate final states.

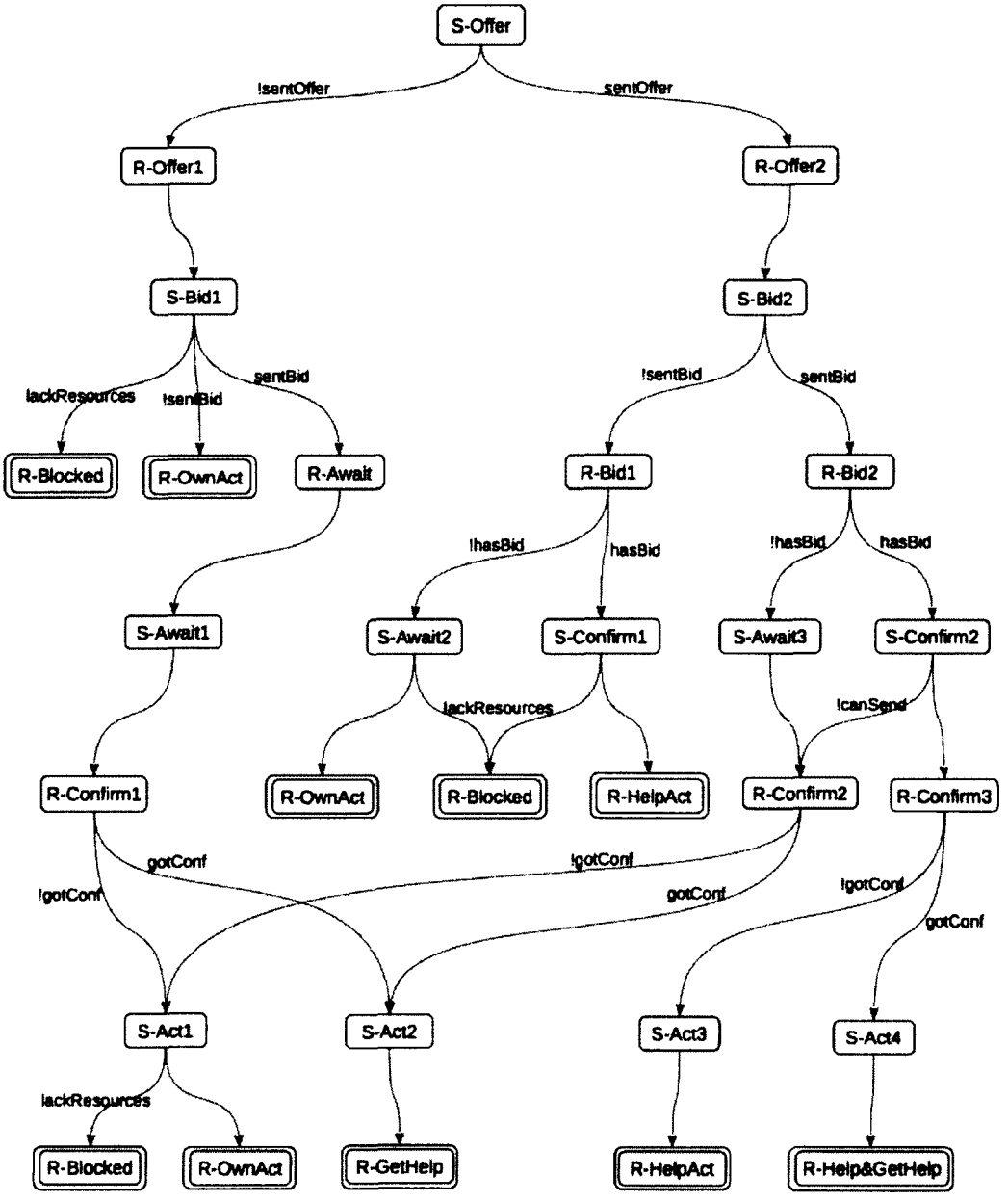


Figure 3.4: The HIAMAP* state-machine representation.

```
State S-Bid1:
{
    if (number of received requests > 0)
        for (each received request)
            calculate NTI;
        if (highest NTI > 0 && has enough resources to send a bid)
            send a bid to the requester with the highest NTI;
            go to state R-Await;
        else if (has enough resources to perform next action)
            go to state R-OwnAct;
        else
            go to state R-Blocked;
    else if (has enough resources to perform next action)
        go to state R-OwnAct;
    else
        go to state R-Blocked;
}
```

Figure 3.5: A RIAMAP* sample state specification, shows an instance of the deliberation on transitions from a given state. The states specifications are translated to executable code in the AIMS framework.

```
State R-Bid2:
{
    read received bids;
    if (number of received bids > 0)
        bestBid := select the bid with highest NTI;
        go to S-Confirm2;
    else
        go to S-Await3;
}
```

Figure 3.6: A HIAMAP* sample state specification.

Chapter 4

The Bidirectionally Initiated Action MAP

This chapter presents the Bidirectionally Initiated Action MAP (BIAMAP), which combines the refined requester and helper-initiated protocols, RIAMAP* and HIAMAP*. Section 4.1 provides an analysis of the requester and helper-initiated behavior over all parameter space, which highlights the motivation for combining the requester and helper-initiated approaches, Section 4.2 presents a discussion on possible combinations of the two one-sided protocols, Section 4.3 presents the design and specification of BIAMAP, and Section 4.4 provides its state-machine representation.

4.1 Requester vs. Helper-Initiated Behaviors

The design of the new combined protocol is expected to balance the requester and helper-initiated behaviors in such a way as to maximize the advantages of the two one-sided approaches. In order to do that we need an analysis of their behavior over all parameter space. In the sequel, starting from the observations by Nalbandyan et al. [2013], we analyze their relative performance in some critical situations with respect to environment dynamism, initial resources, and communication cost, in order to determine how their performance profiles complement each other.

Environment Dynamism

Generally, a high level of environment dynamism hampers the helper-initiated protocol significantly more than the requester-initiated protocol. When the environment state changes at a low rate, the helper-initiated protocol dominates with high initial resources. This happens as in this situation, a typical agent follows its initial selected plan without dramatic changes of costs and therefore it is able to make offers to other agents whenever its wellbeing is higher than the offering threshold. But when the environment change rate is high, a typical agent's wellbeing is below the offer threshold and the agents make fewer offers resulting in fewer help acts. On the other hand, the requester-initiated protocol dominates because it can adjust its teamwork to dramatic changes by broadcasting requests particularly when the communication cost is low.

Initial Resources

Generally, decreasing the initial resources hampers the helper-initiated protocol significantly more than the requester-initiated protocol. When the initial resources are low, for the helper-initiated protocol, a typical agent's wellbeing is below the offering threshold and the agents cannot make offers. On the other hand, the requester-initiated protocol dominates especially with low communication cost. This happens as with low initial resources, typical agent wellbeing is below the requesting threshold and agents can make requests in case of resource shortage while this increased broadcasting activity is mildly penalized with low communication cost. But when initial resources are high, the helper-initiated protocol dominates since typical agent wellbeing is above the offering threshold and agents are able to make more offers to enhance the team performance.

Communication Cost

As the communication cost increases, it hampers the requester-initiated protocol significantly more than the helper-initiated protocol. Hence, the helper-initiated protocol dominates with high communication cost. This is mainly because of the way in which two protocols generate help requests and help offers: the agents with declining wellbeing generate more requests in the requester-initiated protocol, but fewer offers in the helper-initiated protocol. Hence, in the requester-initiated protocol, an increased cost of broadcasting depletes the remaining resources. However, when the communication cost is low, helpful behavior can be provided at a lower cost and the agent team can benefit from increased broadcasting activity with little penalty. Hence, the requester-initiated protocol dominates when communication cost is low.

The design of the new combined protocol seeks to optimize the balance between the requester and helper-initiated behavior in order to leverage the advantages of each across the parameter space and integrate their strengths into a single protocol with superior performance. The design process involves the composition of two individual state machines into one, and performance optimization of the new protocol through iterative adjustment of its parameter values in a series of simulation experiments. The AIMS framework has been specifically developed to support this type of incremental improvement of interaction protocol design.

4.2 Combining the One-Sided Protocols

One might envision different possible combinations of the requester and helper-initiated protocols. One possible combined mechanism is to allow both one-sided protocols to initiate the deliberation on getting involved in helpful behavior, whether to provide or receive direct help, by broadcasting a help request and/or an offer, possibly in a combined message, in the same interaction phase. This approach might increase the quality of finding a match between a requester and a potential helper, because of the variety of available requests and offers to all agents; however, it can increase the complexity of deliberation by incurring extra computation cost of finding team benefit or team loss, and increase the communication cost of broadcasting requests and offers in the same interaction phase, without informed pre-estimation of the need for sending a message.

An alternative approach is to impose a fixed order on the request generation and offer generation phases in the combined mechanism, with one of the two phases always preceding the other. This specified order of interaction phases can be incorporated in

agents' deliberation on whether to generate a help request or an offer.

The design is which the request generation phase precedes the offer generation phase allows agents to broadcast help requests that may be rendered unnecessary by subsequent offers. When resources are low, the requests become frequent, and unnecessary communication expenditures become prohibitive. The rationale for the alternative design, in which the help offer generation comes first, is to let the agents first consider what they are able to offer, and then deliberate on making requests based on what offers they have received. This approach avoids the unnecessary help request generation; we adopt it as basis for our design of the combined protocol.

4.3 The Bidirectionally Initiated Action MAP

The Bidirectionally Initiated Action MAP (BIAMAP) incorporates the proactive requesting and proactive offering of direct help in a single protocol, by realizing that the deliberation on help can potentially be initiated from both helper and requester sides in a combined mechanism design. In BIAMAP, an agent is allowed to make an offer and/or a request in their respected interaction phases. In other words, the agent can potentially take the roles of offerer and requester in the same protocol session. It can then bid to an offer and/or a request based on whether it has made an offer and/or a request.

A protocol session comprises four interaction phases, one more in comparison to the requester-initiated and helper-initiated protocols. For instance, an agent needing help can either bid to received help offers or broadcast a help request. In the current design, the agent should do the latter only if suitable offers are not available. The

interaction phases are described as follows.

1) Help offer generation:

At the start of every round, agent A_i deliberates on offering helpful behavior to its teammates. In case it decides to provide action help, it broadcasts its offer.

2) Help request generation:

Having received offers from teammates, A_i deliberates whether it needs help for its next action. In case it determines its need, it processes the offers by calculating the NTI value for those offers which match with its next action. If its next action matches with any offer for which the NTI value is positive, it does not send a help request. Otherwise, in case there is no suitable offer, it broadcasts a help request.

3) Bidding to requests and/or offers:

Once A_i has received all offers and requests from its teammates, four different situations arise, depending on whether A_i has sent a help offer and/or help request, described in the following.

Case 1. A_i has not sent any help offer or request. In this case, it considers bidding to both the received offers and requests. It deliberates and decides whether to bid to an offer, a request, or both. The protocol interaction sequence for this situation is illustrated in Fig. 4.1.

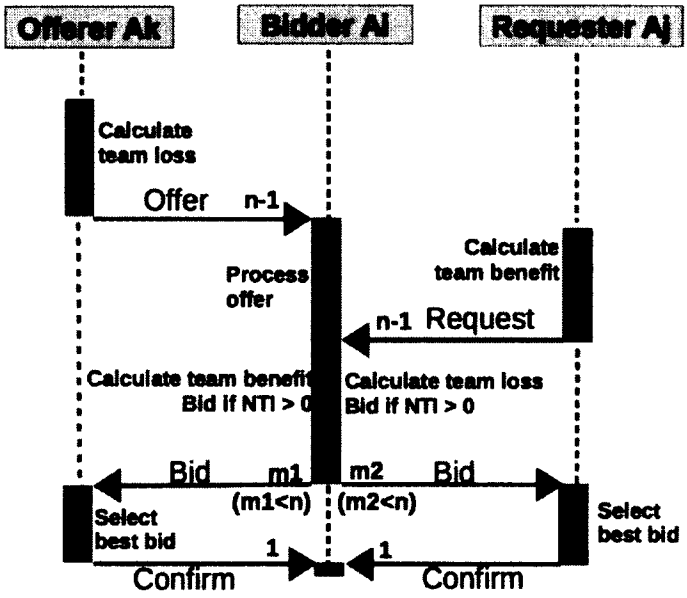


Figure 4.1: BIAMAP case 1: A_i has not sent an offer or a request.

Case 2. A_i has not sent a help offer, but has sent a help request. In this case, it considers bidding to the requests but not to the offers. Hence it only deliberates and decides on bidding to a request. The protocol interaction sequence for this situation is illustrated in Fig. 4.2.

The rationale for not bidding to offers in this case is that A_i has already considered the available offers in the request generation phase, but did not find a suitable one and hence decided to send a help request.

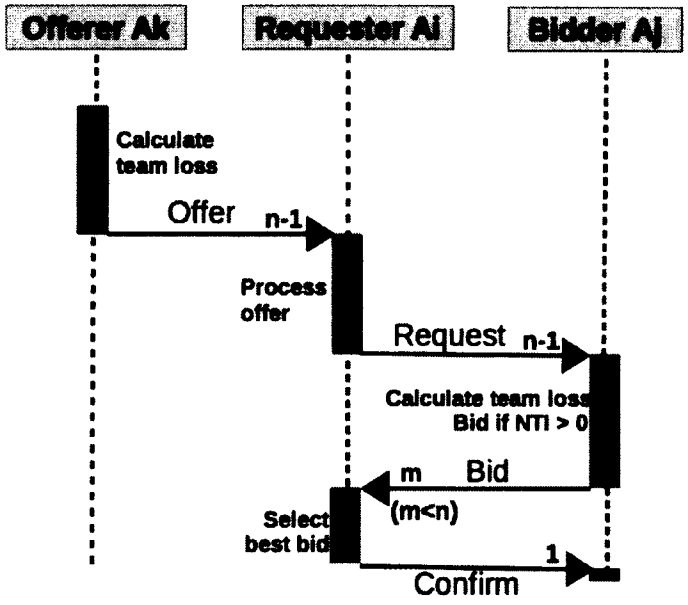


Figure 4.2: BIAMAP case 2: A_i has sent a request and has not sent an offer.

Case 3. A_i has sent an offer, but has not sent a request. In this case, it considers bidding to the offers but not to the requests. Hence, it only deliberates and decides on bidding to an offer. The protocol interaction sequence for this situation is illustrated in Fig. 4.3.

The rationale for not bidding to requests in this case is that the agents who have sent requests have already considered the available offers from all agents, including A_i, but decided to send a request.

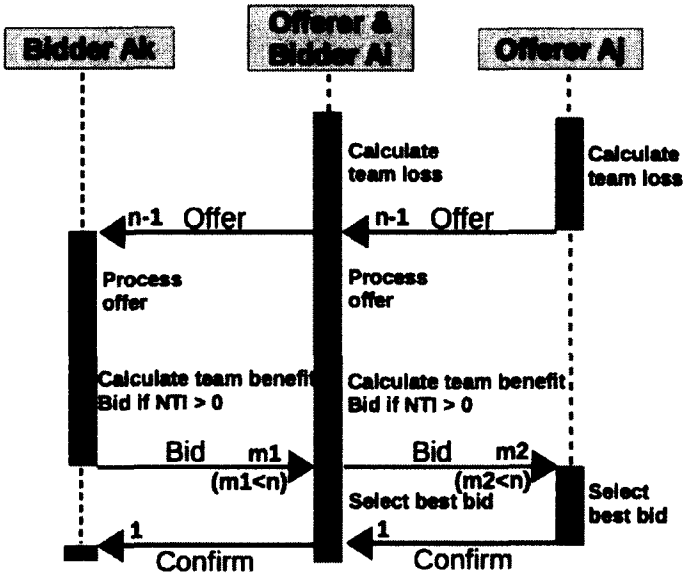


Figure 4.3: BIAMAP case 3: A_i has sent an offer and has not sent a request.

Case 4. A_i has sent both an offer and a request. In this case, it does not consider bidding to any of the received offers or the requests. The protocol interaction sequence for this situation is illustrated in Fig. 4.4.

The rationale for not bidding to offers and requests in this case consists of the two reasons already given in cases 2 and 3.

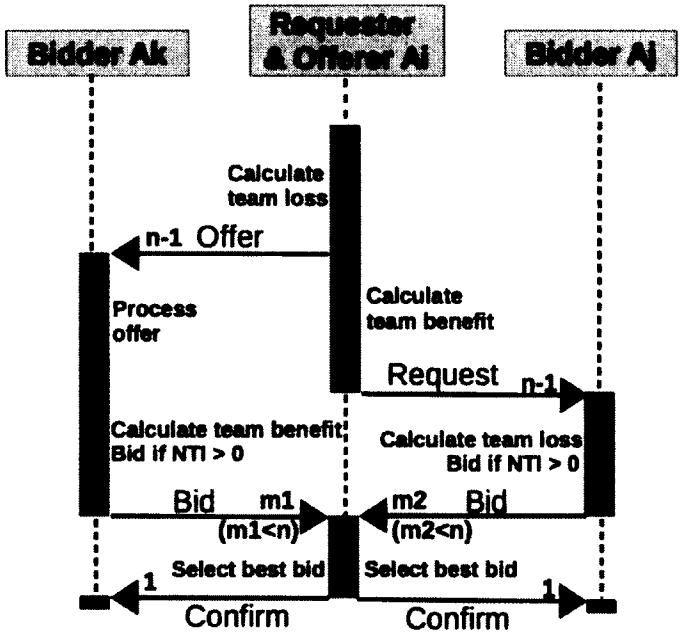


Figure 4.4: BIAMAP case 4: A_i has sent an offer and a request.

4) Confirming the chosen bids:

In this phase, the agent A_i who has sent a help offer or a request, receives possible bids to its offer or request. In each case, it selects the bid with highest NTI and sends a confirmation to the selected bidder agent. In the case that A_i has sent both an offer and a request, it may receive bids for both, hence it may send confirmations to two selected bidders.

4.4 The State-Machine Representation of BIAMAP

We represent BIAMAP using the state-machine formalism illustrated in Fig. 4.5, supported by full specifications of states together with the respective transition rules. At the end of every protocol session, each agent A_i is in one of the final states introduced in Section 3.5, which determines the agent's behavior in the current round: *Blocked*, *OwnAct*, *GetHelp*, *HelpAct*, or *Help&GetHelp*.

As an illustration, consider the behavior of agent A_i in BIAMAP case 3 (Fig. 4.3). Initially, the agent is in state *S-Offer*. If it decides to broadcast an offer, it transits to state *R-Offer2*, in which it receives offers from other agents. Then it transits to *S-Request2*, where it deliberates on sending a request based on the offers it may have received. If it decides not to make a request, it transits to *R-Await1*, in which it ignores requests from other agents. Next, it transits to *S-Bid3*, where it only deliberates on bidding to the offers. If the deliberation results in bidding to an offer, the agent transits to *R-Bid4*, in which it receives bids from other agents for its own offer. After that it transits to *S-Confirm4* and, if it has received any bids, sends a confirmation to the selected bidder. In the case that confirmation was sent, it transits to *R-Confirm5*,

where it may receive confirmation for its own bid to another agent's offer. If it does receive the confirmation, it transits to *S-Act4*. Finally, it transits to *R-Help&GetHelp*, which results in providing and receiving help simultaneously.

A sample state specification is presented in Fig. 4.6. The state specifications are translated into executable code in the AIMS framework.

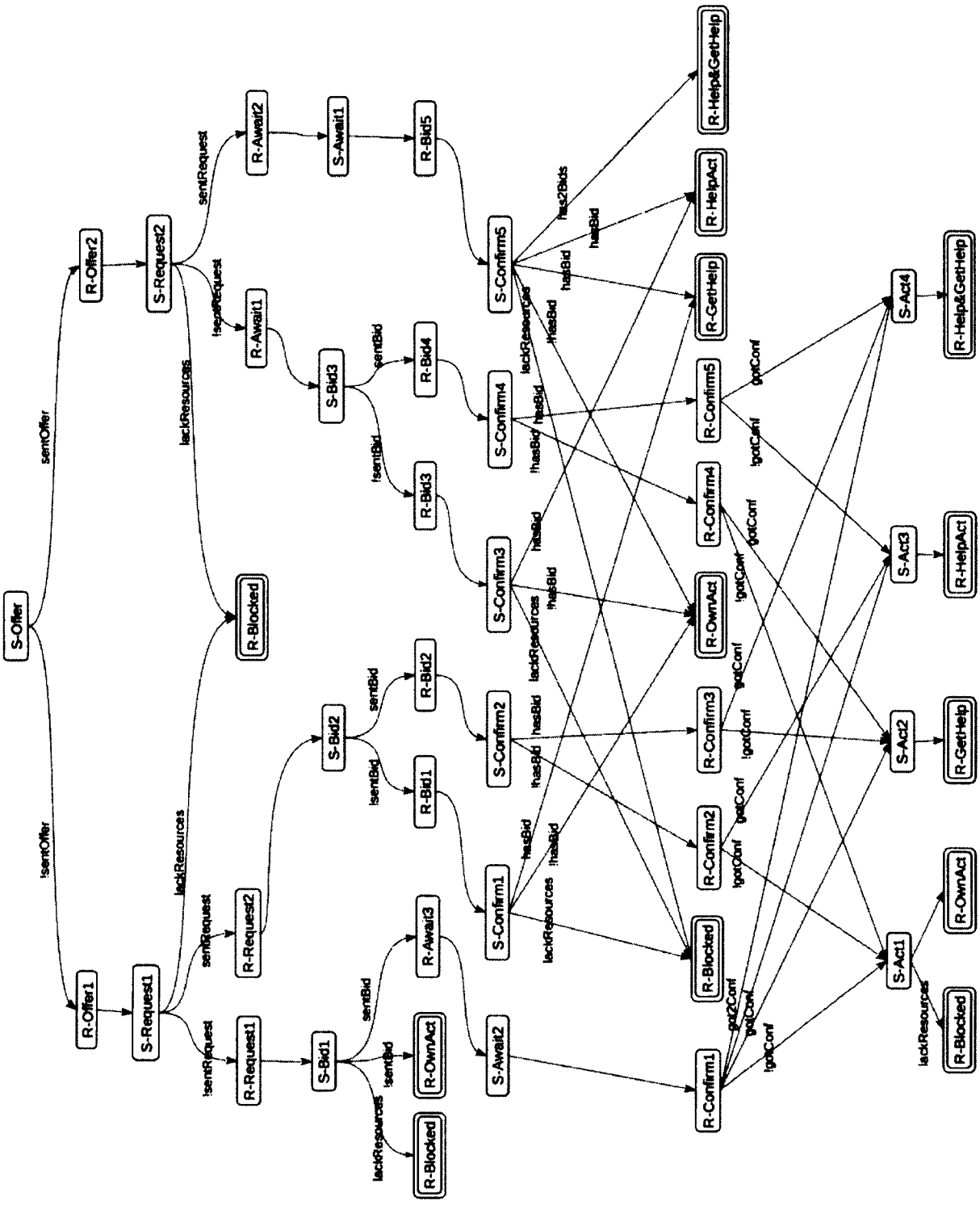


Figure 4.5: The BIAMAP state-machine representation.

```
State S-Request1:
{
        if (has accomplished its subtask)
                go to state R-Request1;
        else if (needs help for its next action)
                if (next action matches with any of offered actions for which NTI > 0)
                        go to state R-Request1;
                else if (has enough resources to broadcast)
                        broadcast request;
                        go to state R-Request2;
                else
                        go to state R-Blocked;
        else
                go to state R-Request1;
}
```

Figure 4.6: A BIAMAP sample state specification.

Chapter 5

Evaluation

This chapter presents the simulation models and performance results for agent teams that employ our new interaction protocols, in comparison with teams that employ previously existing versions of MAP. Using the AIMS framework, we compare the performance of teams through their concurrent simulations in identical dynamic environments for different values of parameters representing environment dynamism, agent resources, and communication costs. The simulation models are based on the same board game microworld (Subsection 5.1.1), the same model of Action MAP (Subsection 5.1.2), and the same parameter settings (Subsection 5.2.1), that were used in previous studies of MAP [Nalbandyan et al., 2013]. In a preliminary round of experiments (Subsection 5.2.2), we optimize the values of wellbeing thresholds (watermarks), that are subsequently used in deliberation on making a help request or offer. In the next round of our simulation experiments (Subsection 5.2.3), we explore the team performance impact of the refinements in the requester and helper-initiated protocols, i.e., the impact of allowing agents to simultaneously provide and receive help.

We examine whether the refined one-sided protocols, RIAMAP* and HIAMAP*, outperform their previously existing counterparts, RIAMAP and HIAMAP, respectively. The next round of experiments (Subsection 5.2.4) demonstrates the complementary behavior of RIAMAP* and HIAMAP*, analogous to the previously observed complementarity of RIAMAP and HIAMAP. In the final round of experiments (Subsection 5.2.5), we explore the team performance impact of combining the requester and helper-initiated approaches into a single protocol. Using the same parameter settings as in the previous round, we examine our new combined protocol, BIAMAP, in comparison to RIAMAP* and HIAMAP*. The observed behavior and performance confirm the design expectations of exploiting the complementarity and leveraging the strengths of the two constituent protocols.

5.1 The Simulation Models

We study agent interaction protocols for direct help in the context of a board game microworld, inspired by the Colored Trails game [Gal et al., 2010] and introduced in [Polajnar et al., 2012]. Its demonstrated suitability for the analysis of the impact of help protocols on team performance in several publications on MAP makes it fully adequate for our comparative studies of new and existing MAP variations in this research. We use the AIMS framework, introduced in Alemi et al. [2014], which allows concurrent simulation of multiple teams in identical dynamic environment. It facilitates representation of agent interaction protocols and modeling of environment dynamism. We use it to compare the performance of teams that employ different help protocols, but are otherwise identically designed and pursue identical tasks in identical dynamic environments.

5.1.1 The Microworld

The players in the game are software agents. The board is a rectangle divided into squares of different colors from a color set $S = \{S_1, \ldots, S_m\}$, as shown in Fig. 5.1. The color of a square represents the component of the environment state that impacts the cost of performing the agent's action of moving to that square. At the start of the game, the board color setting is uniformly random, and each agent A_i is randomly assigned an initial location L_i and a goal G_i on the board. Agents are able to observe the entire board. A_i moves toward G_i, which represents A_i working on its assigned subtask T_i. The resource budget R_i allocated for T_i is defined as $l_i a'$, where l_i is the shortest distance (i.e. number of squares) from L_i to G_i, and a' a positive integer constant called the initial resources coefficient. The game proceeds in synchronous rounds. In every round, A_i can move to a neighboring square, which represents performing an action. Agents are allowed to be on the same square at the same time.

The cost of making a move on the board in any direction depends only on the color of the square to which the agent moves, and on the agent that performs the move, but not on the direction. This makes the color alone correspond to the augmented action of Section 3.2. A_i's individual skill profile is represented as a vector $cost_i$ with positive integer values: when A_i moves to a square with color S_k, it pays $cost_{ik}$ from its resource budget R_i. All the cost vectors for all agents are included in a $n \times m$ positive integer matrix *cost*. A_i may be blocked for shortage of recourses required for its next move. The game terminates when no one can make a move because of either being blocked or having reached the goal.

The team performance is represented by *team score*, which is the sum of all *individual scores* calculated at the end of the game as follows. If A_i has reached its goal,

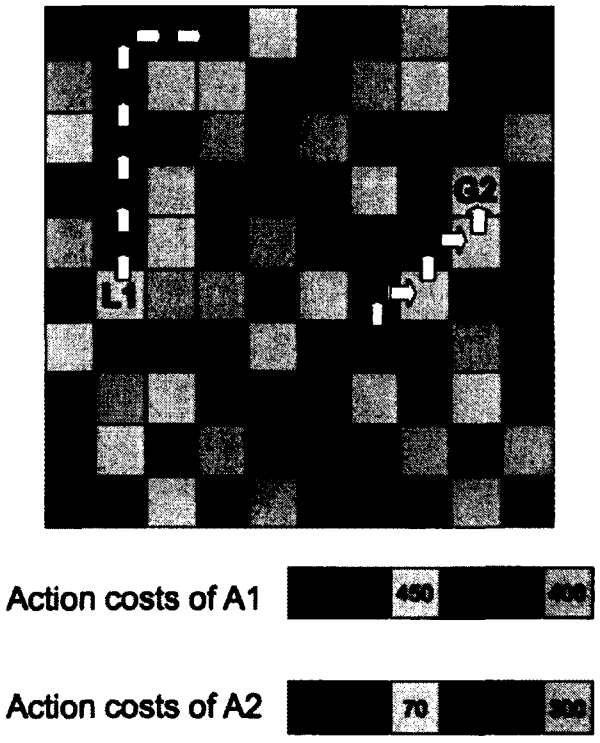


Figure 5.1: The board game microworld. Each agent A_i has its individual vector $cost_i$, indexed by the color set. At the start of the game, A_i plans its path by estimating the least-cost path among the shortest paths, from its initial location L_i to its individual goal G_i. The board colors change dynamically, affecting costs of the selected paths. [Adapted from [Nalbandyan et al., 2013].]

its individual score is the *goal achievement reward* g_i; if it is blocked, its score is $d_i a''$, where d_i is the number of steps it has made, and a'' is a positive integer constant called the *cell reward.*

The environment dynamism is represented by the board color setting changing after each round: the color of any square can be replaced by a uniformly random choice from the color set S. The replacement occurs with a fixed probability D, called the

level of disturbance. Initially, each agent selects its *plan* as the least-cost path among the shortest paths to its goal, and commits to it for the entire game. However, the environment evolves as the board colors change after each round, affecting the cost of the selected path.

5.1.2 The Simulation Models of Action MAP

In the analysis of MAP, we assume that agent A_i knows the stochastic behavior of the environment dynamism. The agent does not know the disturbance value, D, but can estimate it by recording the frequency of changes in the board. The agent applies the estimated value of D in calculating the expected cost of remaining path and individual wellbeing as follows. The selected path P_i of length ℓ, consists of actions $\alpha_0, \ldots, \alpha_{\ell-1}$. $\bar{c}_i$ is the average expected cost of an action for A_i. $\sigma = 1 - D$ represents the *stability* of the dynamic environment. Hence, the estimated cost of $k-th$ round action, α_k, is $\sigma^k cost_i(\alpha_k) + (1 - \sigma^k)\bar{c}_i$. By adding the estimated costs for all the actions in P_i, the expected cost for A_i to accomplish P_i is:

$$Ecost_i(P_i) = (\ell - \frac{1 - \sigma^\ell}{1 - \sigma})\bar{c}_i + \sum_{k=0}^{\ell-1} \sigma^k cost_i(\alpha_k) \tag{5.1}$$

The individual wellbeing is then calculated as in Chapter 2:

$$W_i = \frac{R_i - Ecost_i(P_i)}{(\ell + 1)\bar{c}_i} \tag{5.2}$$

with $Ecost_i(P_i)$ provided by (5.1).

In the deliberation on whether to perform the help transaction, the two parties in

the bilateral distributed agreement (BDA) calculate team benefit or team loss of the help act from their own perspectives. An agent A_i who is receiving help calculates the team benefit and an agent A_j who is providing help calculates the team loss, with incorporation of the notion of proximity to the goal, as follows:

Team benefit:

$$\Delta_i^+ = \delta_{hn}^{(i)}(\ 1 + pbias(\ell_h^{(i)}) - pbias(\ell_n^{(i)})\ ) \tag{5.3}$$

Team loss:

$$\Delta_j^- = \delta_{nh}^{(j)}(\ 1 + pbias(\ell_n^{(j)}) - pbias(\ell_h^{(j)})\ ) \tag{5.4}$$

where:

$\delta_{hn}^{(i)}$ – the difference in A_i's individual scores at the end of the game with and without the help act;

$\delta_{nh}^{(j)}$ – the difference in A_j's individual scores at the end of the game without and with the help act (including the help overhead h);

ℓ_h – the remaining path length at the end of the game if the help *is* provided;

ℓ_n – the remaining path length at the end of the game if the help *is not* provided;

pbias – the proximitiy bias function, which maps the remaining path length, ℓ, into a positive priority value, defined in this thesis as:

$$pbias(\ell) = \frac{A}{1+\ell} \tag{5.5}$$

where A>0 is the *proximity bias coefficient.*

The difference between team benefit and team loss is called *net team impact* (NTI). The help transaction occurs if NTI, $\Delta_{ij} = \Delta_i^+ - \Delta_j^-$, is positive.

Our final comment regarding this simulation model concerns its treatment of observability. Our agents observe the board with two purposes. The first one is planning, and it only requires that the agent see the rectangle determined by its initial position and its goal position. The second purpose is agent's perception of environment dynamism, currently represented by disturbance D. In this thesis, we assume that each agent can see the entire board and thus assess the disturbance with maximum accuracy. In general, MAP and its variations could be studied in conjunction with various observability models.

5.2 Performance Comparisons

5.2.1 The Parameter Settings

The game board has the size 10×10 with six possible colors. Each agent's cost vector includes three entries randomly selected from an 'expensive' range: $\{250, 300, 350, 500\}$ and three entries from a 'cheap' range: $\{10, 40, 100, 150\}$. Hence, each agent's skill profile is specialized for certain colors. Each team includes eight agents. The initial subtask assignment process is random. The *goal achievement reward* is 2000 points. The *cell reward* is 100 points. The help overhead, h, is 20 points. The *RequestThreshold* and *LowCostThreshold*, used in request generation process are 351 and 50, respectively. The *OfferThreshold*, used in the offer generation process, is 299. In our experiments, we vary: the *level of disturbance* in the dynamic environment, D;

the *initial resources* for each step of the path, a'; and the *communication cost* of sending a unicast message, U. The final team scores are averaged over 10,000 simulation runs, using random initial board settings.

Our simulation experiments use the same parameter space as previous studies of MAP [Nalbandyan, 2011, Polajnar et al., 2012, Nalbandyan et al., 2013]. The ranges of the parameters representing environment dynamism, agent resources, and communication cost are limited to a certain degree as follows. The disturbance level, D, does not exceed 0.5, as with a greater level of unpredictability in the environment, rational planning and deliberation are no longer meaningful. The initial resources coefficient, a', varies in a range that provides the agents with moderate amounts of resources, as too severe shortage discourages providing help and overabundance eliminates the need for receiving help. The unicast cost, U, is relatively low compared to action costs, as with very high communication costs, unilateral approaches have advantages over the bilateral approach in MAP [Nalbandyan, 2011, Polajnar et al., 2012].

5.2.2 Optimizing the Watermarks

An agent's individual wellbeing is incorporated in its deliberation on whether it can offer direct help or needs to receive help and hence to broadcast an offer or a request to its teammates. The wellbeing threshold values, *watermarks*, used in both the requester and helper-initiated approaches, need to be optimized in order to advance the agent's interactive behavior. In this section, we present simulation results for individually optimizing W^{LL} and W^{HH} for the refined requester-initiated protocol (RIAMAP*) and refined helper-initiated protocol (HIAMAP*), respectively, and then collectively

for the bidirectionally initiated protocol (BIAMAP). In the following experiments, the other parameters are kept constant at selected mid-range values in the parameter space: the *level of disturbance*, D, is 0.2; the *initial resources* for each step, a', is 160; and the *communication cost* of sending a message, U, is 9.

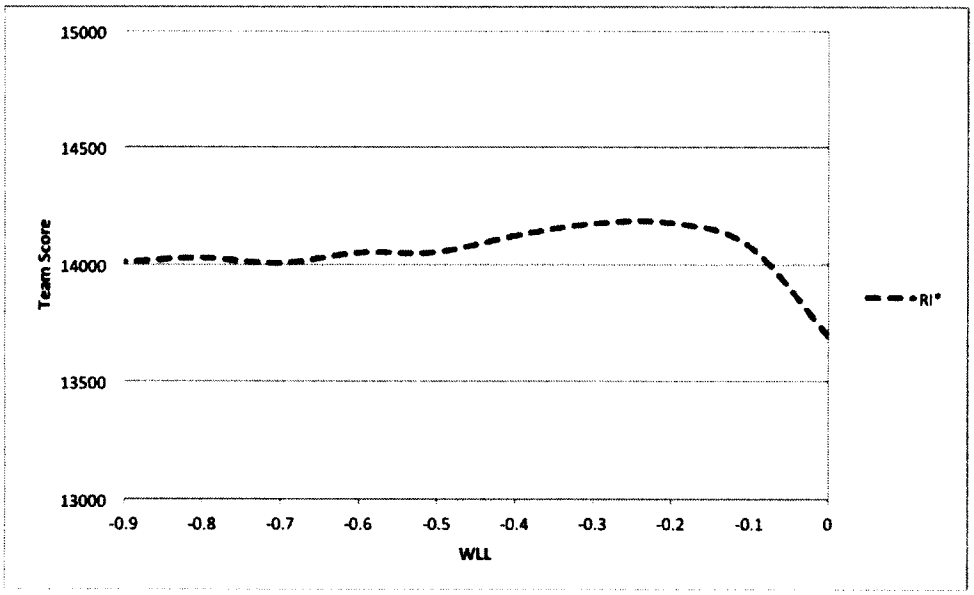


Figure 5.2: Team scores vs. W^{LL}

Fig. 5.2 shows the team performance of employing RIAMAP* while varying the *low watermark* value for proactive requesting of help, W^{LL}. As can be seen, the team performance is optimal when W^{LL} is close to -0.2. The general shape of this graph and the location of its maximum do not change significantly when we vary the values of disturbance, initial resources, and communication cost. Hence, we adopt the location of the maximum in this graph as the experimentally optimized value of W^{LL} for RIAMAP* in the rest of our simulation experiments.

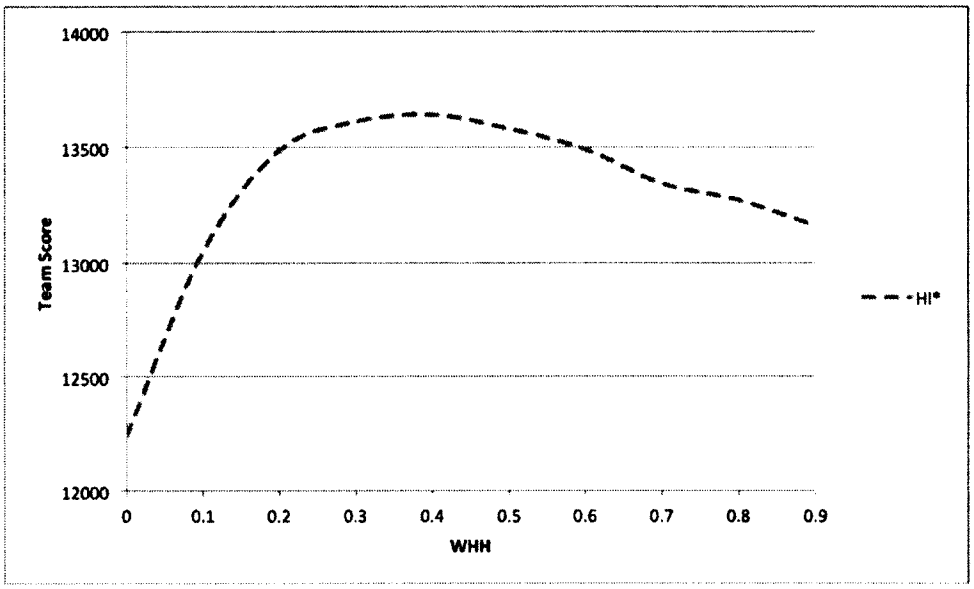


Figure 5.3: Team scores vs. W^{HH}

In a similar experiment, we observe the team performance of employing HIAMAP* while varying the *high watermark* value for proactive offering of help, W^{HH}. Fig. 5.3 shows the experimentally optimal value of W^{HH} found to be close to 0.4. Again, the location of this optimum is not very sensitive to variations in disturbance, initial resources, and communication cost values. Hence, we adopt $W^{HH} = 0.4$ for HIAMAP* in the rest of our simulation experiments.

Fig. 5.4 presents the performance of a team employing BIAMAP, which incorporates both requester and helper-sided deliberation, and uses both the *low* and *high* watermarks. Hence, we observe its team performance while varying both W^{LL} and W^{HH} parameters. It can be seen that BIAMAP team score highly depends on these wellbeing thresholds, and one needs to find their optimal values. In this particular

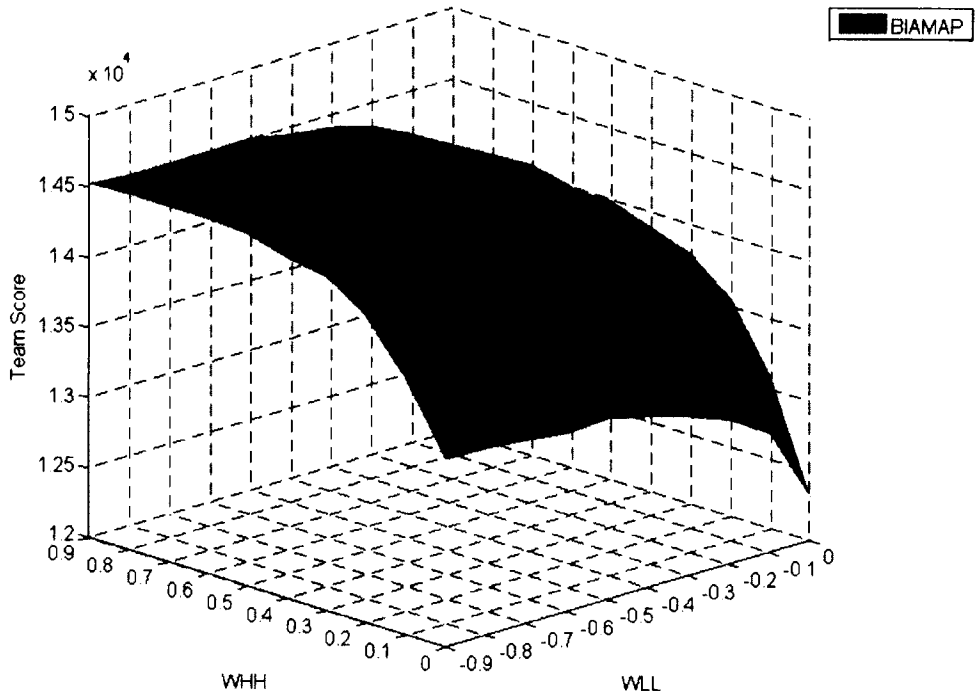


Figure 5.4: Team scores vs. W^{LL} and W^{HH}

situation, the maximum team score of 14547 is achieved when W^{LL} and W^{HH} are -0.8 and 0.6, respectively.

We henceforth adopt the optimized parameter values: W^{LL} = -0.8, W^{HH} = 0.6, along with the setting of other parameters used in the optimization process, namely $D = 0.2$, $a' = 160$, and $U = 9$, as the standard set of values for BIAMAP experiments. In every simulation experiment for evaluating BIAMAP, the parameters that are kept constant have these standard values.

5.2.3 The Team Performance Impact of RIAMAP* and HIAMAP*

In the following simulation experiments, we observe the team performance impact of the refined one-sided protocols that enable agents to provide and receive direct help simultaneously in both requester and helper-initiated approaches. We compare four teams that employ different interaction protocols: RIAMAP, RIAMAP*, HIAMAP, and HIAMAP*. The teams are otherwise identical and operate in identical environments. We explore the relative team performance in three experiments, in which we vary one of the parameters: level of disturbance, D, initial resources coefficient, a', or the unicast cost, U.

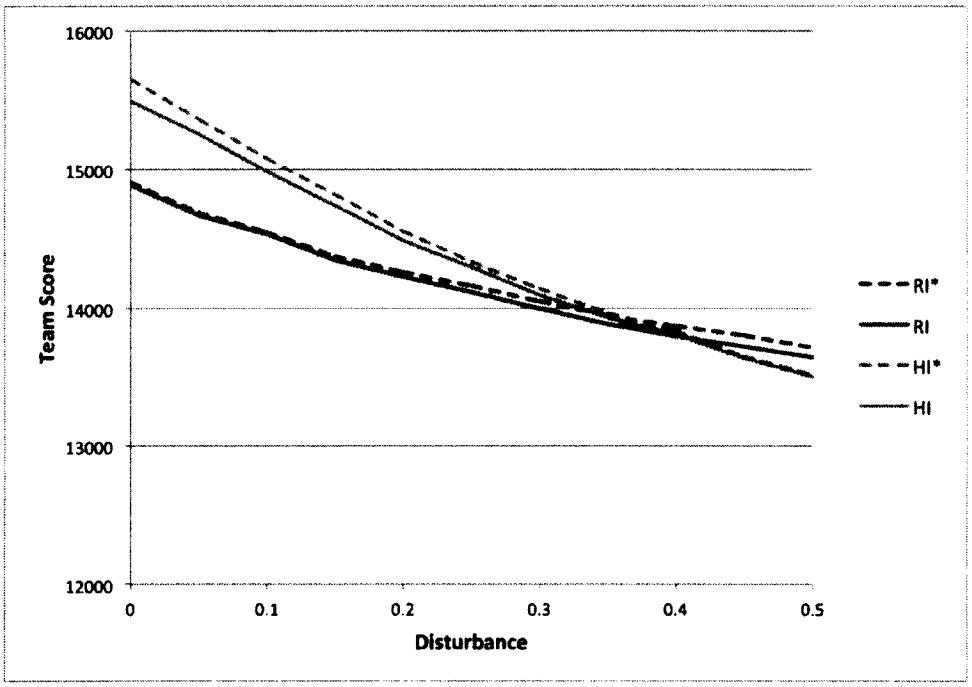


Figure 5.5: Team scores vs. Disturbance

Fig. 5.5 shows the comparative team scores for varying the level of disturbance, D. The other parameter values are set as $a' = 200$ and $U = 21$. The relatively high values of initial resources and communication cost favor the helper-initiated approach, HIAMAP*, when the disturbance is low. However, with high disturbance HIAMAP* degrades significantly and falls below RIAMAP*. One can note that each new protocol (RIAMAP* or HIAMAP*) outperforms the previous version (RIAMAP or HIAMAP, respectively), in situations when the respective one-sided behavior is more effective, as discussed in Section 4.1. For example, it can be observed that RI and RI* achieve same team score when there is no disturbance; but RI degrades more as disturbance increases, and RI* prevails at high disturbance. On the other hand, HI* outperforms HI when disturbance is low, but achieves almost same result when disturbance is high. As discussed before, this occurs because high disturbance hampers the helper-initiated protocols significantly more than the requester-initiated ones. The picture also illustrates the complementary performance profiles of the requester and helper-initiated approaches at different ends of the disturbance range.

In the second simulation experiment, we observe the comparative team scores of RIAMAP, RIAMAP*, HIAMAP, and HIAMAP* for varying the unicast cost, U. The other parameter values are set as $D = 0.2$ and $a' = 160$. We extend the communication cost up to 30, in order to explore its impact on different protocols. Fig. 5.6 shows the simulation results. As can be seen, the requester-initiated protocols, RIAMAP and RIAMAP*, outperform the helper-initiated protocols, HIAMAP, and HIAMAP*, in low U values; and then they degrade significantly more in high U values. The team performance impact of simultaneous providing and receiving help becomes manifest only when the requester or helper-initiated protocols are more effective, i.e. when they have advantage over the other one-sided protocols in particular situations. In this case, when communication cost is low, the requester-initiated approach is more

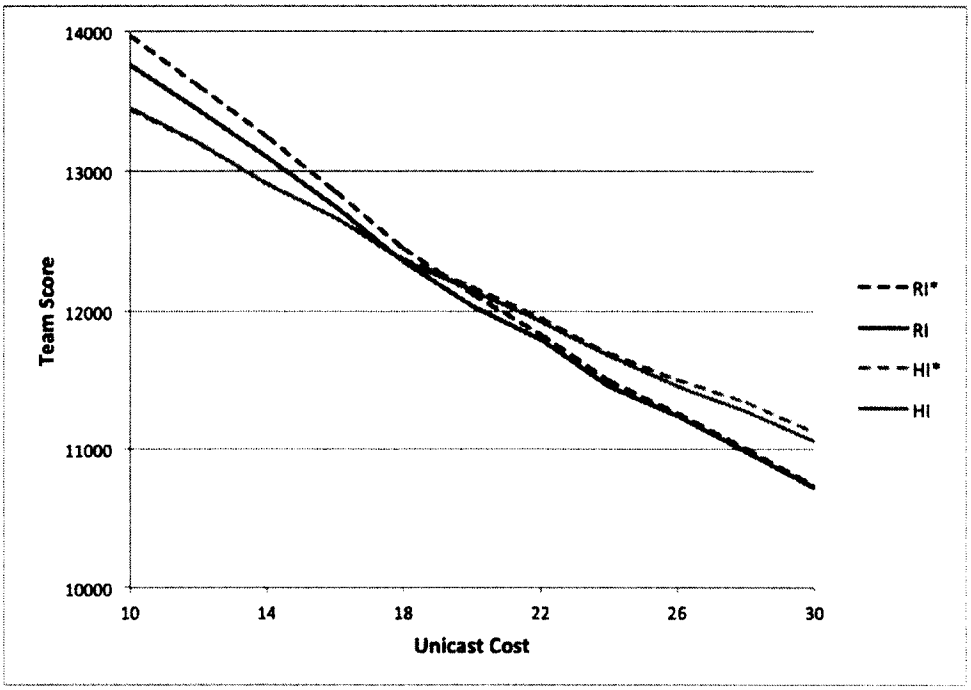


Figure 5.6: Team scores vs. Communication Cost

effective, and hence there is team performance gain by employing RIAMAP* over RIAMAP. At the other side, when communication cost is high, the helper-initiated approach has advantage over the requester-initiated one, and thus we observe some team performance gain by employing HIAMAP* over HIAMAP. Again, the complementary behaviours of the two different approaches are found in the results.

Fig. 5.7 presents the simulation results of the third experiment, in which we observe the behaviors of RIAMAP, RIAMAP*, HIAMAP, and HIAMAP* by varying the initial resources coefficient, a'. We set D to a relatively high value of 0.3, and U to a relatively low value of 3. In this situation, the requester-initiated approach has advantage over the helper-initiated one, as high disturbance hampers it less sig-

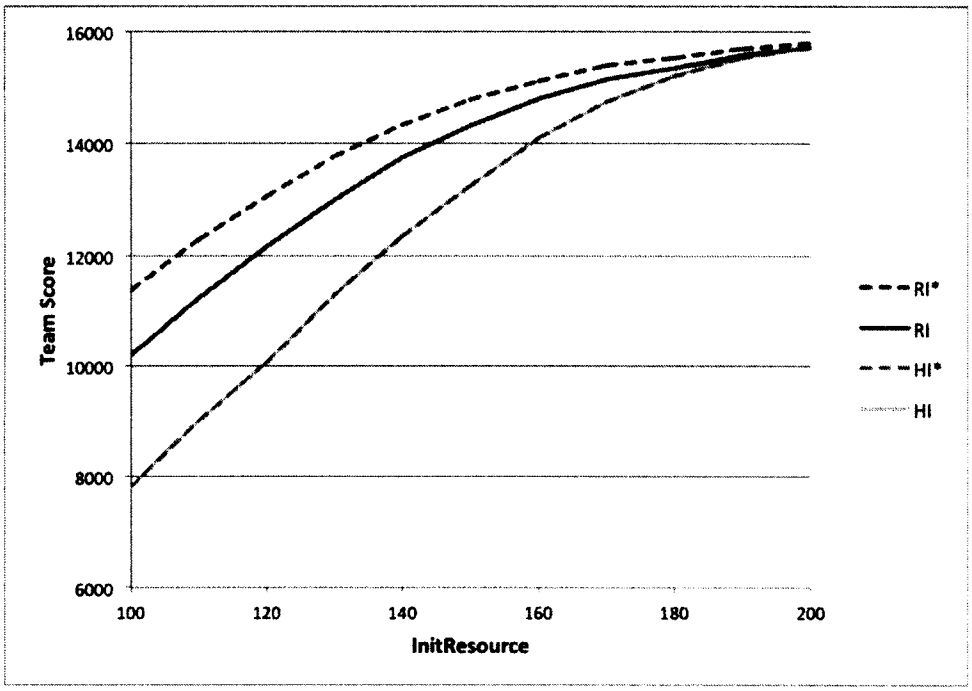


Figure 5.7: Team scores vs. Initial Resources

nificantly and it is more effective when communication cost is low. Accordingly, it can be seen that there is significant team performance gain by employing RIAMAP* compared to RIAMAP, i.e. allowing simultaneous providing and receiving help in the requester-initiated approach. However, with rise of the initial resources, the impact of this feature decreases, as the requester-initiated approach becomes less effective, i.e. there will be less need for making a request, and a typical agent may decide to perform its action on its own, rather than receiving help from a teammate. Furthermore, with more of initial resources, the helper-initiated approach achieves better result, and all the protocols achieve almost same team score when $a' = 200$. Also, the team performance impact of employing HIAMAP* over HIAMAP is not manifest

because of high level of disturbance and low communication cost, which make the helper-initiated approach less effective.

5.2.4 Complementary Profiles of RIAMAP* and HIAMAP*

In this section, we explore the complementary performance profiles of the refined requester and helper-initiated protocols (RIAMAP* and HIAMAP*) across the parameter space. We present three simulation experiments in order to compare the relative performance of two agent teams employing different help protocols, RIAMAP* and HIAMAP*, but otherwise identical and operating in identical environments. In each experiment, we vary two of the following three parameters: level of disturbance, D, initial resources coefficient, a', and the unicast cost, U; the third parameter is kept constant at the standard value selected in Section 5.2.2.

The experiments are designed to examine the requester and helper-initiated behavior hypothesis, discussed in Section 4.1. The behavior analysis investigates the efficiency of each one-sided approach in some critical situations with respect to environment dynamism, initial resources, and communication cost. The reasoning is centered on the agent's individual wellbeing, which impacts its deliberation on direct help. Generally, in a situation which results in low value of individual wellbeing, the requester-initiated approach is more effective; while with high value of individual wellbeing, the helper-initiated approach is more effective.

In the first experiment we vary the level of disturbance, D, and initial resources coefficient, a', while the unicast cost, U, is set to 9. Fig. 5.8 presents the comparative team scores achieved by RIAMAP* and HIAMAP*. It can be observed that the requester-initiated protocol, RIAMAP*, dominates when disturbance is high. This

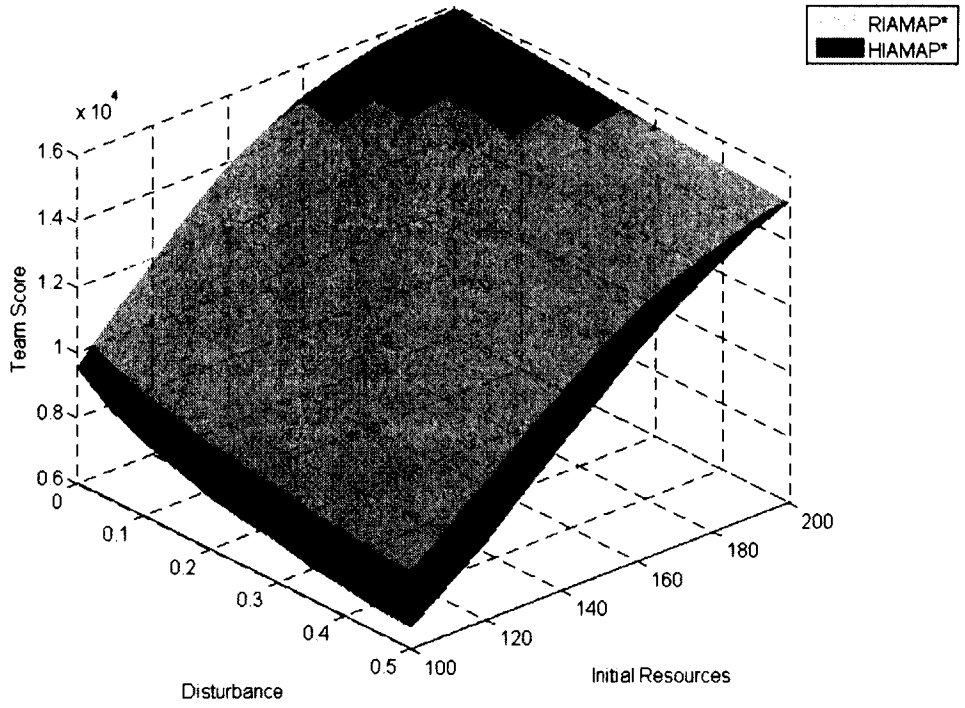


Figure 5.8: Team scores vs. Disturbance and Initial Resources

happens because the individual wellbeing of agents decline as disturbance grows, which hampers the proactive offering of help, but encourages the proactive requesting of help (Section 4.1). For lower disturbance, RIAMAP* still dominates when initial resources are low to moderate, because HIAMAP* is less effective in this situation. However, HIAMAP* dominates when disturbance level is low and initial resources are highly available; i.e. the situation in which the helper-initiated approach is highly effective.

Fig. 5.9 presents the simulation results of the second experiment in which we

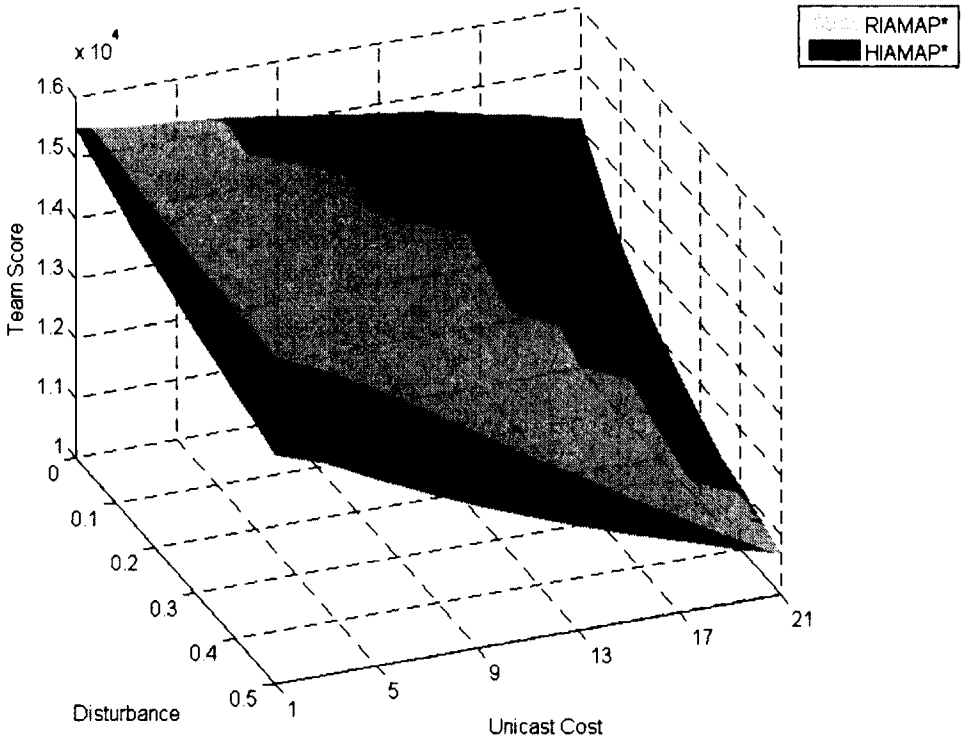


Figure 5.9: Team scores vs. Disturbance and Communication Cost

vary the level of disturbance, D, and unicast cost, U, when the initial resources coefficient, a', is set to 160. The comparative team scores show that the requester-initiated protocol, RIAMAP*, dominates when communication cost is low and also when disturbance is high. This is because the requester-initiated approach is more effective with low communication cost, and also high disturbance level has more negative impact on the helper-initiated approach. On the other side, the helper-initiated protocol, HIAMAP*, dominates when disturbance is low and communication cost is

high, as high communication cost has more negative impact on the requester-initiated approach.

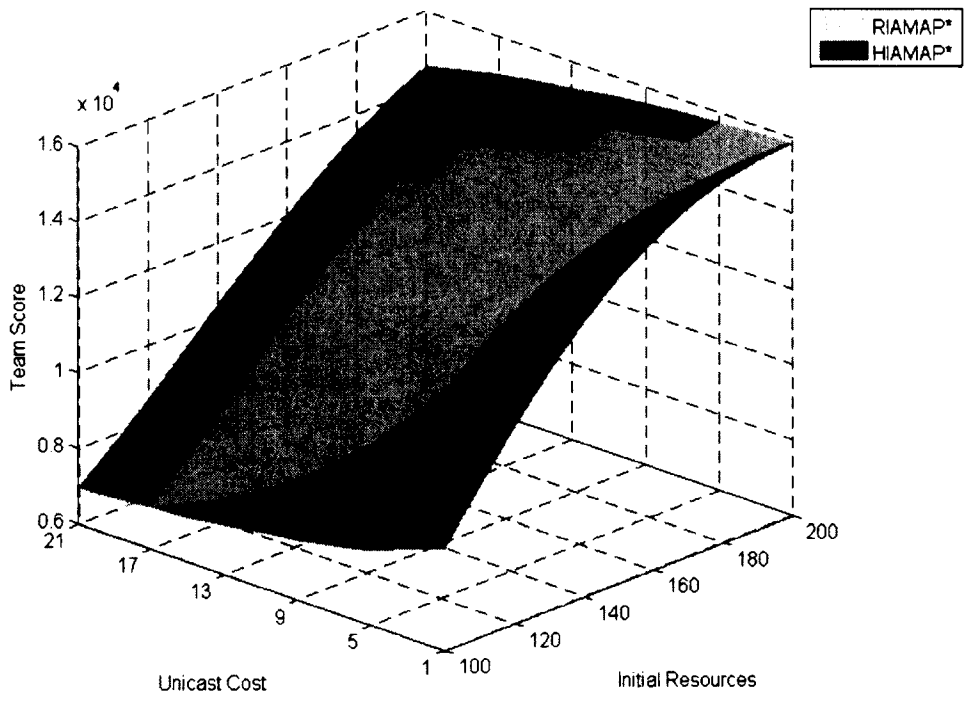


Figure 5.10: Team scores vs. Communication Cost and Initial Resources

In the third experiment, we vary the unicast cost, U, and initial resources coefficient, a', while the disturbance level, D, is set to 0.2. Fig. 5.10 presents the simulation results and again confirms the complementary performance profiles of the requester and helper-initiated approaches. As can be seen the requester-initiated protocol, RIAMAP*, dominates when communication cost is low and initial resources are low. On the other hand, the helper-initiated protocol, HIAMAP*, dominates when com-

munication cost is high and initial resources are high. This is again in agreement with our analysis of the requester and helper-initiated behavior in critical situations.

The simulation results in this section confirm our previous analysis of the relative strengths and weaknesses of each particular one-sided approach, and identify the critical situations where one of them dominates significantly. This analysis of their complementary performance profiles is incorporated in the BIAMAP design strategy, in which we leverage the strengths of the two different approaches in a combined protocol.

5.2.5 The Team Performance Impact of BIAMAP

In the following, we present three simulation experiments in order to examine the new combined protocol, the Bidirectionally Initiated Action MAP (BIAMAP), which is a composition of RIAMAP* and HIAMAP*. We compare three agent teams employing RIAMAP*, HIAMAP*, and BIAMAP, but otherwise identical and operating in identical environments. The simulation results show the impact of combining requester and helper-initiated approaches on the team performance. In each experiment, we vary two of the following three parameters: level of disturbance , D, initial resources coefficient, a', and the unicast cost, U; the third parameter is kept constant at the standard value selected in Section 5.2.2.

In the first simulation experiment, we explore the relative team performance for varying the disturbance level, D, and initial resources coefficient, a', while the unicast cost, U, is set to 9. Fig. 5.11, which is equivalent to Fig. 5.8 with addition of BIAMAP, shows the comparative team scores for the three teams. The immediate observation is that the team which employs BIAMAP outperforms the other two

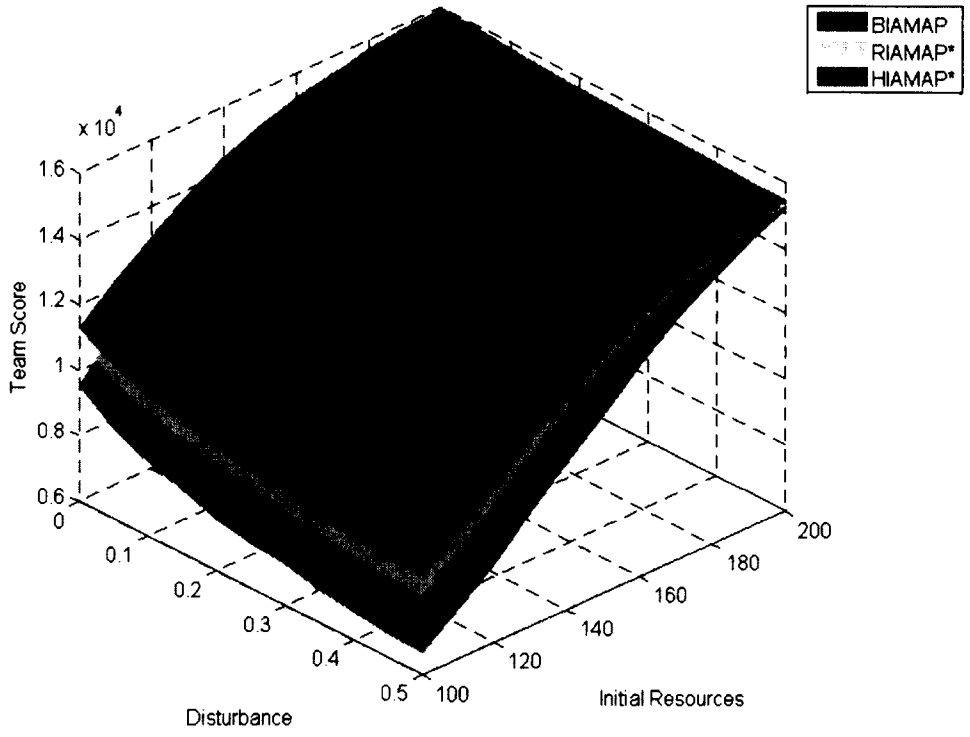


Figure 5.11: Team scores vs. Disturbance and Initial Resources

teams in most of the parameter space. This suggests the superiority of the combined protocol, BIAMAP, that allows initiative from both helper and requester sides. However, in the corner which corresponds to low disturbance and high initial resources, HIAMAP* still outperforms BIAMAP. This is because the helper-initiated approach is very effective in this situation, while the requester-initiated component of BIAMAP is not, and HIAMAP* prevails for being a simple one-sided protocol with no cost of unproductive requester-initiated behavior.

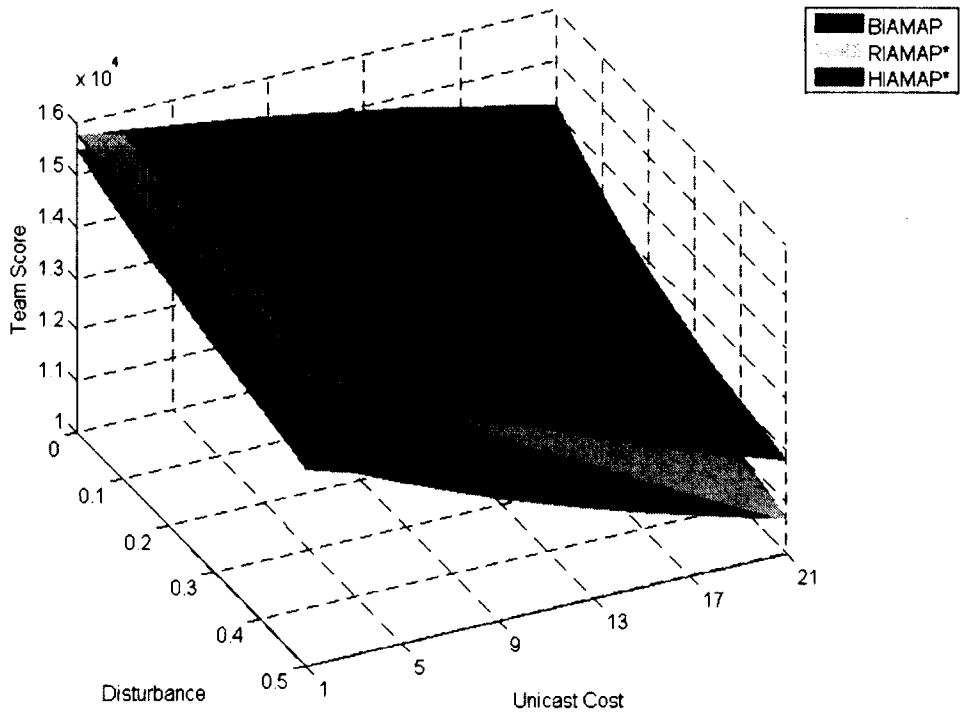


Figure 5.12: Team scores vs. Disturbance and Communication Cost

Fig. 5.12, which is equivalent to Fig. 5.9 with addition of BIAMAP, presents the simulation results of the second experiment in which we vary the the level of disturbance, D, together with the unicast cost, U, while the initial resources coefficient, a', is set to 160. The comparative team scores show that BIAMAP outperforms the two one-sided protocols in most of the parameter space. The exception arises when the communication cost is low, especially in the corner which corresponds to high disturbance level. This is because the requester-initiated approach is very effective in this situation, while the helper-initiated component of BIAMAP is not, and RIAMAP*

prevails for being a simple one-sided protocol with no cost of helper-initiated behavior.

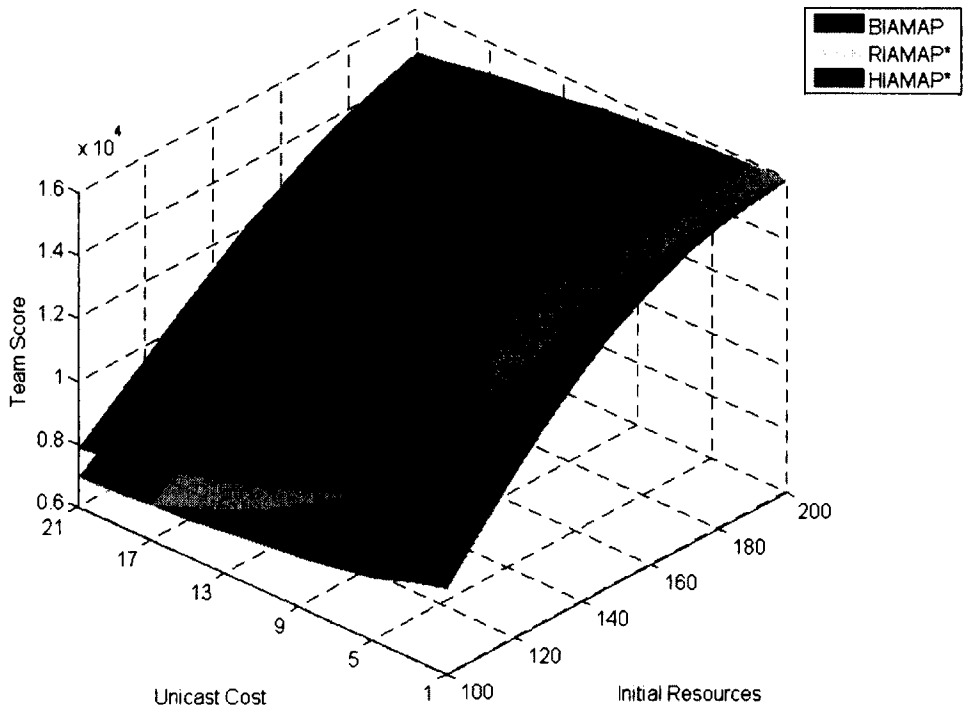


Figure 5.13: Team scores vs. Communication Cost and Initial Resources

In the third experiment, we vary the unicast cost, U, and initial resources coefficient, a', while the disturbance level, D, is set to 0.2. Fig. 5.13, which is equivalent to Fig. 5.10 with addition of BIAMAP, presents the simulation results and again shows the superiority of the combined protocol, BIAMAP, that allows initiative from both helper and requester sides. BIAMAP outperforms the two one-sided protocols in most of the parameter space, except in the situation when the communication cost

is low, especially in the corner which corresponds to low initial resources. This is again because the requester-initiated approach is very effective in this situation, while the helper-initiated component of BIAMAP is not, but still incurs the extra cost of helper-initiated behavior.

The simulation experiments presented in this section confirm that the overall performance of BIAMAP compared to RIAMAP* and HIAMAP* is superior. It outperforms both one-sided protocols in most of the parameter space, except in the critical situations where one of the one-sided approaches becomes unproductive, but BIAMAP still incurs the overhead costs of incorporating that approach. By optimizing the watermark parameters in Section 5.2.2, we have better tuned the balance between the proactive requesting and proactive offering behaviors, and we have better leveraged the advantages of the two different approaches in different situations. The experimental results in this chapter are slightly better than the ones in [Malek Akhlagh and Polajnar, 2014], because of the watermark optimizations.

Chapter 6

Conclusions and Future Work

This thesis presents a new agent interaction protocol, called the Bidirectionally Initiated Action MAP (BIAMAP), for incorporating direct help into agent teamwork. The new protocol is a new version of the Mutual Assistance Protocol (MAP), and inherits its principles and design features such as its agent team model, the local planning autonomy (LPA), and the bilateral distributed agreement (BDA). The novel features of BIAMAP allow an agent to initiate deliberation on direct help by either a request or an offer, and also to simultaneously provide and receive help. The impact of BIAMAP on team performance is investigated through simulation experiments with the same simulation models, microworld, and parameter space used in the previous studies of MAP. The experiment results suggest superiority of BIAMAP compared to existing MAP variations.

Building on previous research on help protocols, we have analyzed two different approaches to initiating deliberation on direct help, including their deliberation patterns and state-machine representations. In the first approach, an agent can proactively re-

quest help from its teammates; in the second, it can proactively offer help. We have first refined the bidding patterns of the Requester-Initiated Action MAP (RIAMAP) and Helper-Initiated Action MAP (HIAMAP) by realizing the possibility that the same agent can both provide and receive direct help simultaneously, which has resulted in their refined versions: RIAMAP* and HIAMAP*. Then, by analysis of the relative strengths and weaknesses of the requester and helper-initiated behaviors and realizing the possibility that a single protocol can incorporate both proactive requesting and proactive offering of help, we have designed the new combined protocol, BIAMAP, which is a composition of RIAMAP* and HIAMAP*, and leverages the advantages of the two one-sided approaches in different situations.

In our simulation models, we have represented the new interaction protocols in the particular state-machine formalism used in the Agent Interaction Modeling Simulator (AIMS) framework. It includes full specifications of states together with the respective transition rules, in a form that is easily translated into executable code in AIMS. We have investigated the impacts of employing RIAMAP*, HIAMAP* and BIAMAP on team performance through simulation experiments in AIMS, by varying the levels of parameters representing environment dynamism, agent resources, and communication costs.

The simulation results demonstrate the team performance gains for agent teams that employ the new protocols. RIAMAP* and HIAMAP* outperform the original one-sided protocols, RIAMAP and HIAMAP. respectively. BIAMAP outperforms all other MAP versions in most of the studied parameter space. This indicates that direct help in agent teamwork is the most effective when it can be both offered and requested within the same protocol, and also simultaneously provided and received by the same agent.

In the current version of BIAMAP, in the deliberation process on whether to perform the help transaction, an agent bids to an offer or a request only when the net team impact (NTI) is positive, i.e., when the team benefit within the receiver's subtask exceeds the team loss within the provider's subtask. Further studies can be established in order to incorporate the complementary saving feature introduced in the "frugal" version of RIAMAP, in which the help transaction may also occur when NTI is equal to zero but the help act saves resources at the team level.

The BIAMAP performance is superior in most of the parameter space except in the critical situations when one of the one-sided approaches becomes unproductive, while BIAMAP still incurs the overhead costs of incorporating that approach. This motivates us to introduce additional metrics into BIAMAP that may allow finer balancing of the one-sided behaviors in such critical situations. The possible design strategy requires an analysis of the potential benefits and costs of maintaining such metrics and verifying whether the benefits outweigh the costs.

For future work, we can also investigate the team performance impact of incorporating the Resource MAP, in which one or multiple potential helpers provide a part or all of the total required resources for performing a teammate's action, and compare it with the impact of Action MAP. We can further specify a combined approach which incorporates both Action and Resource MAP variations, and investigate its potential impact on the team performance in different scenarios.

We have investigated our help protocols in the context of a board game microworld, supported in the AIMS framework. In this study, we have simplified our modeling of the agent team, the subtasks, and the environment, in order to investigate the behavior of direct help in relatively pure form, without the impact of extraneous factors that may vary greatly across practical systems. When adapting our protocols to real-

world engineering applications, one may need to analyze the design requirements with respect to the observability of environment, asynchronicity of communications and actions, scalability issues of increasing the number of agents or the size of environment, time and deadlines for completing subtasks, etc. However, we expect to investigate these problems mainly in the context of specific classes of concrete applications.

Bibliography

Omid Alemi, Desanka Polajnar, Jernej Polajnar, and Denish Mumbaiwala. A simulation framework for design-oriented studies of interaction models in agent teamwork. In *Proceedings of the 2014 Agent-Directed Simulation Symposium (ADS'14)*, pages 28–35. Soc. for Computer Simulation International, April 2014.

Raphen Becker, Shlomo Zilberstein, Victor Lesser, and Claudia V. Goldman. Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research*, 22:423–455, 2004.

Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27:819–840, 2002.

Daniel S. Bernstein, Eric A. Hansen, and Shlomo Zilberstein. Bounded policy iteration for decentralized POMDPs. In *Proceedings of the 19th International Joint Conference on Artificial intelligence (IJCAI'05)*, pages 1287–1292, 2005.

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.

Michael E. Bratman, David J. Israel, and Martha E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355, 1988.

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.

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.

Behrooz Dalvandi. A model of emphathy for artficial agent teamwork. Master's thesis, University of Northern British Columbia, Canada, 2012.

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.

Xiaocong Fan, John Yen, and Richard A. Volz. A theoretical framework on proactive information exchange in agent teamwork. *Artificial Intelligence*, 169:23–97, 2005.

Foundation of Intelligent Physical Agents. FIPA specification part 2 - agent communication language. *Available at: www.fipa.org*, 1997.

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 9th National Conference an Artificial Intelligence*, pages 677–682, 1987.

Michael P. Georgeff and Anand S. Rao. The semantics of intention maintenance for rational agents. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, Montreal, Canada, August 1995.

Michael P. Georgeff and Anand S. Rao. A profile of the Australian Artificial Intelligence Institute. *IEEE Expert: Intelligent Systems and Their Applications*, 11(6): 89–92, December 1996.

Barbara J. Grosz and Sarit Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86:269–357, 1996.

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

Hideshi Itoh. Incentives to help in multi-agent situations. *Econometrica*, 59(3), 1991.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101: 99–134, 1998.

Ece Kamar and Barbara J. Grosz. Applying MDP approaches for estimating outcome of interaction in collaborative human-computer settings. In *Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM) Workshop*, Honolulu, Hawaii, May 2007.

Ece Kamar, Ya'akov Gal, and Barbara J. Grosz. Incorporating helpful behavior into collaborative planning. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'09)*, volume 2, pages 875–882, 2009.

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 8th 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.

Roger Mailler and Victor Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, volume 1, pages 438–445, 2004.

Mojtaba Malek Akhlagh and Jernej Polajnar. Distributed deliberation on direct help in agent teamwork. In *Proceedings of the 12th European Conference on Multi-Agent Systems (EUMAS 2014)*, Prague, Czech Republic, December 2014.

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 7th European Conference on Cognitive Ergonomics*, pages 367–377, 1994.

Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161:149–180, 2005.

Ranjit Nair, David Pynadath, Makoto Yokoo, Milind Tambe, and Stacy Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the 18th International Joint Conference on Artificial intelligence (IJCAI'03)*, pages 705–711, 2003.

Ranjit Nair, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI'05)*, volume 1, pages 133–139, Pittsburgh, PA, July 2005.

Narek Nalbandyan. A mutual assistance protocol for agent teamwork. Master's thesis, University of Northern British Columbia, Canada, 2011.

Narek Nalbandyan, Jernej Polajnar, Denish Mumbaiwala, Desanka Polajnar, and Omid Alemi. Requester vs. helper-initiated protocols for mutual assistance in agent teamwork. In *Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC'13)*, pages 2741–2746, Manchester, UK, October 2013.

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

Jernej Polajnar, Behrooz Dalvandi, and Desanka Polajnar. Does empathy between artificial agents improve agent teamwork? In *Proceedings of the 10th IEEE International Conference on Cognitive Informatics and Cognitive Computing*, Banff, Alberta, Canada, 2011.

Jernej Polajnar, Narek Nalbandyan, Omid Alemi, and Desanka Polajnar. An interaction protocol for mutual assistance in agent teamwork. In *Proceedings of the 6th International Conference on Complex, Interactive, and Software-Intensive Systems (CISIS 2012)*, pages 6–11, Palermo, Italy, July 2012.

Jernej Polajnar, Mojtaba Malek Akhlagh, Narek Nalbandyan, Denish Mumbaiwala, and Desanka Polajnar. Decentralized reactive adjustment of agent teamwork organization to changing environment. In *Proceedings of the 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC'14)*, pages 1457–1462, San Diego, California, October 2014.

Anand S. Rao and Michael P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1991.

Anand S. Rao and Michael P. Georgeff. A model-theoretic approach to the verification of situated reasoning systems. In *Proceedings of the 13th International Joint Conference on Artifical Intelligence*, volume 1, pages 318–324. Morgan Kaufmann Publishers Inc., 1993.

Anand S. Rao and Michael P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the 1st International Conference on Multiagent Systems*, 1995.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach.* Prentice Hall, 1995.

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

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.

Matthew E. Taylor, Manish Jain, Christopher Kiekintveld, Jun young Kwak, Rong Yang, Zhengyu Yin, and Milind Tambe. Two decades of multiagent teamwork research: Past, present, and future. *Collaborative Agents - Research and Development*, 6066:137–151, 2011.

Usman Wajid and Nikolay Mehandjiev. A survey of flexible agent interaction approaches. *Multiagent and Grid Systems*, 9:247–278, 2013.

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 13th 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.

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