PROACTIVE COMMUNICATION IN MULTI-AGENT TEAMWORK

A Dissertation

by

YU ZHANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2005

Major Subject: Computer Science

PROACTIVE COMMUNICATION IN MULTI-AGENT TEAMWORK

A Dissertation

by

YU ZHANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Richard A. Volz
Committee Members,	Wei Zhao
	Thomas R. Ioerger
	Yoonsuck Choe
	Amarnath Banerjee
Head of Department,	Valerie E. Taylor

December 2005

Major Subject: Computer Science

ABSTRACT

Proactive Communication in Multi-Agent Teamwork. (December 2005)

Yu Zhang, B.S.; M.S., Central South University, China

Chair of Advisory Committee: Dr. Richard A. Volz

Sharing common goals and acting cooperatively are critical issues in multi-agent teamwork. Traditionally, agents cooperate with each other by inferring others' actions implicitly or explicitly, based on established norms for behavior or on knowledge about the preferences or interests of others. This kind of cooperation either requires that agents share a large amount of knowledge about the teamwork, which is unrealistic in a distributed team, or requires high-frequency message exchange, which weakens teamwork efficiency, especially for a team that may involve human members.

In this research, we designed and developed a new approach called Proactive Communication, which helps to produce realistic behavior and interactions for multi-agent teamwork. We emphasize that multi-agent teamwork is governed by the same principles that underlie human cooperation. Psychological studies of human teamwork have shown that members of an effective team often anticipate the needs of other members and choose to assist them proactively. Human team members are also naturally capable of observing the environment and others so they can establish certain parameters for performing actions without communicating with others. Proactive Communication endows agents with observabilities and enables agents use them to track others' mental states. Additionally, Proactive Communication uses statistical

analysis of the information production and need of team members and uses these data to capture the complex, interdependent decision processes between information needer and provider. Since not all these data are known, we use their expected values with respect to a dynamic estimation of distributions.

The approach was evaluated by running several sets of experiments on a Multi-Agent Wumpus World application. The results showed that endowing agents with observability decreased communication load as well as enhanced team performance. The results also showed that with the support of dynamic distributions, estimation, and decision-theoretic modeling, teamwork efficiency were improved.

DEDICATION

To my parents, Baojun and Peimei, who taught me that science is a question.

To my husband, Weihua Guan, who taught me that faith is a necessity.

ACKNOWLEDGMENTS

Every time I read the acknowledgment sections of dissertations I came across, I always thought about what I would be writing when it came to my own section. During the interviews of my academic job hunting, the most frequently asked question is how I will teach students. Plutarch once said “The mind is not a vessel to be filled, but a fire to be kindled.” I am the firm believer in this and think the primary role of a teacher is to kindle a fire in the mind of each student. I am very fortunate to have been offered the opportunity to work with such good teachers. They not only fired my eager desire for boundless knowledge but also showed me the short way by which I can reach the part which is most interesting to me.

First and foremost among these teachers is my advisor, Dr. Richard Volz. His devotion to his students, his vision and wisdom to help them avoid detours, his care in teaching them science, rather than merely getting them through the program, his taking great care in providing them what they need to do their research, and his ethical standards combine to provide the best advisor I could ask for. Never did Dr. Volz show anything but patience in responding to any of my questions (some were simple or even silly). Rather than directly saying yes or no, his answers always convinced me that everything needs more thought.

My committee members deserve many thanks for their advice and friendliness. Dr. Wei Zhao made himself available to provide help and advice, and made sure that I saw the big picture when it came to my research. Dr. Tom Ioerger has supplied me with many useful resources and comments. He helped me bring different kinds of

considerations to the research to make it more comprehensive. Dr. Yoonsuck Choe and Dr. Amarnath Banerjee were helpful in pointing out limits of the approach I presented, and deserve much credit for making me think about the research from completely fresh perspectives.

I also would like to thank Dr. Dianxiang Xu, who worked with me on the journal paper and provided many valuable comments about my research. Dr. John Yen, Michael Miller, Sen Cao, Linli He, Maitreyi Nanjanath, Jonathan Whetzel, Jesse Plymale, and Norman Ma, all provided help and made this serious research process full of fun.

My family has been a source of strength and wisdom during my Ph.D. study. For more than 5 years, I have not visited my parents, but they delivered their care and encouragement to me at any chance they found. The help I got from Weihua, my husband, is priceless. My greatest motivation has always been to be worthy of his love.

Finally, this research was supported in part by DoD MURI grant F49620-00-I-326 administered through AFOSR.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

1.1. Motivation

An *agent* is defined as a mapping from perceptions to actions [102]. It can be achieved via hardware (e.g. robotics) or software systems. The agent resides in the environment, behaves autonomously, purposively, and flexibly; it may have sensing, adaptive, social, and emotional capabilities [127]. The capabilities of a single agent are limited by its knowledge, its computing resources, and its perspective. Particularly, when interdependent problems arise, agents in the system must coordinate with one another to ensure that interdependent problems are properly managed. Thus, they form *multi-agent systems*. In a multi-agent system, multiple agents that cooperate towards the achievement of a joint goal are viewed as a *team*. *Teamwork* is a cooperative effort by a team of agents to achieve a joint goal [121].

Sharing common goals and acting cooperatively are critical issues in multi-agent teamwork [1, 13, 21, 90]. To date, control paradigms for cooperative teamwork have allowed agents to communicate about their intentions, plans, and the relationships between them [65, 97, 114, 115, 116, 119, 121]. Using communication, team members can share common goals and coordinate their actions by distributing valuable teamwork-related information. In order to do so, each of the team members should track the activities of the others, reason about possible conflicts or constraints, establish

---

This dissertation follows the style of Artificial Intelligence.

certain parameters for performing joint actions, and provide or ask for any information that is needed to perform tasks. Existing solutions for the communication problem have four major disadvantages:

- Agents share a large amount of knowledge about the teamwork, which is unrealistic in a distributed team.
- Communication interactions are hard-coded in teamwork processes, which is not universal.
- Cooperation processes involve high-frequency message exchange, which weakens teamwork efficiency.
- Current solutions ignore communication risk, which is one of the most important factors of agent decision-making.

Moreover, some researchers have found that communication, while a useful paradigm, is expensive relative to local computation [2]. Therefore effective, universal, and practical communication mechanisms are needed for helping agents produce effective and realistic behaviors and interactions in teamwork.

1.2. Investigating Effective Communication

Most of the literature (see Section 2.2) reports on technologies empowering agents from outside, such as teaching them to obey social conventions [111]. We investigate this problem by bringing agent initiatives into play.

We investigate effective human team cooperation and incorporate the findings into multi-agent teamwork. Humans are naturally capable of observing the environment and others so they can establish certain parameters for performing actions

without communicating with others. A shared mental model [100, 118], one of the major psychological underpinnings of teamwork, enables an effective human team to anticipate information needs of teammates and offer the information [135]. We call this ability *proactivity*. Therefore, effective team cooperation can be achieved, if agents are able to observe the environment and each other, predict needs for teamwork-related information and distribute such information proactively.

1.2.1. Observability

Observability is the ability to observe the environment and other agents, and from it, make inferences about them. Although it has gained some attention [95, 68, 7, 59], observability in multi-agent teamwork has not been explored deeply. We argue that the reasons for this might be threefold. First, from the team point of view, observability is a capability of an individual agent, rather than of a whole team. It is difficult to abstract a team's observability based on every team member's individual observability. Second, since belief reasoning is theoretically intractable [47], the process of an agent using its observability to reason about teammates' beliefs becomes highly complex. Third, agents lack an effective way to reason about others' observabilities. However, in a dynamic, distributed teamwork environment, apart from prior knowledge such as the team goal, observability is a major means for an individual agent to obtain information. An agent with observability may produce effective communication by observing the environment and its teammates and then estimating their beliefs without generating unnecessary messages.

1.2.2. Proactivity

Proactivity is the ability to take initiative by exhibiting goal-directed behavior [127]. Intelligent-agent researchers maintain that proactivity is one of the hallmarks of agency [127]. Agents with proactivity can respond to external stimuli in a timely way, and they can also prepare knowingly for some unexpected future [135]. Hence the ability to anticipate the information needs of teammates and assist them proactively is highly desirable. While an agent can anticipate certain information needs of teammates, it may not always be able to predict all of their needs, especially if the team interacts with a dynamic environment. Therefore, when an agent needs some information, it is also necessary to anticipate the information production of teammates and ask for the information actively. Hence, proactivity allows agents to proactively tell others about a piece of information when producing it or to actively ask for a piece of information when needing it. Proactivity may produce effective communication in three ways. First, messages will be conveyed to agents when they need an information item, rather than sending all information to them. Second, proactive tell can partially eliminate the need to ask. Third, if there is no proactive tell, active ask may eliminate multiple requests for information, i.e., only ask one time per need.

1.2.3. Challenges

The challenges in achieving effective communication in an agent team exist in three aspects. First, the distributed nature of an agent team and the dynamic nature of the world often make it infeasible for an agent to have complete and up-to-date information about other teammates and the world. The resultant uncertainty may

seriously affect quality of communication among agents. Furthermore, agents have different capabilities for solving the problems, such as different observabilities, which lead to different abilities for obtaining information. This increases the difficulty of deducing what others know and consequently what they need. Third, agents are distributed in the world, so they do not realize each time at which a piece of information is produced or needed by others. Therefore delivering a tell or ask to others at the proper time becomes critical for a team.

1.3. Our Approach and Its Contributions

The goal of this research is to devise effective communication mechanisms, enabling agent initiatives and dynamic cooperation in multi-agent teamwork. We design and develop a new model called Proactive Communication, for supporting realistic behavior and interactions in complex and dynamic domains. The central thesis of this research is that

> *Proactive Communication captures and represents the complex, interdependent communication decision-making processes among agents, and achieves effective communication by giving agents the capabilities of observability and proactivity. Observability helps agents to monitor the environment and track teammates' mental states. Proactivity allows agents to act in anticipation of future information productions or needs to tell or ask each other about teamwork-related information.*

Our approach endows agents with observability and proactivity via three distinct but closely related perspectives:

- Observation-Based Proactive Communication (OBPC). It allows agents to use their observabilities to track others' mental states as well as decreasing the communication load. Different from other observation approaches, agents can observe not only the environment, but also actions of others, and use this knowledge to decide which information might be known by others and therefore does not need to be exchanged.
- Dynamic Information Prediction (DIP). It is a dynamic estimation of the probability distributions of information productions or needs and the use of these data to capture the complex internal processes of decision-making regarding communication. The major feature distinguishing this approach is that agents take advantage of their historical knowledge to estimate the distributions of the information need and production times.
- Decision-Theoretic Proactive Communication (DTPC). It is a decision-theoretic determination of communication strategies. During multi-agent teamwork, agents should be able to deal with uncertainties, since they may only have incomplete information about the teamwork, the environment, and the potential value and cost of information delivery. One way to deal with this problem is a decision-theoretical approach [25]. Broadly speaking, the decision theory is a means of analyzing a series of strategies in order to decide which should be taken, when it is

> uncertain exactly what the result of taking the strategy will be [92]. However, departing from the traditional decision-theoretic approach, DTPC emphasizes communication benefiting the team and focuses on decision interactions between needer and provider, i.e., their decisions are interdependent, so they must consider how their counterpart's decisions impact their own.

The major novel contribution of this research is the concentration on interactions between agents and the emphasis of relations connecting them. This feature makes communication benefit the team as well as enhancing agents' ability to take initiatives. Specifically, 1) we use agents' observabilities to track team members' mental states, so they can infer what the others know and when and therefore can decrease the communication load; 2) we introduce an idea of dynamic information prediction, which allows agents to anticipate coming information production or need time based on historical knowledge; 3) we introduce an model that agents estimate others decisions in the decision-theoretic communication, which empowers agents to deal with communication interdependencies in team cooperation.

1.4. Structure of the Dissertation

This section introduces the motivation of this research and the overall idea of our approaches and contributions. The rest of this dissertation is organized as the following:

- In Chapter II, we review related work to this research.

- In Chapter III, we represent system architecture and agent execution cycle, and give an overview to Proactive Communication and its three components: OBPC, DIP and DTPC.
- In Chapter IV, we focus on OBPC. We 1) define syntax and semantics to observability, and 2) develop algorithms for using observability to decrease communication load.
- In Chapter V, we focus on DIP. We 1) introduce a statistical approximation of distributions of information production and need, and 2) introduce the data acquisition process for performing the approximation.
- In Chapter VI, we focus on DTPC. We 1) define communication policies, which can be used by agents in different communication situations, and time points relevant to information production and need, 2) define utility function which is used to evaluate each policy, 3) introduce agent decision making processes and 4) develop algorithms for decision-theoretic proactive communication.
- In Chapter VII, we 1) introduce criteria of applicable domain for Proactive Communication, 2) design an application domain, 3) design and analyze two sets of experiments we have run in the domain.
- Finally, in Chapter VIII, we conclude this dissertation and discuss some future work.

CHAPTER II

RELATED WORK

This research is about communication in multi-agent teamwork and is built on several areas of previous research. In this section, we review the literature of these areas and the progress of the work at Texas A&M University (TAMU) that is the foundation for this research, including teamwork theories and teamwork systems (see Section 2.1); decision making models in agent research (see Section 2.2), decision-theoretic modeling communication (see Section 2.3), other effective communication approaches (see Section 2.4), observability and belief maintenance (see Section 2.5), problem-specific prediction (see Section 2.6), psychological study of shared mental models (see Section 2.7), and context of work at TAMU (see Section 2.8).

2.1. Teamwork as Represented by Artificial Intelligence

Recently, researchers have been interested in building teamwork in distributed and dynamic domains, where each autonomous team member works cooperatively to solve a part of a problem in parallel. However, a team of agents is more flexible and efficient than a group of single agents only when a flexible and efficient means of coordinating the agents exists. In many ways, the teamwork problem is similar to that of parallel computing: doubling the number of processors used in a computation usually will not double the speed with which the solution is found. The extra processing power does not become an advantage until a sophisticated means of cooperative processing is found. This challenge inspires many teamwork theories, such

as joint intention [20, 81], shared plan [44, 46], commitments and conventions [63, 64], and planned team activity [70]. This section introduces these theories, followed by three examples of teamwork system implementations.

2.1.1. Teamwork Theories

2.1.1.1. Joint Intention

Joint intention is one of the most important teamwork theories [20, 80, 81]. It was developed based on individual intention, which is a logical formalization called persistent goals [19]. Cohen and Levesque derived an operator, PGOAL, which describes how an agent's intentions are related to its beliefs, commitments, and actions. An agent A has a persistent goal G, if all of the following are true: 1) A wants G to be true at some point in the future; 2) A believes that G is not yet true; 3) A believes that either G will be true or G will be impossible before it abandons its goal.

PGOALS are used to define intentions, in the form of the primitives INTEND1 and INTEND2. INTEND1 is defined as the persistent goal to perform a particular action by an agent. In other words, intending to take an action is a kind of persistent goal. Thus, intentions are future-directed. This is a near-approximation to present-directed intention: the agent desires to have done an action immediately after believing that it was about to do it, i.e. intentions are directed towards something happening next. However, because INTEND1 is a commitment to perform a particular action, it does not handle the case where the agent does not know what action it needs to perform to bring about the goal. INTEND2 is defined as the persistent goal to have done some

actions to bring about the goal (this means the agent has a plan), and the agent would not select these actions if they are thought not to lead to the goal.

Cohen and Levesque use their theory of persistent goals to build a theory of joint intentions. Joint intentions are intended to clarify the relationships among belief, desires, and intentions (BDI) for multiple agents [9, 10, 96]. Joint intentions are developed on three levels. First, they define weak goals, which specify the conditions under which an agent holds a goal, and the actions it must take if the goal is satisfied or impossible. Second, they define joint persistent goals for multiple agents. Finally, they define joint intentions in terms of weak goals and joint persistent goals. Joint intentions are attractive because they are presented in an implementable framework. For example, Jennings developed an implementation of joint intentions for industrial robots [65].

Joint intention theory imposes a strong "observant and proactive" requirement. It uses mutual beliefs to form joint intentions. An agent who personally comes to believe that a joint goal is either achieved, unachievable, or irrelevant, must commit to let all other team members mutually believe that this is the case. While mutual belief, being an infinite recursion about other agents' beliefs, is undecidable in theory [47], a computational approximation is required in practice [65, 97]. Thus, issues of observation, prediction and proactive communication are raised for practical implementations.

2.1.1.2. Shared Plan

Shared plan [44, 46] is another important teamwork theory. It was developed to deal with collaborative activities among human-agent mixed teams. It considers team

collaboration not as a group of single agents patched together, but as an integrated system that needs to be designed from beginning to end [43, 108]. As opposed to joint intention, shared plan does not have joint conceptions. It assumes that each agent has its own mental state (including intentions, capabilities, and commitments) and shares it with others. The formal representation of these aspects of the mental states of team members is called a Shared Plan. The shared part can guarantee teamwork, such as two agents working together to perform an action.

Grosz and Kraus propose five types of plans: FIP for full individual plans which means an individual agent has a full recipe for doing an action, PIP for partial individual plans which means an individual agent only has partial knowledge of doing an action, FSP for full shared Plans which means a group of agents has complete recipe of some group activity, PSP for partial shared plans which means a group of agents only has partial recipe of some group activity, and SP for shared plans which means a group of agents has a certain level of belief in their abilities to perform group actions [44]. The definitions of FSP and PSP only explicitly state some of the requisite knowledge, others which are implicit produced from agents' interactions form SP.

These plans are defined in terms of beliefs and intentions in agents' mental states. At the beginning, agents have only partial plans (individual or shared). By reasoning individually, communicating with others, or observing the environment, these partial plans are completed. In the special case where an agent finds that it cannot perform an action, the whole group will revise its procedures.

The evolving process mentioned above obviously requires agents' observation and communication. Shared Plan theory touches the proactive communication problem in other ways. First, as in joint intention, the theory requires that a group of agents must have a mutual belief of a partial procedure in order to have a collaborative plan for an action. Second, in several places the theory proposes that in collaborative activities, participants not only do means-ends reasoning about their own mental states and actions, they also reason about others' mental states to support others' actions better. For example, the term Int.To (intending to) presents an agent's intention to do an action while the term Int.Th (intending that) presents an agent's intention that some propositions hold true [44]. Thus, Int.Th concerns how others' intentions affect the agent's intention. Third, the theory requires agents to know that their teammates are capable of carrying out their actions. Grosz [44] notes that agents must communicate enough about their plans to convince teammates of their capabilities to carry out actions. If agents can predict this requirement and tell it proactively, the process can be simplified. Fourth, Grosz, in axiom A1, [44] points out that an agent cannot knowingly hold conflicting intentions. Note that the axiom is not valid if the agent is unaware of the conflict. Since some of these intentions involve others' mental states, the requirements for observation and proactive communication are the same to avoid such conflicts.

2.1.1.3. Commitments and Conventions

Jennings emphasizes that coordination is a key property that guarantees better multi-agent team performance [63, 64]. Without coordination, a multi-agent system can

become a collection of incohesive individuals. He developed a model of coordination, whose two central concepts are (joint) commitment and (social) convention. Jennings views a commitment as a promise to take a certain action, and conventions as rules for monitoring these commitments. He argues that "all coordination mechanisms can ultimately be reduced to joint commitments and their associated social conventions" [63, 64].

Properties of commitments and conventions can be found in numerous sources [9, 4, 8, 27, 36, 107, 109]. Commitments and conventions have been adopted widely in solving multi-agent cooperation problems. In agent-oriented programming (AOP), Shoham treats commitments as obligations of agents and uses commitment rules to decide their actions [112]. The BDI model [10] uses commitments to direct an agent's actions and planning. In Reusable Task Structure-based Intelligent Network Agents (RETSINA), [119] devises a complex negotiation protocol to force agents to agree on their commitments and then to perform socially complex actions. In their collaborative agent system (COLLAGEN), Rich and Sidner provide a set of conventions based on principles underlying human collaboration for collaborative discourse between humans and agents [97].

Jennings emphasizes that conventions are used to decide what information needs to be tracked about agents, and how to track them. For instance, a convention may require an agent to report to its teammates any changes it detects with respect to the attainability of the team goal. This need to report raises the requirement of analysis of communication needs before agents communicate with each other. Jennings also

gives an example of specific conventions for high- and low-bandwidth situations, in which some knowledge is not communicated to all agents if the bandwidth is not available. This raises the issue of the need of effective communication [63, 64]. Jennings does not explore deeply such problems as how conventions are selected or what the tradeoffs and guarantees associated with the selection of particular conventions are.

Our work provides a solution to effective communication. In our work, agents use observation to deduce the amount of communication needed. Meanwhile, agents can predict future information production and need by analyzing historical information records. DTPC is a sophisticated process that guarantees agents choose the right rules (conventions) by which to communicate.

2.1.1.4. Planned Team Activity

A group of Australian researchers propose planned team activity in the logical and practical design of rational agents cooperating in a team [70]. They suggest that joint plans (common to all team members) that specify the means of satisfying joint goals are supplied in advance, rather than being generated by the agents. Their argument is that the agents embedded in a dynamic environment can respond rapidly to important events by adopting applicable plans. The joint plans are represented by concepts of team skills and team members' roles. These plans usually will be qualified by preconditions that specify under what circumstances they are applicable. The plan execution for each agent consists of the selection and hierarchical expansion of these plans.

To achieve the planned team activity, common knowledge necessary for coordination and synchronization of agents' activities is imposed on the agents. The common knowledge that includes mutual beliefs about the world and about each other's actions places strong requirements upon agents' observation. Kinny et al. propose that the common knowledge can be achieved alternatively by communication between agents. This approach implies the need of effective communication. The assumption that the plans of individual agents are known at compile time might enhance the team's proactivity by the possibility of reasoning in advance about which team members potentially can achieve certain goals.

2.1.2. Teamwork Systems

2.1.2.1. STEAM

STEAM (Shell for TEAMwork) is a teamwork system built on joint-intention theory. STEAM addresses two important issues of joint-intention theory: 1) There is no practical method given for forming joint intentions; 2) A single agent's defection automatically causes the failure of the entire group task. Tambe describes methods for solving these two problems [121]. He solves the first problem, that of forming joint intentions through communication. He frames the solution in terms of joint intention itself. In order to synchronize a joint intention, the cooperating agents form weak achievement goals to bring others into their plan. Agents who have accepted the joint goal as their own form weak achievement goals. The same mechanisms that enforce communication when plans break down under joint intentions are used here to ensure synchronization when attempting to form a plan.

The second problem, what to do in case an individual agent defects from the group, is more interesting. This case necessitates replanning—allowing a single defector to cause the failure of the group as a whole is clearly unacceptable in robust systems. To implement the replanning process, Tambe created a set of role-monitoring constraints, which describe each agent's importance to the plan as a whole. The constraints describe cases where one of the following apply: 1) AND-combination, the actions of each member of a group of agents are vital to the achievement of the goal; 2) OR-combination, the actions of any one of a set of agents would suffice to achieve the goal; 3) Role dependency, one agent's actions depend on another agent's actions, such that without the second agent, the first can not complete its role. When an agent defects from a group action, the remaining agents invoke a *repair* action. Each examines the dependency structure to see whether the remaining group can complete the plan; if so, they continue. If the failure was in fact the result of a single agent's defection (Tambe calls this situation a critical role failure), the agents reorganize and carry on with new roles. If there is no possible reorganization that can complete the goal, then the goal fails.

Another interesting feature of STEAM is selective communication, where agents communicate only information with high utility to the completion of the plan. Tambe and Rosenbloom proposed that agent-monitoring is a key capability required for intelligent interaction [120]. Selective communication involves monitoring other agents’s observable actions and inferring their high-level goals, plans, and behaviors. Communication is generated based on monitoring and reasoning about the cost of

communication in deciding whether to communicate or not. However, Tambe's work focuses on establishing the joint intentions of team members in trying to achieve a joint goal, but not on analyzing the information needs among team members in order to provide information proactively, which is our focus. The differences between selective communication and our approach are examined in greater detail Section 2.3.1.

The reliance on communication among agents in joint intention theory means that the possible domains are limited, however. Joint-intention theory can not be used when agents are unable to signal each other or to cooperate with agents that were not designed with joint intention in mind. We want a system that can cooperate with agents in general, not just those that were designed to cooperate with the present system, another reason we use observation as well as communication for inter-agent cooperation, thus agents do not need to communicate with each other about the information which they can be seen by themselves or which they believe can see by others.

*2.1.2.2. GRATE**

GRATE* is an extended version of GRATE (Generic Rules and Agent model Test-bed Environment) [63, 64]. In GRATE*, joint responsibility is built on joint intention. GRATE* specifies that preconditions must be attained before collaboration can guarantee that individuals behave together either when joint activity is progressing satisfactorily or when it runs into difficulty. Like STEAM, it also requires agents to agree on the team plans that are to be executed.

However, several simplifying assumptions used to approximate a formal description of joint responsibility deprive GRATE* of scalability for dealing with more general systems. First, GRATE* is used in industrial settings in which foolproof communications can be assumed [65], and thus communication is the only way to track agents. By comparison, we track agents by both communication and observation. We also use observation and decision-theoretic proactive communication to decrease the amount of communication. Second, Jennings supposes that agents are able to predict, with a reasonable degree of accuracy, the time it will take to execute each of their domain-level activities. In order to do so, each action recipe presents its starting time and duration of the action. We argue that since agents are in a dynamic environment, the starting time and duration of an action vary with a number of uncertain elements, such as when an action's precondition is attained. Thus, we need a prediction of communication needs associated with preconditions and effects of an action, rather than fixed action starting time and duration in action presentation. Third, GRATE* maintains knowledge about other agents through acquaintances models, which are used to keep track of what teammates' capabilities are. However, the question of how much knowledge should be used in these models is left unaddressed. In contrast, we use observation to track teammates' mental states, in order to reason about what they can see and what they can infer from what they can see.

2.1.2.3. COLLAGEN

COLLAGEN (COLLaborative AGENt) [97] is a collaborative human user-interface system that is built on shared-plan theory. In COLLAGEN, communication is

assumed to be reliable. However, from a human-usability perspective, limiting the number of communications is still desirable. To address this issue, recent empirical work by Lesh, Sidner and Rich [79] utilizes plan-recognition in COLLAGEN. The focus of that work is on using the collaborative setting to make plan-recognition tractable. For instance, ambiguities in plan-recognition may be resolved by asking the user for clarification.

COLLAGEN includes observation as one kind of communication (another kind is discourse) and assumes all agents' and users' actions are mutually observable through a directed-manipulation graphical interface. We separate observation and communication because observation involves a complex belief-maintenance process and hence is the basis of communication decisions. Work on COLLAGEN did not investigate how much knowledge must be maintained for effective collaborative dialogue with the user. In contrast, we are able to provide such knowledge by analyzing team plans, i.e., the preconditions and effects of plans. Furthermore, analyzing the dialogue plans for risk points may allow systems such as COLLAGEN to decide whether to use communication for clarification, regardless of plan-recognition ambiguity.

2.2. Decision Making Models in Agent Research

Researchers in psychology, cognitive science and computer science have generated a variety of computational models of decision making, oriented toward understanding and modeling behaviors of human decision making, under situations with risky, time pressure, high stakes and dynamic uncertainty settings [142, 72, 71,

86, 143, 144, 138, 18]. Perhaps the most familiar fields for AI are two: classic decision theory and naturalistic decision-making. We introduce each of these models.

2.2.1. Classic Decision Theory

Classic Decision theory [93] is used to select an optimal action. It generally includes four areas: 1) decision theory, 2) Bayesian probability theory, 3) Markov decision process and 4) game theory.

2.2.1.1. Decision Theory

The assumption underlying decision theory is rationality, i.e. the decision maker won't intentionally select an action that is inferior to some other actions. The theory requires that the decision maker specifies a set of possible actions, a complete and mutually exclusive set of uncertain states, and a set of evaluative dimensions. The decision maker then assesses the utility of each action based on the probability of each uncertain state and the weight of each evaluative dimension. The theory enables decision makers to calculate a utility reflecting the overall desirability of each action. With the general decision-making procedure, the decision theoretic models vary. For example, the basic model, maximization of expected utility (MEU) [102] consists of summarizing the value of each potential outcome multiplied by the probability that the outcome would in fact be obtained. This product sum is then compared with the expected values for the other actions. The action that has the largest expected value is the one that should be selected, called MEU. In the work [142], it summarizes other 14 types of decision theoretic models, including Maximization of Subject Expected Utility (MSEU) which is the same model as MEU except that utility is substituted for dollar

value, Lexicographic (LEX) model which assumes that each option has attributes that will promote valued outcomes, etc.. It can be seen that in addition to the assumption about rationality described above, each of these decision theoretic models assumes that the decision maker has preferences for the outcomes and that these preferences can be measured, e.g. by a utility function.

Our decision-making is based on the same rationality consideration, thus the decision maker will choose a communication policy with the maximum utility. We handle the unknown random variables in the utility function by utilizing the Empirical Distribution Function to estimate their probabilities.

2.2.1.2. Bayesian Probability Theory

Bayesian probability theory [92, 93] is used to draw inferences about situations. It requires that decision maker to identity a set of states (e.g. weather condition of Houston when the decision maker is in College Station). Then for each pair of states the decision maker can establish whether the pair is independent or not. The decision maker then can build a graph in which each node for a state and an arc points from state A to state B if the latter depends on the former. The resulting graph is known as a Bayesian network [93]. The Bayesian network provides a computational framework to calculate the probabilities of preferences to the decision maker. The next steps are to assess the probability that each hypothesis is true, identify all the potential observations that might bear on those hypotheses in the future, and quantify the impact of each such observation. Then as new observations occur, decision makers can use algorithms from the theory, such as Bayesian rule, to updating probabilities in the hypotheses. The

decision making method in this theory is the same with that of the decision theory. For each state, we calculate a utility for each state and we will prefer the state with the highest MEU.

2.2.1.3. Markov Decision Processes (MDPs)

In essence an MDP is an iterative set of classical decision problems [102]. As the Bayesian network, MDP is also represented by a graph in which one node denotes a state. Performing an action in that state will result in a transition to one of a number of states each connected to the first state, with some probability, and incurs some cost. After a series of transitions a goal state may be reached, and the sequence of actions executed to do this is known as a policy. Solving an MDP means to find a minimal cost policy for moving from some initial state to a goal state [88]. A big problem of MDPs is that, it unrealistically assumes that agent knows at every point what state it is in. This means that it is possible to measure some aspect of the world and from this measurement the agent can tell precisely what state the world is in. In reality, it is more likely that from the measurement something there is still uncertain in the world. In such case, the states of an MDP are replaced by the agent's beliefs about those states, and we have a partially observable MDP (POMDP) [88]. Because POMDP can capture so many real problems, it is currently a hot topic in agent research, despite the fact that they are intractable for all but the smallest problems.

Our problem also deals with partially observable environment. However, different from POMDP, our agents make decisions based on not only data collected from history, but also estimations of a sequence of future communication interactions

between information providers and needers. Different sequences will lead to different values of communication policies to be chosen by the agents and the agents will choose a best one with the maximum utility (meaning the minimum cost). To handle the intractable problem occurring in decision-making processes, we use some approximation to balance the quality of decisions with the complexity of computation. While this is not exactly precise, it is shown to be a practical solution in our experiments.

2.2.1.4. Game Theory

Game theory is a branch of economics that studies interactions between self-interested agents. Perhaps the most compelling area that the game theory has been applied on multi-agent systems is negotiation [76, 104, 48]. Game theoretic studies of rational choice in multi-agent systems typically assumed that agents were allowed to select the best strategy from the space of all possible strategies, by considering all possible interactions. The search space of strategies and interactions that needs to be considered has exponential growth, which means that the problem of finding an optimal strategy is in general computationally intractable. The study of finding efficient (polynomial time) algorithms for intractable problems in multi-agent negotiations is an ongoing area of work [92].

In our system, agents share a common goal, and would therefore be willing to assistant others. Therefore, decision-making strategies are different from that of the game theory that contains self-interested agents. For example, in a team, agents help each other to maximize utility for the whole team; while in a game, every agent acts in

the manner maximizing its own utility but minimizing utility for competitors. Many human teams (including ours) involving joint decision-making as information gathering and task allocations have the computation complexity problem. Again, we use some approximations to balance the quality of decisions with the complexity of computation.

2.2.2. Naturalistic Decision Making

Naturalistic decision directly relates to the way experienced people actually make decisions in natural settings [142]. Comparing to the decision theory that is about selecting an option, the naturalistic decision focuses on diagnostic decision-making (situation assessment). Among many models developed under this field, recognition-primed decision (RPD) model [72] is the most common one. The RPD model emphasizes the recognition of situational dynamics as one of the key drivers in selecting an action. The RPD model describes how decision makers can rely on their experience to recognize situations and identify viable courses of action without comparing the relative benefits or liabilities of multiple actions. The decision maker first identifies the situation as familiar or typical. This recognition enables the decision maker to know several important things, such as which goals to take, what to expect and which actions typically work. The RPD model focuses attention on the importance of situation awareness for successful decision making in field settings. For example, [33] uses RPD to support human teams to make faster and better decisions when there is not time for extensive reasoning.

RPD model requires the decision maker to have enough experience to assess novel and dynamic environment it encounters. Also the process which enable experienced decision maker to develop their situation awareness requires the designer have completely understanding to domains; this tends to discourage the generic problem solving across different domains. Therefore RPD has wide applications in military training tasks, such as training with Tactical Decision Making Under Stress (TADMUS) [18].

2.3. Decision-Theoretic Modeling Communication

Though growing up long before the concept of an intelligent agent was conceived, decision-theoretic modeling has gained increasing interest as a technique for communication in multi-agent systems.

2.3.1. Selective Communication

STEAM uses selective communication, by which agents communicate only information with high utility [121]. The decision depends not only on the cost and benefit of the communication, but also on the likelyhood that the information may be already mutually belived. One of two communciation strategies will be chosen: C for communicating and NC for not communicating. If C is selected, the team has a reward for knowing the information but also has the cost for sending the information. If NC is chosen, two outcomes are possible. There is some probability that the information was already commonly known, in which case the team is given an extra penalty for miscoordimation, besides the reward. Otherwise the team has the reward.

There are four major differences between STEAM and our approach. First, STEAM focuses on establishing joint intentions of team members, but not on analyzing the information production and need among team members in order to be able to provide information proactively or ask for information actively, which is our focus. Second, in STEAM, the decision to communicate is based on monitoring other team members' sensory capabilities and role constraints, while our model can dynamically predict information production and need among agents based on collections of historical data. Third, the communication strategies of STEAM are either to communicate or not, while our model offers agents more realistic options and considers the interdependency of their decisions. Fourth, STEAM does not include risk in decision-making, while we consider the risk an important part in the utility function. This is particularly necessary in hostile environments.

2.3.2. Probabilistic Plan Recognition

Huber and Durfee suggest using a probabilistic plan recognizer [53, 54], similar to their 1993 work, to deduce the status of the commitments of other agents involved in a joint plan. In fact, it is easier to reason about the commitments of other agents when they are known to be following a joint plan than it is to reason about their plans when there is no such basis of knowledge between the observing and the observed agents. All an observing agent need do is compare the observed agent's current actions with the current plan to determine the status of that agent's commitment. Huber and Durfee's system is called the University of Michigan Procedural Reasoning System (UM-PRS) [54]; it is used for mapping plans, and for plan recognition by a Bayesian Net approach

given in [53] (see also [17] for an introduction to bayes nets, or [93] for a detailed description). Huber and Durfee use no vision in their system, assuming that logical predicates can be detected directly by the cooperating agents. Strategies such as plan recognition normally have high computational complexity that weakens teamwork efficiency. However, a major point of our work is that the underlying system interprets team plans of the agents do some of the fundamental work for handling communication. For example, by analyzing preconditions and effects of the team plans and operators, we generate a set of information to be exchanged among the needers and the providers.

2.3.3. Game-Theoretic Recursive Modeling

Because agents who have different knowledge and capabilities must work together, they must communicate the right information to coordinate their actions. Gmytrasiewicz et al. developed a rigorous approach for modeling the utility of communication, based on decision and game-theoretic methods [39, 40]. The model is called Recursive Modeling Method (RMM). An agent that is considering sending a message should base its decision on an estimation of whether the message's recursive impact on the sender and receiver's beliefs will improve the expected outcome of its decisions [39, 40]. In this framework, an agent begins with a recursively elaborated set of models about another agent. Using the probabilistic nature of these models, the agent can compute the expected utility for the other agent's alternative decisions. It then models how exchanged information will influence the probabilities, and thus affect the expected utilities of the other agent's decision.

The common thing between our approach and RMM is that we both model a message's recursive impact on the receiver and the sender in turn by using the decision-theoretic method. However, there are four major differences between our approach and RMM. First, we focus on a multiple agent team cooperating to achieve a common goal, while RMM is primarily suitable for two-player teams and considers agents' decision-making from the perspective of an individual agent in a self-interested environment.

Second, RMM estimates a message's recursive impact regarding to how this message is relevant to the receiver's goal. While in our model, *relevance* of a message is inferred by reasoning about the goals of other agents (specifically, the preconditions of these goals constitute the information relevance to other agents). We model the impact regarding to timeliness and correctness of the information that is not considered by RMM.

Third, RMM uses a decision tree containing probabilities of other agents' actions and the probabilities are domain knowledge. However, obtaining these probabilities when the environment is dynamic and not fully observable is difficult. Our model, in contract, does not rely on pre-determined knowledge, but computes the timeliness and correctness of the information based on possibly incomplete and uncertain knowledge of other agents.

Fourth, RMM uses iterative deepening of RMM levels to detect convergence or cycles. This makes it very costly and time consuming to compute a solution. If the depth of a hierarchy is finite and complete, the model can traverse this hierarchy and

retrieve the best strategy to play. However, because this nested model can involve many branches and extend to deep levels of recursion, so when there is a loop or when there is not enough time to traverse the whole hierarchy, RMM may not be able to return the optimal solution. Our approach makes decisions based on methods for inexpensive approximation to balance the quality of decisions with the costs of making them. While this is not exactly precise, it is shown to be a practical solution in our experiments.

2.3.4. Optimal Communication among a Team

Bui, Kieronska and Venkatesh present a formal framework based on the game theory with incomplete information for modeling the coordination and communication problem among a team of collaborative agents; they also defined what optimal communication means in this setting [11, 41]. The framework defines the notion of team optimality (TOP) to be taken as the ideal solution to the team coordination problem. Communication is considered to be an extended team problem where agents are allowed to broadcast messages. Optimal communication then is defined as a combination of communication strategies with maximal value and minimum cost. To reduce computation complexity caused by unknown parameters in the utility function, the framework uses domain-dependent assumptions to reduce hypothesis space. It also assumes the probability distributions about the unknown parameters are given and suggests these can be learned by the agents through their repeated interactions with the environment and with one other. However it does not give the detail about the learning process. We propose a solution that agents attach some relevant data to messages sent

to each other and use the practical EDF to model distributions of the unknown parameters.

2.3.5. Multi-Agent Markov Decision Process

Xuan and Lesser present a multi-agent extension to Markov Decision Process (MDP) to optimize both actions and communication [133]. They model communication as an explicit action that incurs a cost. They describe how to model communication and the cost of communication properly and define the optimality of combining communication acts for a group of cooperative agents. However, their framework is heuristic and does not consider communication risk in decision-making. To relieve prohibitive computation complexity in the optimization problem, they use social conventions. For example, one of the conventions they use is “no news is good news”. If agents do not hear anything from others, they assume everything is fine and process their work without communication. However, they need to negotiate a new plan if the progress is not as intended. We are interested in the most comprehensive case where cooperative agents must determine which message they should transmit, and when, assuming that communication incurs a cost and a risk.

2.3.6. Dec_POMDP_Com

Decentralized Partially Observable MDP Communication (Dec_POMDP_Com) [41] is a theoretic model for decentralized control of multiple decision-makers that share a common set of objectives. Similar to the Multi-Agent MDP model, Dec_POMDP_Com aims to find a joint policy that maximizes the expected utility over the hypothesis that consists of policies of actions and policies of communications. The

difference between them is that to decrease computation, the former uses heuristic social conventions while the latter uses a myopic greedy algorithm to approximate optimal communication. For example, at the offline planning stage, a best policy is specified and will be used throughout the rest processes. Our approach is more dynamic in that, without using heuristic assumption or offline planning stage, the best policy is always guaranteed at every time of decision-making. To deal with uncertainty, we take advantage of history data of information production and need and use a practical statistical analysis to approximate distributions of the information production and need. This decreases the number of possible outcomes of unknown parameters in the utility function to a finite set.

2.3.7. COM-MTDP

Communicative Multi-agent Team Decision Problem (COM-MTDP) [95] offers a theoretic model that considers the uncertainties and costs in real world scenarios, addressing some of the deficiencies of BDI systems. This model compares complexity results when either free communication, no communication or general communication is assumed. While the model accounts for the value and the cost of communication, it does not consider the risk that we examine in our approach. The authors applied a single case of communication, which allows an agent to send a single message indicating that a certain goal has been achieved. Our work studies a more general problem: the agents take advantage of the timing and frequency of information production and need and are allowed to communicate (possibly) several times until sending out or receiving a teamwork-related information item. We are interested in

problems where agents may act independently to perform their own tasks but may need to exchange their information from time to time to coordinate in more efficient ways. This includes each agent deciding when and what to communicate to whom. Additionally, a single agent's decision is not independent and needs to consider the impact of its counterparts' decisions.

2.4. Other Effective Communication Approaches

2.4.1. Centralization Modeling

In the contract net protocol [115], when an agent needs help from the others in the group, it broadcasts a task-announcement message. The other agents evaluate their resources and submit bids to the original agent. The original agent then evaluates these bids and assigns the task to the most suitable one. The contract net protocol is appropriate in a decentralized control regime where the agent does not know in advance the other agents' information. With the generality of the broadcast, this approach becomes inefficient in many cases. We propose to eliminate the broadcasting of this communication with an assisted coordination approach [38]. In this approach, a central manager agent tracks the overall status of the group; any agent wishing to locate peers sends a message to the manager agent and receives the address of the peer agent. Lashkari's collaborative framework [78] is another example of this approach.

2.4.2. Comparative Reasoning

Sugawara reports on the use of comparative reasoning and analysis techniques for learning and specifying coordination rules for a system in which distributed agents coordinate in diagnosing a faulty network [117]. The investigation is focused on

optimizing coordination rules to minimize inefficiency and redundancy in the agents' coordinating messages. Upon detecting sub-optimal coordination (via a fault model), the agents exchange information on their local views of the system and the problem-solving activity, and construct a global view. They then compare the local view to the global view to find critical values and attributes missing from the local view that gave rise to the sub-optimal performance. These values and attributes are used in constructing situation-specific rules that optimize coordination in particular situations. For example, network-diagnosis agents may learn a rule that guides them to choose a coordination strategy in which only one agent performs the diagnosis and shares its result with the rest of the diagnosis agents.

2.4.3. Social Conventions

Shoham and Tennenholtz suggest that a society of agents adopt a set of conventions [50, 111]. Each agent will obey these conventions and will be able to assume that all others will obey them as well. On one hand, these rules will constrain the plans available to the agents, but on the other, they will guarantee certain behaviors on the part of other agents. This approach totally eliminates communication and uses convention rules to guide agents' actions. As an example, Shoham and Tennenholtz present a number of traffic laws for a restricted domain of mobile robots. They show how social conventions ensure that no collisions or deadlocks occur, and agents are still allowed enough freedom to plan close to optimal paths.

2.4.4. Focal Points

In environments where communication is impossible, Fenster and Kraus explore a coordination technique common to communication-free human interactions, namely focal points [34, 75, 105]. This approach is based on the intuition that humans are sometimes capable of sophisticated interaction with little communication, and that it ought to be possible for agents to emulate this behavior. Focal points are based on the naturalness and intuitiveness of certain objects (or solutions) in the world. Since agents do not have the common sense needed to judge the naturalness and intuitiveness, the designer endows them with an algorithm capable of identifying focal points to which they adhere. They then develop a domain-independent algorithm and test it in simulations of various instances of an abstract world. They find that given a problem and a set of possible solutions from which the agents need to choose, focal points are prominent solutions of the problem to which agents are drawn [34]. In most randomly generated situations, there is more than a 90% probability that agents will make a common choice.

2.5. Observability and Belief Maintenance

In a dynamic, distributed teamwork environment, apart from prior knowledge such as the team goal, observability is a major means for an individual agent to obtain information. An agent with observability may monitor its teammates by observing the environment and their actions and then estimating their beliefs without generating unnecessary messages. In what follows, we review literature about belief and belief maintenance after observation.

2.5.1. Knowledge and Belief

In [112, 94], the internal state of an agent is called its mental state, and it is represented by modal logic. One important modality in mental state is belief. Building belief and the process of belief revision and update are very complex [74]. If we want to introduce belief, we have to introduce knowledge first. Knowledge, belief, and the relationship between them have been studied extensively in philosophy for a long time.Most work in Artificial Intelligence (AI) on knowledge and belief has its origins in the philosophical work of Hintikka [49]. Moore was an important early researcher who introduced Hintikka's ideas into AI [89]; the most comprehensive and up-to-date discussion of knowledge and belief in computer science appears in [31].

Most formalization of knowledge and belief is expressed in modal logic. The standard logic for knowledge, called the S5 system, contains the following axioms:

- K $[K\varphi \wedge K(\varphi \supset \psi)] \supset K\psi$
- T $K\varphi \supset \varphi$
- 4 $K\varphi \supset KK\varphi$
- 5 $\neg K\varphi \supset K\neg K\varphi$

The K axiom says that an agent's knowledge is closed under deduction, while the T axiom says that what the agent knows is true. Axiom 4 implies that the agent knows what it knows, while axiom 5 says that it knows what it doesn't know.

A logic of belief results from dropping the T axiom from S5 and using the operator BEL instead of K. The derived system is called K45. In fact, the most

common logic of belief is a strengthening of K45 by the D axiom: $BEL\varphi \supset \neg BEL\neg\varphi$ (the resulting system is called KD45). Intuitively, the D axiom ensures that the agent's belief is internally consistent.

2.5.2. Visibility, Seeing, and Knowledge Logic

In recent years, observability has been used widely to understand behaviors of multi-agent systems. One study of particular interest is logic for visibility, seeing and knowledge (VSK) [128, 129, 130, 131], which explores relationships between what is true, visible, perceived, and known. The VSK logic is an extension of modal logic. The semantics of the VSK logic is based on Kripke possible worlds [31]. A space of Kripke structures ("worlds") is defined, each of which encodes the instantaneous state of environment plus the internal state local to each agent. Then several equivalence relations are defined to capture the meaning of the modal operators. For example, for each world, there is a relation, $\sim_v$, that determines what other worlds are indistinguishable, and similarly for S and K. The content of what an agent sees or knows is determined by these equivalence relations. For example, an agent is said to know φ if φ is satisfied by all the worlds accessible from the current world. Wooldridge goes on to prove some properties about the interrelations among accessibility, sensibility and knowledge in this system, and he also offers a proof of the theory with a guarantee of completeness [128]. Wooldridge also investigates a number of interaction axioms among agents, such as under which conditions agent *a* sees everything agent *b* sees, or agent *b* knows everything agent *a* sees [128].

However, there are three major issues regarding agent cooperation that are not addressed in VSK logic: 1) the effects of actions play a major role in helping an agent infer what others likely know, while there is no way to treat actions through observation; 2) agents do not have an effective way to utilize their observation to manage communication and 3) VSK models knowledge (from observation), but not belief. Agents should be allowed to believe different or even incorrect things and maintenance of multiple agents' beliefs are a difficulty problem. Our approach uses these issues and agents' observations of the world to determine which information is already believe by other agents, and therefore does not need to be communicated.

2.5.3. Beliefs of Agents

The VSK logic introduced in last section is basically suitable for describing and reasoning about belief and observability of a single agent. In multi-agent systems, agents are expected to not only reason about belief and observability of itself but also of others.

Beliefs of Agents (BOA) is a multi-agent belief maintenance and reasoning model, from both theoretical and practical aspects [101]. It is able to represent multiple states of beliefs and justify beliefs with different strengths [60]. To achieve fast and efficient reasoning, BOA implements multi-agent belief reasoning in a first-order logic back-chainer by sacrifices some degree of expression (i.e. it does not handle things like nested belief) [7].

Comparing with a multi-agent truth maintenance system which maintain the integrity of observed and communicated information [28, 55, 84, 85], BOA answers two difficulties that are not addressed by the later.

The first is resolving conflicting beliefs about a certain thing. In the multi-agent truth maintenance system, a single agent is generally not free to change the status on its own accord and must coordinate with the other agents so that they are all consistent on the status of the information. However, agents may come to conflict beliefs about a certain thing. For example, agent *a* may receives a message from agent *b* saying it is raining now, but agent *a* currently does not observe the rain. BOA resolves this problem by reasoning about the justification for the beliefs, including direct-observation, observability, effects of actions, inferences, persistence and default knowledge [60].

Second, the main purpose of reasoning about beliefs and observabilities is to help agents assist each other. Typical a belief is a fact (proposition) with the value true or false. BOA represents the fact that another agents belief is unknown (neither true nor false) or whether (either true or false). Then agent *a* would provide a piece of information to agent *b* if agent *a* believes agent *b* does not know the value for the information; also agent *a* would ask about needed information from agent *b* if agent *a* believes whether agent *b* know the truth value of the information.

2.5.4. Seeing Is Believing

The observability and reasoning of a single agent have received researchers' attention for some time. Perception reasoning is one of these research directions [72,

73]. For example, "seeing is believing" has been adopted for perception-based belief reasoning [25, 91]. These theories are intended to apply to perception in humans and to perception in agents at the level of symbolic interface between a vision system and a belief system. They give a logical analysis of perception and then consider when perception should lead to change of belief. Similar work can be found in [122]. Van Linder et al. describe different ways agents can acquire information: seeing, hearing, and jumping (default reasoning) [82]. They also propose a classification of the information that an agent processes according to credibility. Agents then can solve various conflicts that may arise when acquiring information from different sources, based on information credibility.

2.5.5. Nested Belief Reasoning

Isozaki and Katsuno [61] propose an algorithm for estimating others' beliefs from observation. An agent maintains its own belief by checking three factors: 1) observation factor: if one observes a proposition now, one believes the proposition now; 2) effects factor: if one has just observed an action, then one believes in all of its effects, even if one has not yet observed them; 3) memory factor: if no new information is available, one's previous belief remains valid. An agent *a* can estimate agent *b*'s belief at different times:

- *b's* initial belief: agent *a* checks *b*'s observation factor at initial time;
- *b*'s belief at a time later than initial: agent *a* checks *b*'s observation factor, effects factor, and memory factor at that time.

Isozaki and Katsuno [62] also propose a way to reason about nested beliefs (which are one's belief about what another believes) based on observation. However, neither of their works represents the process of observation, i.e., what can be seen and under which conditions.

2.5.6. Cooperation by Observation

Kuniyoshi, Rougeaux and Ishii [77] proposed a cooperation framework called "cooperation by observation". Its basic function is to allow minimal communication supported by mutual observation of actions. Agents cooperate by using visual action-recognization to classify task patterns. Their framework presents several standard attentional templates, e.g. who monitors whom. They define a team attentional structure as one in which all agents monitor each other. Viroli and Omicini [123] devise a formal framework for observation that abstracts conditions that cause agents' interactive behaviors. Kaminka and Tambe [67] use observation to monitor failed social relationships between agents, but they do not give details about how agents' belief about their teammates' mental states are updated.

2.6. Problem-Specific Prediction

There are many learning techniques for problem-specific estimation. Here we just name a few.

2.6.1. OVERSEER

Kaminka et al. propose OVERSEER, a statistical model for exploring plans used by a team to predict team responses effectively during execution [68]. They consider communication to be observable action (only to sender and receiver agents)

and use plan recognition to predict future observed messages. They assume that the duration of a plan is an exponential random variable, and parameters in exponential distribution can be acquired from domain experts or learned from previous runs. However, they neither explain why the duration of a plan conforms to exponential distribution, nor investigate the learning processes in depth. More detail of this work can be found in [68] and [69].

2.6.2. Successful Story Learning

In a dynamic teamwork system, agents need to consider not only the dynamic changes of the system, but also the actions of their teammates. Agents may have some historical data for predicting the actions of others. Schmidhuber and Zhao [106] consider a system with three self-interested agents. The agents learn evidence released during the course of interaction and use a backtracking method called "successful-story algorithm" to establish success histories of behavior, i.e. agents keep actions that have been successful and remove actions that have failed. In this way, the successful histories can be enforced despite interference from other agents.

2.6.3. Regression Modeling

Hu and Wellman [52] adopt regression methods for online derivation of relations between other agents' actions and their internal states. They find that performance of an agent can be quite sensitive to its assumption about the policy of other agents, and when there is substantial uncertainty about the other agents, minimizing assumptions might be the best policy. Another example is Jensen, Atighetchi and Lesser [66]. They investigated techniques for allowing agents to gain

statistical knowledge about non-local effects (NLEs) and found that a combination of three simple learning techniques (empirical frequency distributions, deterministic properties of schedules, and linear regression) can be surprisingly effective.

2.7. Psychological Study of Shared Mental Model in Human Teamwork

Teamwork is a collaborative activity with diverse knowledge resources and distributed team formats. As a team increases in size, it is often difficult or impractical to put all necessary information on a single agent. Furthermore, though today‘s advanced telecommunications and collaboration technologies allow collaborations within geographically distributed team members, coordination in a distributed, large-scale team is still problematic because working from a distance brings increased coordination overhead, communication overload and substantial delays [30]. Ioerger presents an overview of current research of human teamwork, focusing on modeling teamwork in human-behavior representation simulations in command-and-control domains [59]. We review the human teamwork from psychological aspect focusing on shared mental model.

Team psychology research literature suggests that mechanisms like shared mental models aid coordination [12, 73, 99]. Shared mental models refer to organized knowledge that members share about things like the task, each other, goals and strategies [12]. A recent study of teamwork in a flight simulation task found that shared mental models had a positive effect on team coordination, which improved performance [83]. Studies of software teams have also found that their team members need to acquire, share, and integrate substantial amounts of knowledge of the

application domain to ensure positive outcomes [23, 124]. Another study with software requirement analysis teams found that teams that exhibited a "collective mind", i.e., a shared understanding of the group's task and each other [125], were more coordinated because members understood how their work contributed to group outcomes [22].

In this research, shared mental models is developed based on shared knowledge about team structures and teamwork procedures, which help team members to form accurate information and expectations about the task and each other. This is achieved through observing the environment and teammates' actions, predicting teammates' information production and need, and communicating teamwork related information.

2.8. Context of Work at TAMU

The long-term research goal of the research group at TAMU is to develop an intelligent-team training system (ITTS), which involves both humans and agents. By playing vitual team-member roles, agents train humans and improve the humans' teamwork skills. Previous work includes three parts:

1. TaskableAgents, a single agent architecture that provides adaptive task decompositions.
2. CAST (Collaborative Agents for Simulating Teamwork), an architecture for simulating multi-agent teamwork.
3. Various proactive information exchange algorithms with different focuses on inter-agent communication.

2.8.1. TaskableAgents

TaskableAgents is a general single agent architecture which was originally motivated by the purpose of simulating activities of staff officers in tactical operation centers in Army combat simulations [139, 140, 57]. TaskableAgents is mainly consisted of TRL (Task Representation Language) and APTE (Adaptive Protocol for Task Execution), a task decomposition algorithm.

TRL provides descriptors for representing four fundamental types of information: goals, tasks, methods, and operators [57]. Each descriptor starts with a keyword, such as :TASK or :METHOD, a symbolic name, and a list of formal parameters. The parameters allow arguments to be passed in when a task is invoked. In TaskableAgents, greater emphasis is placed on encoding pre-determined tasks and methods. This knowledge defines what to do under various circumstances by providing procedural descriptions similar to high-level programming languages.

The tasks and goals assigned to an agent are carried out by the APTE algorithm (Adaptive Protocol for Task Execution) for task decompositions [57]. Conceptually, there are two phases to APTE. In the very first time step of the simulation, APTE takes the top-level tasks given to the agent and expands them downward as a tree by: 1) selecting appropriate methods for tasks, 2) instantiating the process networks for selected methods, 3) identifying sub-tasks that could be taken in the initial situation, and recursively expanding these sub-tasks further downward. Once the expansion is down to the set of concrete operators, the execution takes the first step as selecting one (perhaps based on priority or preference) and executes it. In every subsequent time

step, APTE must repair the task-decomposition tree. This partly involves marking the action just taken and moving tokens forward in the next process. More importantly, APTE also re-checks each of the termination conditions associated with the tasks and methods in the current tree. If a termination condition has been reached (generally indicating failure), APTE backtracks and tries to find another method that satisfies the parent task. If a task at some level has successfully completed, then a step forward can be taken to the parent process.

In TaskableAgents, agents communicate with each other through the use of built-in TRL operators for sending messages. After being received, messages are stored in a queue local to each agent. At regular intervals (between normal decision-making cycles), all messages stored in an agent's queue are emptied into its knowledge base. The agent can then process the messages accordingly using message-handling methods written in TRL [139, 140].

Different from the planning system that usually relies on goal-regression to select sequences of actions, the TaskableAgents focuses on dynamically selecting and managing tasks. It has two distinguishing features: 1) reasoning about how to select the most appropriate method for any given task, and 2) being able to react to significant changes in conditions and find alternative methods when necessary. TRL is expressive enough to allow the specification of complex procedures for the agent to follow, and the APTE algorithm enables flexible behavior in the form of reactivity to changes in conditions.

2.8.2. Collaborative Agents for Simulating Teamwork

CAST is a multi-agent architecture that simulates and supports teamwork involving both human and software agents [135, 137]. Motivated by psychological studies about human teamwork demonstrating that intelligent team behaviors rely on overlapping shared mental models among team members, CAST is based on the shared mental model [118, 100], which states that, by default, all agents are assumed to share common knowledge about the roles, capabilities and responsibilities in which they are involved within the team, and that they believe all other agents have the same beliefs [137]. This assumption reduces the amount of knowledge a team member should have and simplifies the belief reasoning among agents. From a teamwork-theory perspective, CAST is close to shared-plan theory. CAST starts with only partial knowledge of the shared environment and the other participants and uses communication and individual information-gathering to determine what the appropriate action is, who should perform it, and so on.

Belief reasoning has been recognized as being intractable [47]. Consequently, representing and updating agents' mental states is a challenging problem. CAST deals with this problem in two ways. First, we make a teamwork procedure (such as roles, responsibilities, and plans), and all agents share it. This procedure is represented by MALLET language (Multi-Agent Logic-based Language for Encoding Teamwork). Second, it uses Petri Nets to model the team member's plans, as a computational form of a shared mental model.

MALLET is a team knowledge representation language [136]. The team knowledge includes team structures (such as team members, agents, roles, responsibilities, and capabilities) and teamwork processes (such as team goals, team plans, and individual plans). In terms of a MALLET specification, members of a team share a static portion of common knowledge described by MALLET. MALLET assumes that a team has a set of goals to be achieved. Goals are achieved by assigning a team of agents to plans and then invoking these plans, meaning that the agents are ordered to achieve the goal. Each plan consists of a set of steps, each of which is either a primitive operator (e.g., moveleft), or a composite operation (e.g., a sub-plan). Both plans and operators have preconditions and effects associated with them. Each precondition and effect is a conjunction of predicates. The difference between operators and plans is that operators do not have any body. Plans are essentially designed to describe processes which give plans a hierarchical structure. The processes consist of invocation of operators, or arbitrary combinations using various constructs such as sequential, parallel, branch, and iteration. The syntax of processes can be defined recursively based on these constructs.

2.8.3. Proactive Information Exchange

Based on TaskableAgents and CAST, the research group at TAMU bore rich fruits in proactive information exchange research. Three major algorithms are developed with different focuses on inter-agent communication.

Team-to-Individual Plan Conversion (TIP-C) algorithm [6] is the first attainment towards multi-agent communication. It takes plans written in a multi-agent

teamwork language (MALLET) and converts them to equivalent individual plans in a single agent language (TRL). The algorithm analyzes agent responsibilities and automatically inserts necessary and appropriate communication acts to the individual plans and will still facilitate proper dynamic teamwork. These communication actions can help to produce the complex team activities such as delegation of responsibilities, carrying them out, and providing backup behavior.

Dynamic Inter-Agent Rule Generator (DIARG) algorithm is the preliminary implementation of the idea of proactive information exchange [135]. It embeds in the CAST kernel that enables CAST agents to decide on-the-fly how to provide information proactively to teammates to assist their work. DIARG generates communication in order to resolve ambiguity of responsibilities and to predict information needs among agents and generate the necessary (proactive) communication to fulfill these needs. DIARG includes two parts: offline and online. The offline part analyzes the preconditions and effects of operators and generates an information flow describing potential information needs. An information flow is defined as a three tuple <info, providers, needers>, where info is the predicate name together with zero or more arguments; providers is a list of agents who might know such information; and needers is a list of agents who might need to know the information. The online part infers the potential information needs by reasoning preconditions and effects of actions/plans and generates information flow that is a list of needers and a list of providers for every piece of information.

Proactive Information Exchange (PIEX) [60, 101] improves DIARG in two important ways. First, unlike DIARG which is based on analyzing a static predicate network, PIEX monitors teammates' responsibilities and encodes them in a shared plan [60]. Agents anticipate information needs based on these responsibilities; this is more flexible than the offline information flow generated by DIARG. Second, PIEX includes reasoning about other agents' beliefs to reduce unnecessary message exchanges. It provides belief justifications to resolve conflict beliefs. It is also able to represent *unknown* and *whether* states for other's beliefs. The communication becomes more efficient by narrowing down the receivers of a message to those agents who does not know the information, and narrowing down the provider for a message to those agents who know whether the information is true or false.

Our research, Proactive Communication, is similar to PIEX in that both utilize agents' sensing capabilities to reduce what information must be sent. Agents will infer some aspects of the mental states of other agents by observing the environment and the actions of the other agents. This, together with reasoning about what others can see, will allow an agent to decide when it does not need to send information to other agents and whom to ask when it needs information, in a manner that reduces overall communication.

Moreover, Proactive Communication provides a communication solution that makes decisions under uncertainty according to cost, risk and the value of information the communication conveys. DIARG requires a domain expert to publish frequencies associated with information production and information need. It looks at the general

frequency, i.e., for a piece of information, in spite of how many agents produce it or need it, there is only one frequency related to the information production or the information need. Hence, this approach is too rigid to apply to different situations. For agent communication, DIARG also imposes obeys a regular rule, which says that information that is needed more frequently than it is produced must be told proactively; otherwise it must be asked for actively. Proactive Communication develops a more general way to deal with frequencies of information production or information need. Also it uses the decision theory to guide agents to make optimal communication decisions under dynamic situations.

CHAPTER III

PROACTIVE COMMUNICATION: AN OVERVIEW

In this section, we introduce our system archtecture and agent execution cycle, and give an overview of Proactive Communciation mechanism. As discussed in Section 2.8, this research is based on CAST [135]. We investigate how to add two important capabilities humans use, observability and proactivity, to CAST agents in order to emulate, as closely as possible, the principles used by humans to achieve effective Proactive Communication.

3.1. The OP-CAST Architecture

We develop an OP-CAST architecture, as shown in Fig. 3.1, for Observant and Proactive CAST, which is an extension of CAST. The extension is threefold:

- Giving agents observabilities and developing Observation-Based Proactive Communication (OBPC) algorithms for reducing communication load through agents' observabilities.
- Developing Dynamic Information Prediction (DIP) methodology for helping agents make communication decisions, by predicting information production and needs among the agents dynamically.
- Developing Decision-Theoretical Proactive Communication (DTPC) methodology by which agents communicate proactively by evaluating cost, risk and value of communication in the decision-theoretical approach.

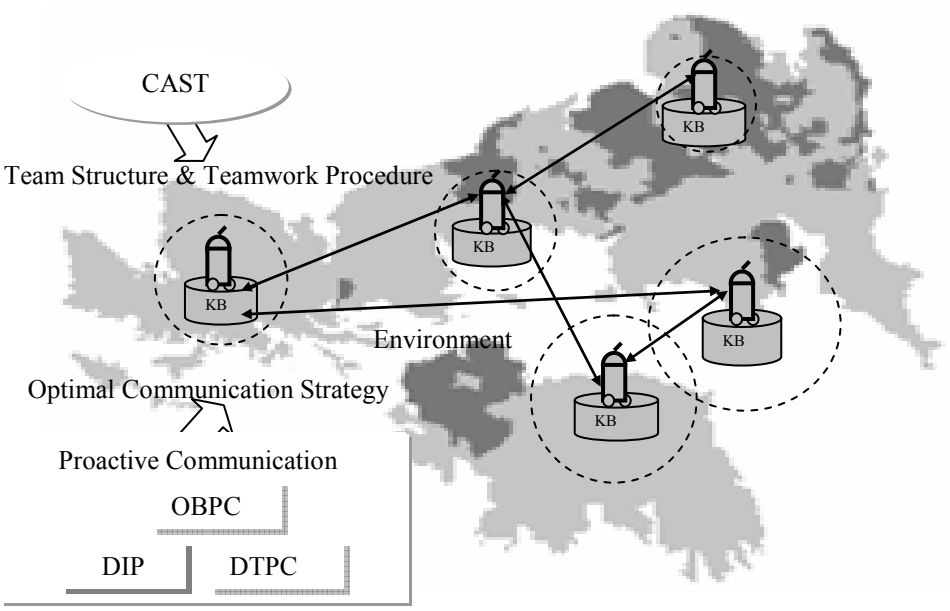


Fig. 3.1. OP-CAST Architecture.

An agent team is composed of a set of agents. An environment simulation provides an interface through which the agents interact with the environment. The team members share the knowledge of teamwork processes as well as team structures, controlled by CAST. Each agent has an individual knowledge base (KB) to specify its beliefs to the environment and other agents. During plan execution, individual agents observe the environment and their teammates' behaviors[1]. Dotted circles in Fig. 3.1 indicate agents' observability radiuses. Different agents have different radiuses and their radiuses may overlap. Agents communicate with each other by exchanging

[1] In this research, observation is not limited to vision; rather it means perception through sensors with which the agents are equipped.

teamwork-related information[2]. The teamwork-related information and its provider and needer are not chosen arbitrarily; they are specified by exploring team plans. Straight lines in Fig. 3.1 connect information provider and needer and arrows show the communication could occur in either direction[3]. Decisions about optimal communication strategies are supported by Proactive Communication.

3.2. Agent Execution Cycle

Our system uses discretized time. At each time step, every agent has an execution cycle, shown in Fig. 3.2:

- First they observe the environment and teammates' actions and adjust their own beliefs;
- If they produce or need some information, they will predict the information need or production of others;
- They choose optimal communication strategies. The decision may be to communicate or not at this time;
- They execute the strategy chosen;
- They act with teammates and the actions affect the environment and enter the next time step.

[2] There are two kinds of information that can be communicated. One is the information explicitly needed by an agent to complete a given plan, i.e., conjuncts in a precondition of plans or operators that the agent is going to perform. The other is the information implicitly needed by the agent. For example, if agent *a* needs predicate p and knows p can be deduced from predicate q, even if the providing agent does not know p, it still can tell agent *a* about q once it has q, because it knows that agent *a* can deduce p from q. This research, however, deals only with agents communicating information that is explicitly needed.

[3] This research, however, does not consider chaining communication, such as communicating via third-party brokers.

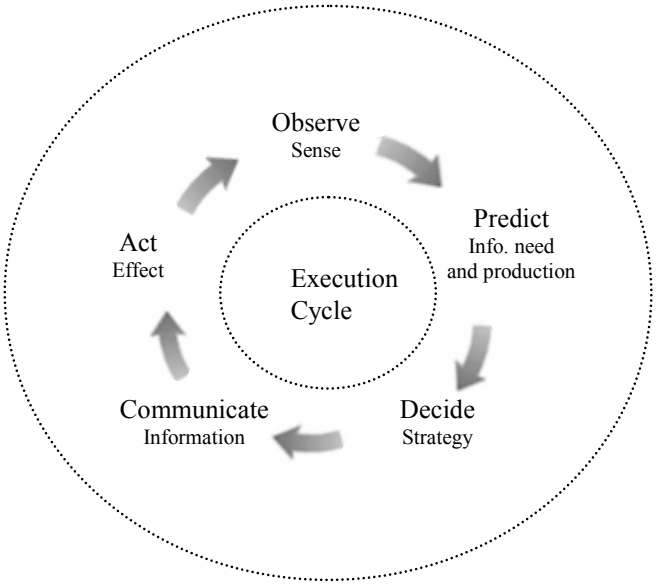


Fig. 3.2. Agent Execution Cycle.

3.3. Proactive Communication

Proactive Communication is a decision-theoretic communication mechanism to choose the optimal communication strategy during multi-agent teamwork, by giving agents the capabilities of observability and proactivity. Observability helps agents to monitor the environment and track teammates' mental states. Proactivity allows agents to anticipate future needs or changes and tell or ask each other about teamwork-related information. To achieve these objectives, we divide the problem into three pieces and developed solutions for each.

3.3.1. Observation-Based Proactive Communication

To endow agents with observability, we express what an agent can see. Additionally, successful teamwork requires interdependency among agents [45], which suggests that agents should know something about others' observabilities. Consequently, we also express what an agent believes another agent can see. In order to explain how agents use their observabilities to observe the environment and others and how they reason about others' observabilities, we clarify notions of:

- What an agent can see, what it actually sees, and what it believes from its seeing.
- What an agent believes another agent can see, what it believes another agent actually sees, and what it believes another agent believes from its seeing.

The purpose of introducing observability is to reduce excessive communication, but there are some fundamental issues to be addressed first. We define which kind of information will be communicated, who needs it and who provides it, by reasoning about team plans. We also develop two algorithms:

- To determine whether or not an agent having information should tell another agent, based on its belief about what the other agent can see.
- To determine whether to ask some specific agent for needed information, based on the needer's belief about what the specific agent can see.

OBPC is introduced in Chapter IV.

3.3.2. Dynamic Information Prediction

Agents need to know values of teamwork-related information. However, the values of the information change in a dynamic environment. It is impossible for agents to know all information at all times.

For a piece of information, we assume that the production time interval by an agent and the need time interval by an agent are random according to some unknown distributions. A key aspect of DIP is to estimate the time of information production and need of teammates based on these distributions. As a basis for accomplishing this, we have agents send a partial past history of the time intervals of their information production or need when they send or ask for information. This additional information can often be sent at modest cost and subsequently enables the receiving agent to make predictions about the information production or need times of other agents. After gathering previous data on information production and need opportunistically, we use a practical Empirical Distribution Function (EDF) [15] to approximate the distributions of information production and needs of other agents. The distributions are used in the utility functions of communication strategies to help agents decide whether or not to tell or ask for a piece of information. DIP is introduced in Chapter V.

3.3.3. Decision-Theoretic Proactive Communication

We develop a decision-theoretic approach to determine whether to proactively tell (relative to the need for the information) a piece of information to one or more other agents, and whether and which agent(s) to ask for a piece of information (relative to the production of the information). We equipped agents with a set of communication

strategies for different decision-making situations. The utility (difference between cost, risk and value) of the strategies allows agents to decide which one to apply. Since not all parameters in the utility function are known, we approximate their distributions with respect to the dynamic estimation of distributions of information production and need. The utility function will be used to evaluate agents' decisions and their estimations of other agents' responding decisions. Taking others' decisions into account enables the agents to deal with the decision interdependency of team cooperation and to communicate in a way benefiting the whole team. DTPC is introduced in Chapter VI.

3.4. Summary

The OP-CAST agents are endowed with two capabilities to pursue realistic behaviors and effective interactions: observability and proactivity. The first one enables them to track teammates' mental states and decrease the communication load, and the second one allows them to estimate information production and need of teammates, so they can assist teammates at the proper time. These capabilities are encoded in three mechanisms we developed. OBPC formally defines agents' observabilities and deduces extraneous communication by reasoning about the observabilities. DIP estimates distributions of information production and need. The distributions are used in capturing complex decision interdependency among agents. DTPC provides agents an optimal communication strategy when they act in uncertain and dynamic environments.

CHAPTER IV

OBSERVATION-BASED PROACTIVE COMMUNICATION

4.1. Motivation and Overview

A major problem with CAST is that significant status information must be communicated among agents, and there is no attempt to utilize agents' sensing capabilities to reduce the amount of information sent. A more realistic approach (from a human perspective) is to give the agents sensing or observing capabilities. Although partial observability of dynamic, multi-agent environments has gained much attention [95, 68, 60], little work has been done to address how to process what is observable and under which conditions; how an agent's observability affects the individual's mental state and whole team performance; and how agents can communicate proactively with each other in a partially observable environment.

To address these issues, we introduce an explicit treatment of an agent's observability that aims to achieve more effective communication among agents. We employ the agent's observability as the major means for individual agents to reason about the environment and other team members. Agents will infer some aspects of the beliefs of other agents by observing the environment and the actions of the other agents. Together with reasoning about what others can sense, these inferences will allow an agent to decide when it does not need to send information to other agents and whom to ask when it needs information, in a manner that reduces overall communication.

We implement the following methods to achieve OBPC:

- Reasoning about what information each agent on a team will produce, and thus, what information each agent can offer others. This is achieved through: a) analysis of the effects of individual actions in the specified team plans; b) analysis of observability specifications, indicating what each agent can perceive about the environment and other agents, and under which conditions.
- Reasoning about what information each agent will need in the process of plan execution,through the analysis of the preconditions for individual actions involved in team plans.
- Reasoning about whether an agent should act proactively when producing some information. The decision is made in terms of: a) which agent(s) needs this information; and b) whether or not the agent who needs this information is able to obtain the information independently by observing the environment and other agents' actions.
- Reasoning about whether an agent should ask actively when needing some information. The decision is made in terms of: a) which agent(s) produces this information; and b) which agents are able to obtain the information through observation.

The following sections first present preliminary contextual information. Then, we introduce a representation of observability and algorithms for reasoning about it. Finally we describe algorithms of OBPC.

4.2. Preliminaries

In OP-CAST, the team members share the team knowledge represented in MALLET, which provides descriptors for encoding knowledge about teamwork processes (i.e. individual and team plans and operations), as well as specifications of team structures (e.g., team members and roles) [136].

4.2.1. Plans

Plans are at the center of activity. They describe how individuals or teams can go about achieving various goals. Each plan has a process consisting of a set of operations, each of which is either a primitive operator, or a composite operation (e.g., a sub-plan). A DO statement is used to assign one or several agents to carry out specific operators or sub-plans. Fig. 4.1 is an example plan for the multi-agent version of Wumpus World (refer to Chapter VII for details; a complete version of Multi-Agent Wumpus World MALLET plan is attached in Appendix B).

```
(plan startKill(?fi)
  (pre-cond (newKnow ?wumpusId ?x ?y))
  (process
    (seq
       (DO ?fi (moveToWumpus ?fi ?wumpusId ?x ?y))
       (DO ?fi (shootWumpus ?wumpusId))
       (DO ?fi (retract (newKnow ?wumpusId ?x ?y)))
       (DO ?fi (nextStep ?fi))
    )
  )
)
```

Fig. 4.1. An Example of the Plan.

StartKill is an individual plan for a fighter agent *?fi*. The plan has a precondition which must be satisfied by the fighter before it tries to execute this plan, i.e. the fighter must know the location of a newly found wumpus. *MoveToWumpus* is an individual sub-plan by which the fighter will move to an adjacent location to the wumpus. *ShootWumpus* is an individual operator specified as follows:

(ioper shootWumpus (?wumpusId)

(effects (dead ?wumpusId)))

The effect of this operation will be automatically asserted to the fighter's KB after execution (Section 4.4.5 elaborates an algorithm for updating KB). After *shootWumpus* is executed, this wumpus' id and location will be retracted from the fighter's KB so that the fighter will not kill the same (dead) wumpus at next step.

4.2.2. Actions

MALLET operators are defined based on standard STRIPS (STanford Research Institute Problem Solver) operators [35], i.e. as discrete state transitions with preconditions and effects, which are logical conjunctions. Using STRIPS representation is important because we want to reason about precondition and effect to make communication decisions (see Section 4.5). MALLET has three forms of action: individual action, team action and joint action [32].

We view the world in terms of discrete state transitions and assume actions are instantaneous operations, i.e. they are performed instantaneously. An individual action is the execution of an instantiated operator in a DO statement. It is represented as:

<action> ::= (DO <doer> (<operator-name> <args>*)),

where <doer> is the agent assigned to the action and <operator-name> and <args> correspond to the name and arguments of the operator. Team action is very similar to individual actions except that <doer> denotes a list of agents involved and these agents must perform the action simultaneously [32]. We assume that precondition of an action, individual action or team action, must be believed by <doer> before the action can be performed and the effect must be believed after the action is performed. If the precondition is not believed by <doer>, then Proactive Communication will be implicitly considered (see Chapter VI). Fig. 4.1 illustrates the example of invoking the *shootWumpus* action.

Joint action uses a descriptor *joint-do*. It includes a list of DO statements and specifies three different joint types: AND, OR or XOR. For the type AND, each individual DO action must be executed by the corresponding individual agent before the complementation of the joint activity, which requires all involved agents acting simultaneously. For an OR, at least one DO must be executed by the corresponding individual agent while for an XOR, only one DO needs to be executed. Below is an example type AND joint action:

```
(joint-do AND (?ag1 ?ag2)
    (DO ?ag1 (liftTable))
    (DO ?ag2 (cleanCarpet)))
```

which requires two agents *?ag1* and *?ag2* to cooperate to do a clean operation.

Comparing team action with joint action, they are in common on that all agents involved must perform the action simultaneously. The difference between the two is

that, for the team action, the agents evaluate exactly the same precondition before the action, perform exactly the same action, and apply exactly the same effect after the action; while for the joint-action, the agents only perform the action they are assigned, hence they will only evaluate precondition of that action, perform that action and apply effect of that action.

Our approach focuses on observing individual actions. The ideas can be extended to team actions and joint actions, which essentially are the collection of simultaneous individual actions performed by individual agents.

4.2.3. Environment and Properties

Another important setting for agent teamwork is environment. The environment is composed of objects. Each object has properties. A property is a predicate represented as follows:

<property> ::= (<property-name> <object> <args>*)

<object> ::= <agent>|<non-agent>,

where <object> could be either agent or non-agent, and <args> is a list of arguments describing the property. Sample properties in the Multi-Agent Wumpus World are as follows:

(location ?o ?x ?y),

(dead ?wumpusId).

The environment evolves from the state at one time to the state at the next time with an action possibly being taken during the time interval, saving only the current environment state.

During a teamwork process, the environment simulation provides an interface through which the agents can observe properties and their teammates' actions. We treat the environment as a knowledge base (KB) denoting objective truths of the world. Since actions are domain-dependent, when agents perform the actions, they send a signal to the environment KB. Thus, the actions will be added to the environment KB as a fact. We assume the environment KB is accessible to all agents. Then, the actions can be sensed by those whose observability permits them at the time the actions are performed.

4.2.4. Agent Beliefs

Each agent maintains beliefs about the environment and about other agents in its own KB. These beliefs are used at the time when the agent evaluates precondition of plans or actions it is involved. At that time, an attempt is made to match each conjunct of the precondition to the agent's KB via unification, using variables for any or all of the arguments. Unification will provide values for the free variables that make the conjunct *true*. If there are no such values, then the value for the conjunct is *false*.

The version of MALLET used in this work is based upon the Closed World Assumption that assumes that anything that is not true is false [56]. A limitation of this assumption is that, there is nothing in the language to distinguish between *false* and *unknown*, which are other two important states for belief [60]. For example, suppose an agent cannot prove a precondition *I* from its KB, then what the precondition would be evaluated to, *not I* or *unknown*? This problem will occur frequently, since in many

domains, not everything can be inferred from observability. Therefore, proper handling of *false* beliefs and *unknown* beliefs is important.

The version of MALLET used in this work avoids this problem by defining *wait* semantics for preconditions. That is, if a precondition evaluates to *false*, the agent waits (possibly indefinitely) for the precondition to become *true*. This, of course, could lead to some branches of a parallel process (or even an entire plan) being blocked forever. Proactive Communication algorithms can recognize this situation and invoke communication decision processes to determine what information, if any, is exchanged. From the perspective of MALLET, then, there is not a need to distinguish in the KB of an agent between *false* and *unknown* with regard to the value for an information item *I* used in any precondition.

It still would be better if a more general approach were used. Newer versions of MALLET consider alternative semantics for preconditions, such as failing upon false preconditions, waiting for a maximum length of time and then fail, or trying to achieve the precondition by invoking a planner [32]. Except for the first and last cases, it still is left to the Proactive Communication algorithms to determine whether or not to communicate when a precondition evaluation fails, and thus would not impact the work presented here in a significant way.

For effect of plans or actions in which the agent is involved, all conjuncts are treated as positive facts, with the interpretation that *not I* means to remove *I* from the agent's KB.

4.3. Agent Observability

We define the syntax of observability and give semantics to this observability.

4.3.1. Syntax of Observability

To represent agent observability, we define a meta-predicate CanSense which takes three arguments:

CanSense(<observer> <observable> <cond>)

where <observer> specifies the agent doing the observing, <observable> identifies what is to be observed, and <cond> specifies the conditions under which the <observer> can sense the <observable>.

Successful teamwork requires interdependency among the agents [44]. This suggests that an agent should know at least some things about what other team members can sense. However, an agent may not know for sure that another agent can sense some things. Rather, an agent may only believe that another agent can sense something. We then use

B(<believer> CanSense(<observer> <observable> <cond>))

to mean that one agent believes another agent can sense something under certain conditions. Belief is denoted by the modal operator B and for its semantics, we adopt the axioms K, D, 4, 5 in modal logic [31].

The syntax we use for observability is given in Fig. 4.2.

```
<observability> ::= (CanSense <viewing>)*
                    (B <believer> (CanSense <viewing>))*
<viewing>       ::= <observer><observable> <cond>*
<believer>      ::= <agent>
<observer>      ::= <agent>
<observable>    ::= <property>|<action>
<cond>          ::= <property>
<property>      ::= (<property-name> <object>* <args>*)
<action>        ::= (DO <doer> (<operator-name> <args>*))
<object>        ::= <agent>|<non-agent>
<doer>          ::= <agent>
```

Fig. 4.2. The Syntax of Observability.

For example, the observability specification for a carrier in the Multi-Agent Wumpus World is shown in Fig. 4.3, where *ca*, *rca*, *fi*, *rfi* represent the carrier, carrier's detection radius, fighter and fighter's detection radius, respectively.

An agent has two kinds of knowledge, shared team knowledge, encoded in MALLET, and individual knowledge, contained in its KB. The syntax of observability can be used either as rules in an agent's KB [141], or as capability incorporated into MALLET. In this research, we encode observability as rules in agents' KBs.

```
((CanSense ca (location ?o ?x ?y))
        (location ca ?xc ?yc) (location ?o ?x ?y)
        (radius ca ?rca) (inradius ?x ?y ?xc ?yc ?rca)
) ;The carrier can sense the location property of an object.

((CanSense ca (DO ?fi (shootwumpus ?w)))
        (play-role fighter ?fi) (location ca ?xc ?yc) (location ?fi ?x ?y)
        (adjacent ?xc ?yc ?x ?y)
) ;The carrier can sense the shootwumpus action of a fighter.

((B ca (CanSense fi (location ?o ?x ?y)))
        (location fi ?xi ?yi) (location ?o ?x ?y)
        (radius fi ?rfi) (inradius ?x ?y ?xi ?yi ?rfi)
) ;The carrier believes the fighter is able to sense the location property of an
   object.

((B ca (CanSense fi (DO ?f (shootwumpus ?w))))
            (play-role fighter ?f) (≠ ?f fi) (location ca ?xc ?yc) (location fi ?xi ?yi)
            (location ?f ?x ?y) (radius ca ?rca) (inradius ?xi ?yi ?xc ?yc ?rca)
            (inradius ?x ?y ?xc ?yc ?rca) (adjacent ?x ?y ?xi ?yi))
) ;The carrier believes the fighter is able to sense the shootwumpus action of
   another fighter.
```

Fig. 4.3. An Example of Observability.

4.3.2. Semantics of Observability

To give semantics to observability, we need to consider two perspectives: 1) an agent's observability, which means we need to clarify relationships between what it can sense, what it actually senses, and what it believes from its sensing; 2) an agent's belief about another agent's observability, which means we need to clarify relationships between what it believes another agent can sense, what it believes another agent actually senses, and what it believes another agent believes from its sensing.

4.3.2.1. An Agent's Observability

Our notion of observability derives from Woolridge's VSK logic [128]. Let Sense(a, ψ) denote the notion that agent a senses ψ[4]. Sensing ψ means determining the truth value of ψ, together with unification of any free variables in ψ. The Sense operator is similar to the S operator in the VSK model. The major differences are that, first, in the VSK model S leads to knowledge, $S_a(\psi) \rightarrow K_a(\psi)$, but we only model belief from observation (discussed further below), and agents should be allowed to believe different or even incorrect information. Second, instead of saying that the agent senses the true fact, it is more natural to say that if something is true, the agent will sense the true value, but also, if it is false, the agent will sense the false value. We model the Sense operator as follows:

$$\forall a, \psi, \text{Sense}(a, \psi) \equiv [\psi \rightarrow S_a(\psi)] \wedge [\neg\psi \rightarrow S_a(\neg\psi)].$$

Since $(\psi \vee \neg\psi)$ is an tautology, it follows that

$$\forall a, \psi, \text{Sense}(a, \psi) \rightarrow [S_a(\psi) \vee S_a(\neg\psi)].$$

Next, we consider the relation between sensing something and believing it. We adopt an analogous assumption to the one that "seeing is believing". While philosophers may entertain doubts because of the possibility of illusion, common sense indicates that, other things being equal, one should believe what one sees [5, 91]. The VSK model also suggests that $S_a(\psi) \rightarrow K_a(\psi)$ is the axiom adopted by a trusting agent (of

[4] In our approach, each agent focuses on reasoning about current observation. Time is implicitly taken to be the time of the current step.

no illusions, no sensor fault etc.). When ψ is observed, we assume that the agent believes the truth value of ψ. This is formalized in the axiom below:

$$\forall a, \psi, \text{Sense}(a, \psi) \rightarrow \{[\psi \rightarrow \text{B}(a, \psi)] \wedge [\neg\psi \rightarrow \text{B}(a, \neg\psi)]\},$$

which says if ψ is true, agent *a* believes ψ; if ψ is false, agent *a* believes ¬ ψ.

Finally, we model our observability expression as below:

$$\forall a, \psi, \text{c}, \text{CanSense}(a, \psi, \text{c})$$

$$\equiv \text{c} \rightarrow \text{Sense}(a, \psi)$$

$$\equiv \text{c} \rightarrow \{[\psi \rightarrow \text{S}(a, \psi)] \wedge [\neg\psi \rightarrow \text{S}(a, \neg\psi)]\},$$

which means that if the condition c holds, then agent *a* actually does sense the truth value of ψ.

4.3.2.2. An Agent's Belief about Another Agent's Observability

An agent's belief about what another agent senses is based on the following axiom:

$$\forall a, b, \psi, \text{c}, \text{B}(a, \text{CanSense}(b, \psi, \text{c})) \wedge \text{B}(a, \text{c}) \rightarrow \text{B}(a, \text{Sense}\,(b, \psi)),$$

which means that if agent *a* believes that agent *b* can sense ψ under condition c, and agent *a* believes c, then agent *a* believes that agent *b* senses ψ. Note that agent *a* evaluates condition c according to its own beliefs.

One might wonder if agent *a* can infer the truth value of ψ when it knows that B can sense ψ because it can be easily shown that belief is transmissible between agents, i.e., $\text{B}(a, \text{B}(b, \psi)) \rightarrow \text{B}(a, \psi)$ or $\text{B}(a, \text{B}(b, \neg\psi)) \rightarrow \text{B}(a, \neg\psi)$. However, we do not have such a strong statement of belief on the part of *a*. In order to have the necessary condition given above, we would have to have the condition

$$\forall a, b, \psi, \mathrm{B}(a, \mathrm{Sense}(b, \psi)) \rightarrow \{[\psi \rightarrow \mathrm{B}(a, \mathrm{B}(b, \psi))] \wedge [\neg\psi \rightarrow \mathrm{B}(a, \mathrm{B}(b, \neg\psi))]\}.$$

But, this condition is not necessarily true. All that *a*'s belief that *b* can sense ψ implies is that *b* knows the value of ψ, which is weaker than the statement given above.

4.4. Belief Maintenance

We denote the agent who performs belief maintenance as self and the KB for self as KB_{self}. Self's observability is closely tied to its beliefs about what itself, and what other agents, can sense. The latter is particularly important because it decides the agent's beliefs about others. In this case, the value of the thing that another agent might observe is not of immediate relevance. Only the fact of whether or not the other agent can make the observation. This is treated, again, with the Closed World Assumption. That is, an observability condition is given for the other agent. If self needs to know whether the other agent can sense something, it evaluates this condition. If it evaluates to true, it believes that the other agent can sense the thing. If it evaluates to false, it believes that the other agent cannot sense the thing. There is no *unknown* to consider because the Closed World Assumption is used throughout. False is represented, not explicitly, but by the absence of a true fact[5].

Ioerger [60] has described a belief maintenance system allowing self to maintain tuples about an agent's (possibly a different agent than self) beliefs in the form <*agent I value*>, where *value* for *I* can have one of four values: *true*, *false*, *unknown*, and *whether*, and *agent* is what self believes to have the belief value

[5] This is called Negation as Failure, a concept closely related to the Closed World Assumption [1].

expressed in the tuple. The value *whether* is of value for self's belief about the *agent*'s belief; it means that the *agent* believes whether the value for *I* is true or false, but this value is unknown in KB_{self}. While when the *agent* in the tuple is self, *whether*'s meaning is a bit different; it means self believes the truth value of *I* and this value is known in KB_{self}.

As the version of MALLET we are using is based on values *true* or *false*, we do not need to maintain *unknown* or *whether* directly. With respect to self's belief about another agent's observabilities, we maintain only a fact that indicates the agent can sense the item in question, which indirectly means that the agent can sense *whether,* when the observability condition is satisfied with respect to self's KB.

In the following sub-sections, Section 4.4.1 introduces the concept of belief consistency and compatibility which is the core purpose of belief maintenance [28, 61]. Section 4.4.2 introduces the structure of KB_{self} and how to construct dependencies among beliefs and how to make inference. Section 4.4.3 presents the overall updating function *updateKB*. Section 4.4.4 introduces self's observability reasoning function *reasonSelfObs*. Section 4.4.5 introduces self's belief about others' observabilities reasoning function *reasonSelfBel*. Section 4.4.6 describes the low level belief updating function *update* which maintains belief consistency and compatibility.

4.4.1. Belief Consistency and Compatibility

Belief consistency and compatibility is the core purpose of belief maintenance [78, 135]. Belief can be classified in two types: 1) ground predicates *p* which evaluate to true or false, and 2) functions with arguments *f(?x)* where *?x* denotes a set of

arguments[6]. *f(?x)* does not evaluate to true or false, but denotes some other value. For example, the function location(w1) can take on the value (1 1), meaning the location of W1 is (1 1). In JARE, functions are modeled as predicates in which the function name is converted to the predicate name and the argument list for the predicate includes not only the function arguments, but a list of arguments for the results of the function. Unification will provide the values for the results, if there are any, in which case the predicate evaluates to true; otherwise, the predicate evaluates to false. For example, location(w1) with arguments (?x ?y) is represented as (location w1 ?x ?y).

Belief consistency means that no information and its negation are both believed [28]. Therefore the pair (p, $\neg p$) and ($p(x)$, $\neg p(x)$) can not be believed together in KB_{self}.

However, belief maintenance should consider more general cases such as the following examples:

- Some functions can only have one value at one time. For example, if location(w1) has the value (1 1), then it cannot have another value (2 2), because if w1 is on (1 1) it cannot be on anywhere else.
- Some different predicates cannot be believed concurrently in KB_{self}. For example, (clear x) and (on y x) cannot both be believed because if y is on x, then x cannot be clear.

These examples represent constraints within single predicate or among multiple predicates. These constraints are normally domain dependent and cannot be resolved

[6] We adopt JARE syntax that variables are indicated by symbols prefixed with a '?', and constants are represented by symbols or numbers.

on a general level. Isozaki names this kind of constraint an *incompatibility constraint* and proposes a formula to represent it between two predicates (or within the same predicate) [61]:

incomp(p(?x), q(?y), term1, term2)

where *term1* $\in$*?x* and *term2* $\in$*?y*. *Incomp* means that a ground instance of *p(?x)* and a ground instance of *q(?y)* are incompatible if they are different and *term1* is identical to *term2*. For example,

incomp((location ?o1 ?x1 ?y1), (location ?o2 ?x2 ?y2), ?o1, ?o2),

where (?x1$\neq$?x2)$\vee$(?y1$\neq$?y2), means that if an object is located on one place, it is not located on any other place. Another example,

incomp((clear ?o1), (on ?o2 ?o3), ?o1, ?o3)

means that if one object is on another object, the latter is not clear. To implement this idea, we define a function with the same name *incomp(p, q)* which will return true if two predicate instances *p* and *q* are incompatible.

4.4.2. Inferring Agent Beliefs

We use a backward-chaining theorem prover called JARE (Java Automated Reasoning Engine) [58] to handle belief inference. JARE achieves efficiency by avoiding re-computing references (which is used in forward-chaining inference) and by a little more restrictive representation (e.g. no templates). Rules in JARE are in Horn form which requires that the head of a rule must be a positive literal [102]. A rule is made out of one or more predicates (a rule containing a single predicate is often treated as a fact). The following is a JARE rule:

$$A \wedge B \rightarrow C$$

where C, the head, is called *conclusion*, and A and B, the body, is called *antecedent*. C is derived if both A and B are true. We also say that C is justified by A and B, and {A, B, $A \wedge B \rightarrow C$} is a justification for C. Then C depends on A and B. Since A and B could be conclusions inferred from other rules, the actually antecedents on which C depends could be found by tracing back through A, B, their antecedents, their antecedents, and so on. In our implementation, we assume that no rule contains cycles[7] and the body is made out of positive predicates[8]. Therefore, rules form a directed acyclic graph where nodes are the heads and directed arcs denoting the dependencies.

KB_{self} is initialized as three parts:

- Facts, e.g., identities of objects (agent or non-agent), agents' roles etc.. We assume these facts are commonly believed by all agents, and are certain truth which won't be changed over time.
- Observability rules (self's and others') (Fig. 4.3 presents several observability rules for the Multi-Agent Wumpus World).
- rules which describe what are caused by beliefs generated through observation. The K axiom of the model of belief says that an agent's belief is closed under deduction [31]. For example, if an agent observes

[7] Assumption-based Truth Maintenance Systems (ATMS) is used to solve the problem that rules may contain cycles [26].
[8] Non-monotonic logic is used to model the KB where the body of rule is made out of positive and negative predicates [102, 103].

that the switch is on, it believes the light is on though it cannot directly observe this.

The initial KB_{self} is supposed to have no belief about the world or about other agents' beliefs. These beliefs will be generated dynamically during the teamwork process, mainly by inferring the observability rules and the causation rules. The order of belief inference and the order of belief update by the beliefs derived from the inference are important because of the dependencies among the rules. We handle this by doing the inference first and then the update. Specifically, after self infers a rule, the belief derived from this rule is saved in a temporary place rather than be directly asserted to KB_{self}. Then the order of inference does not matter because all rules share the same base on which the inference is made. But for clarity, we still make the inference in this order – self's observability rules, others' observability rules and causation rules. After all beliefs are derived (possibly from multiple justifications), they will be processed to guarantee that one belief only has one value. Finally, the update process starts and these beliefs are asserted to KB_{self}. In next section, we introduce the process of the belief inference and the belief update in detail.

4.4.3. An Overall Belief Maintenance Algorithm

After a piece of information is inferred from KB_{self}, it may not be asserted to KB_{self} immediately, because there may be different values for this information generated from multiple sources and these values may contradict one another. Five sources generate such values: 1) self's observation, i.e., belief derived from self's observability rules; 2) others' observation, i.e. belief derived from others' observability

rules; 3) causation, i.e. belief derived from causation rules; 4) effects, i.e., conjuncts inferred from the effect of the action self performs; and 5) communication, i.e., messages other agents send to self by communication[9].

In any situation in which belief is acquired from multiple sources, conflicts may arise – in terms of inconsistency or incompatibility. For example, observation may produce p and causation may produce ¬p but we cannot omit either of them. A strategy is needed that prescribes how to maintain KB_{self} in this case. Castelfranchi proposes that such a strategy should prescribe that more credible information should always be favored over less credible information [16]. Ioerger introduces multiple justification types for beliefs and places them in a preference ordering according to strength [60]. To define a strategy conforming to these ideas, we assume that each belief is associated with a priority that decreases in the order shown in Table 4.1[10].

Table 4.1. Belief Strengths.

Source	Priority
Self's observation	5
Others' observation	4
Effects	3
Causation	2
Communication	1

[9] Effects and communication are not defined as rules in KB_{self}. This is because that effects and communication may contain negative predicates but JARE does not allow the head of a rule to be negative. Though we can improve this by renaming ¬p/¬p(X) to notP/notP(X) and maintain the truth value of the pair, it would be highly inefficient. So effects and communication come from external sources but not from KB_{self}.

[10] Belief persistence is handled separately in belief update session in order to maintain belief consistency and compatibility (see Section 4.4.6).

The rationale for this order is as follows[11]. An agent always believes what it senses and what other agents sense because we assume "seeing is believing". The belief about effects of actions the agent performs is secondarily reliable by assuming the agent cannot deny the actions performed by itself. Last, the beliefs caused by observations override what the agent hears by assuming the agent trusts its own inference more than what others tell it. The truth value of a belief is always supported by the rule with the highest priority and whose antecedent is satisfied.

One thing worth of mention is that there may be multiple equally preferred rules with the same strength. For example, an observable item could have multiple justifications from observation. In our implementation, the preference depends on the order in which the rules are applied, i.e. the newly generated value will override the old one, implying that the last rule has the highest priority.

An algorithm for overall belief maintenance along with the observation process is shown in Fig. 4.4. It is executed independently by each agent, self, after the completion of each step in which self is involved, i.e., upon completion of an action. During an update cycle, self will sequentially perform:

- At time t-1, self performed action.
- Immediately after completion of the action at time t-1, self will do *updateWorld* by its last action. Basically, the environment simulation updates the environment KB after the action by self.

[11] This is the order fitting our system and assumptions. Different orders may be applied for different problems. In our system, we never directly obtain a value from inference of others' observation, because we only know that they can sense an item, not the value they sense. However, in a more general setting this source of information may be possible; hence, we included it for completeness.

- Because self can infer the effect of its own action, it will keep the effect and the credit of the effect in a temporary location called *infoList*. Since there may be multiple conjunct inferred, each will be indexed in *infoList*.
- Self will do observation and reason causation, keeping results and credits in *infoList*.
- Self will check messages, keeping results and credits in *infoList*.
- Then, for each piece of information in *infoList*, self will choose a value with the highest credit and do two things: 1) *update* its KB by this value, and 2) communicating this value, if so decided (this is not shown in Fig. 4.4).
- Loop back to next action.

This algorithm maintains belief consistency by the fact that, when there might be conflict assertions, only the one with the highest credit will be asserted to KB_{self}.

The *updateWorld* function is simply a call to the environment telling it to update itself in accordance with the parameters provided. *ReasonSelfObs* infers self's observability rules. *ReasonSefBel* infers self's beliefs about others' observabilities. The method *update* is a low level procedure for updating KB_{self}. The next three sections describe the latter three functions in turn.

```
/* The algorithm is executed independently by each agent after the
        completion of each step in which the agent is involved, i.e., upon
        completion of an action. An action may just be a no-op (e.g., if the
        agent is waiting for a precondition to be true).
   The executing agent is denoted self.
   Below,
        let KB_self denote the knowledge base for the agent self.
        let KB_env denote objective truths about the environment.
*/
updateKB(self, action, KB_self){
    infoList=null;

    updateWorld(self, action);        //notify the environment to update KB_env

    {par
        ∀ I in the effect of action
            infoList ← (I, 3);          //the credit of effect is 3

        infoList ← reasonSelfObs(self, KB_self);
        infoList ← reasonSelfBel(self, KB_self);
        ∀ I derived from causation rules
            infoList ← (I, 2);

        ∀ coming message about I
            infoList ← (I, 1);
    }//end of par

   ∀I∈ infoList
        let info be the value for I with the highest credit;
        update(KB_self, info);
}
```

Fig. 4.4. An Overall Belief Maintenance Algorithm.

4.4.4. ReasonSelfObs: Reasoning Beliefs about Agent's Own Observability

The algorithm for inferring what an agent has observed, according to its observability rules, is given in Fig. 4.5. This algorithm builds beliefs in KB_{self} by checking two things.

```
reasonSelfObs(self, KB_self){
    list=null;

   ∀ rule ∈ KB_self of the form (CanSense self (property-name object args)
   cond)
      if KB_env |= cond
        if KB_env |= (property-name object args)
              list←((property-name object args), 3);
        else
              list←(¬(property-name object args), 3);

   ∀ rule ∈ KB_self of the form (CanSense self (DO doer (action-name args))
   cond)
      if KB_env |= cond
        if KB_env |= (DO doer (action-name args))
              list←((action-name doer args), 3);
        else
              list←(¬(action-name doer args), 3);

    return list;
}
```

Fig. 4.5. An Algorithm of Reasoning Agent's Observability.

When evaluating observability of a property, (CanSense self (property-name object args) cond), self checks if KB_{env} entails cond. If so, and if this property holds in the environment, self adds (prop-name object args) and its credit to *inforList*. If the

property does not hold in the environment, self adds ¬(property-name object args) and its credit *to infoList.*

In the case of (CanSense self (DO doer (action-name args)) cond) where doer≠self, self checks if KB_{env} entails cond as well. If so and if this action holds in the environment, self adds (action-name doer args) and its credit to *inforList*. If the action does not hold in the environment, self adds ¬(action-name doer args) and its credit *to infoList*.

4.4.5. ReasonSelfBel: Reasoning Beliefs about Others' Observabilities

Fig. 4.6 introduces an algorithm for inferring what an agent can determine about what other agents can sense. The algorithm records which agents are believed to sense what. We still consider two cases.

In the case of (B self (CanSense Agd (property-name object args)) cond), if KB_{self} entails cond, self believes Agd senses the property and adds this belief and its credit to *infoList*. If KB_{self} does not entail <cond>, self believes Agd does not sense the property and adds this belief and its credit to *infoList*.

In the case of (B self (CanSense Agd (DO doer (action-name args))) cond), cond is evaluated with respect to KB_{self} and self will update *infoList* in the similar way.

```
reasonSelfBel(self, KB_self){
    list = null;

   ∀ rule ∈ KB_self of the form (B self (CanSense Agd (property-name object
   args) cond))
        if KB_self |= cond
                list←((Sense Agd (property-name object args)), 3);
        else
                list←¬(Sense Agd (property-name object args));

   ∀ rule ∈ KB_self of the form (B self (CanSense Agd (DO doer (action-name
   args)) cond))
        if KB_self |= cond
                list←((Sense Agd (action-name doer args)), 3);
        else
                list←(¬(Sense Agd (action-name doer args)), 3);

    return list;
}
```

Fig. 4.6. An Algorithm of Reasoning Others' Observabilities.

4.4.6. Update: Maintaining Belief Consistency and Compatibility

When new information is to be asserted to KB_{self}, it may inconsistent or incompatible with old ones. The function *update*, shown in Fig. 4.7, manages history and is responsible maintaining for consistent and compatible beliefs in KB_{self}. The obvious assumption of this algorithm is that what is not changed during update is assumed to stay the same, i.e. persistence. Since the number of time steps could be infinite, self keeps only current beliefs in KB_{self}, except that the most recent one is kept, even if it is not generated currently. Therefore self still believes some information,

even though self does not infer it from KB_{self} or infer it from last action or being told it by others.

Belief consistency and compatibility are maintained from two perspectives. If the assertion is a positive literal, it will be asserted to KB_{self} if it is not already there; implying that the negated literal derived from the Closed World Assumption would be overridden by the addition of positive literal. Also all information which is incompatible with the assertion is retracted. If the assertion is a negative literal, the positive literal (if any) will be retracted from KB_{self}.

```
update(KB_self, info){
    if info is a positive literal p
        if KB_self |≠ p
            assert(KB_self, p);
        ∀q ∋ incomp(p, q)
                retract(KB_self, q);
    else //info is a negative literal ¬p
        if KB_self |= p
            retract(KB_self, p);
        ∀q ∋ incomp(¬p, q)
            retract(KB_self, q);
}
```

Fig. 4.7. A Belief Update Algorithm.

4.5. OBPC: Observation-Based Proactive Communication

The information worth exchanging comes from analysis of agents' goals (e.g. preconditions of plans or actions that the agents are going to perform). If they do not

know a precondition, they cannot act. Therefore telling them proactively or they actively asking for it improves efficiency.

Proactive Communication answers the following questions pertinent to agent proactivity during teamwork. First, when does an agent send the information to its teammates if it has a new piece of information (either from performing an action or observing)? A simple solution could be sending the information when requested. That is, the agent would only send the information after it has received a request from another agent. In our approach, the agent observes its teammates and commits to proactive tell once it realizes that one of the teammates needs the information to fulfill its goal and does not have it now. Meanwhile, if the agent needs some information, it does not passively wait for someone else to tell it; it asks for this information actively.

Second, what information is sent in a session of information exchange? There are three kinds of information that can be communicated. One is the information explicitly needed by an agent to complete a given plan, i.e., conjuncts in a precondition of plans or actions that the agent is going to perform. The second is the information implicitly needed by the agent. For example, if agent *a* needs predicate p and knows p can be deduced from predicate q, even if the providing agent does not know p, it still can tell agent *a* about q once it has q, because it knows that agent *a* can deduce p from q. The third includes the information for synchronization among team members performing actions and joint actions. This research, however, deals only with agents communicating information that is explicitly needed.

We developed two observation-based communication protocols: *O-Tell* and *O-Ask*. These protocols are used by each agent to decide whether to generate inter-agent communication when information exchange is desirable. The *O-Tell* and *O-Ask* protocols are based three types of knowledge.

The first is information needers and providers. In order to find a list of agents who might know or need some information, we use information flow developed by DIARG [137]. DIARG infers the potential information needs by reasoning about the goals of the other agents based on the team plan used by the agents. To be specific, it analyzes the preconditions and effects of actions and plans and generates information flow which is a list of needers and a list of providers for every piece of information. An agent is a provider for the effects of any action/plan it is capable of performing. An agent is a needer for the precondition of any action/plan it needs to execute.

The second is beliefs generated after observation. Agents take advantage of these beliefs to track other team members' mental states and use beliefs of what can be observed to reduce the volume of communication. For example, if the provider believes that the needer senses *I*, the provider will not tell the needer; if the needer believes that a specific provider has *I* by observing the action performed by the provider, the needer will ask this provider, rather than ask all of them.

The third is beliefs inferred from the effect of the action performed by the agent. The agent will tell these beliefs to the needer if it believes that the needer does not sense them.

Algorithms for deciding when and with whom to communicate for *O-Ask* and *O-Tell* are shown in Fig. 4.8.

Considering the intractability of general belief reasoning [47], our algorithm deals with beliefs nested no more than one layer. Also the algorithm involves only two parts, i.e., sender and receiver. It does not consider the third party communication such as agent *a* asks *b* to ask *c* for some information. Therefore, the belief about if another agent senses an action executed by a third agent is not included. The algorithm is sufficient, thought, for our current study on proactive behaviors of agents, which focuses on peer-to-peer proactive communication among agents.

```
/*O-Ask will be independently executed by each agent (self) when it needs the
   value of information I.
*/
O-Ask(self, I, KB_self){
    if KB_self |≠I
        if ∃ Agp≠self, ϕ∈action ϶ (KB_self |=(ϕ Agp args))∧(I∈ Prec(ϕ)∨I∈Efft(ϕ))
                ask Agp for I;
        else randomly select a provider
                ask the provider for I;
}

/* Independently executed by each agent (self), after it observes I or produced I
   as effect of an action.
*/
O-Tell(self, I, KB_self){
    ∀Agn ∈ needers
        if KB_self |≠ (Sense Agn I)
                tell Agn I;
}
```

Fig. 4.8. Observation-Based Proactive Communication.

For *O-Ask*, the needer requests the information from a provider who may know it. This provider may be explicitly determinable if its action that determines I is observed by the needer. If such agent cannot be found, the needer randomly chooses a provider from the provider list and asks the provider for *I*.

For *O-Tell*, the provider tells the agents who need *I*. The needer(s) is(are) determined from the information flow. The provider's beliefs about the needer's sensing capabilities become the basis for this reasoning. The provider will tell *I* to the needer only if the provider does not believe the needer can sense *I*. The implication here is that communication will not go to the needer whom the provider believes can sense *I*. By this means, the communication load can be reduced by an agent's belief about another agent.

4.6. Summary

This section has presented an approach to dealing with agent observability for improving performance and reducing inter-agent communication. Each OP-CAST agent is allowed to have some observability to sense the environment, and to watch what others are doing inside its detection range. Based on the observation, the agent updates its knowledge base and infers what others may sense at the current time. Reasoning about what others can sense allows agents to decide whether to distribute information to others. We have proposed a proactive communication mechanism to confer some advantage to related team members for realizing team interaction and cooperation proactively also.

CHAPTER V

DYNAMIC INFORMATION PREDICTION

5.1. Motivation and Overview

To decide a communication policy (such as *O-Ask* or *O-Tell)*, OBPC adopts the same method as the original CAST approach. It requires a domain expert to publish frequencies associated with information production or information need, and defines inflexible decision-making rules. Thus information with some specific frequency will be *O-Tell*ed and others will be *O-Asked* (see Section 4.5). Moreover, it only looks at the general frequency, i.e., for a piece of information, in spite of how many agents produce it or need it, there is only one frequency related to the information production or the information need. Obviously, this method is too rigid to handle dynamic and complex situations.

We develop a more general way, called Dynamic Information Prediction (DIP), to deal with frequencies of information production or information need. For a piece of information, we take each needer and provider into account separately, and predict time points at which production or need occurs. Rather than relying on a domain expert to input such knowledge, DIP anticipates distributions of information production or need dynamically, by utilizing previous data about information production or need.

We assume the time intervals for the production or need for a piece of information are random according to some unknown distributions. We also assume needers and providers keep a record of their own information production or need time

intervals. By acquiring history from others, the needer can estimate the distribution of time intervals during which an item of information will be produced by a given provider. Similarly, the provider can estimate the distribution of time intervals during which an item of information will be needed by a given needer. Agents make such estimations dynamically, by analyzing the trail of the list of time intervals.

There are two statistical approaches to describe a probability distribution: parametric and non-parametric. The parametric approach utilizes a certain formula to model the probabilities. In some domains, obtaining an accurate model of a distribution requires complex knowledge acquisition from domain experts, or a complex learning process on the part of the agent. Hence, from a practical point of view, the parametric approach may be too complicated to support efficient online inference. Alternatively, the non-parametric approach does not require knowledge of how the probabilities are distributed. It assumes that the sampling distribution of collected data is analogous to the population distribution. This feature allows the Empirical Distribution Function (EDF) [15] to be used to approximate the distributions of information production and need.

To sum up, the idea of DIP is to gather previous data on information production or information need opportunistically and use EDF to approximate their distributions. The following sections first introduce our rationale of choosing EDF, and then the mechanism of EDF, and considerations which make DIP applicable.

5.2. Considcertations of Statistical Models

To predict the time at which an agent produces or needs a piece of information, we need to model the time interval X between the time at which the agent last produced or needed the information and the time at which the agent will produce or need the information next. In our framework, X is measured as the number of steps taken by the agent, and therefore is a discrete variable.

Modeling this probability distribution brings a challenge. In principle, this distribution can be arbitrarily complex, and its structure may vary enormously from domain to domain, and even from information to information within the same domain. We first need to decide which approach, parametric or non-parametric, is more suitable to our problems and assumptions.

Choosing an appropriate statistical model always depends on the assumptions we make about the variables and the objectives we wish to achieve. Some source distinguishes parametric and nonparametric on the basis that parametric make specific assumptions with regard to one or more of the population parameters that characterize the underlying distributions for which the test is employed, while nonparametric makes no such assumptions about population parameters [110].

Although we may approximate the discrete time variable using a continuous distribution, we restrict our discussion to discrete distributions here. Our problem is to model the time interval. Among commonly used discrete distributions, it is difficult to find one which fits into our problem directly. For instance, the Poisson distribution is often applied to counting the number of events in a certain time period, but not the time

interval itself. It also requires that the mean of the distribution equals the variance, which discourages the uses of Poisson distribution under many practical settings [98]. Although more complicated models may exist to approximate the distribution of a time interval, we want a direct approach, given the main objective of our study.

An alternative non-parametric approach may accommodate a weaker assumption on how the random variable X is distributed. By using a non-parametric approach, we do not need to restrict the data to a specific family of distribution, but assume the sampling distribution of collected data to be analogous to the population distribution. For example, for an unknown distribution of P(X), if we have no other information, the entire sample will be the best estimate of the population as long as the current samples are randomly generated from P(X) [15].

In our problem, we do not know much about X's distribution and cannot make any distributional assumptions based on the current knowledge. Hence a non-parametric approach can be applied, where we use the current sample set to approximate the true distribution. We propose to use the EDF [15] of X to approximate X's distribution. The only assumption we make is that the sampling distribution of collected data is analogous to the population distribution.

5.3. Empirical Distribution Function

As will be shown in Chapter VI, given models of the information produced or needed that the agents experience during teamwork, the problem of information prediction is to determine the probabilities that an agent produces or needs an information item at a certain time point, or more precisely, the probability of such

production or need before that time. This corresponds to the cumulative probability that the information is produced or needed at certain time, $Pr(X \le t_j - t_{j-1})$, where t_j is the certain time point and t_{j-1} is the last production or need time which is known. However, as will be seen in Chapter VI, under some circumstances t_j is known and in others it is not. In the former case, the cumulative probability may be calculated directly. In the latter, one simple approach is to calculate the expected value of t_j as an intermediate step. Then, the cumulative probability can be estimated from the expected value, $Pr(X \le E(X))$, where $E(X)$ is the expected value of $X = t_j - t_{j-1}$. However, $Pr(X \le E(X))$ is just a simple approximation to $Pr(X \le t_j - t_{j-1})$. A more accurate approach is to utilize the law of total probability, which is described by the formula as follow:

$$Pr(X \le t_j - t_{j-1})$$

$$= \sum_{\tau=t_{j-1}}^{\infty} Pr(X \le t_j - t_{j-1} \mid t_j = \tau) \times Pr(t_j = \tau).$$

This is the approach we take to calculate $Pr(X \le t_j - t_{j-1})$ when t_j is unknown (refer to Appendix A for calculation details). In this section, we focus on how to determine the underlying distributions for X.

Let $\{X_1, \ldots, X_k\}$ be a collected sample, then the Cumulative Density Function (CDF) of X is estimated by [24]:

$$CDF(X) = \frac{\#\{X_i : X_i \le X, 1 \le i \le k\}}{k},$$

and Probability Mass Function (PMF) of X is estimated by [24]:

$$PMF(X) = \frac{\#\{X_i : X_i = X, 1 \le i \le k\}}{k}.$$

For example, suppose there are 10 (then k = 10) sample data {20, 30, 24, 33, 24, 40, 33, 30, 33, 24}, among which the number of 20 is 1, the number of 24 is 3, the number of 30 is 2, the number of 33 is 3 and the number of 40 is 1. Then CDF(24) = $\frac{1+3}{10}$ = 40% and PMF(24) = $\frac{3}{10}$ = 30%. Fig. 5.1 shows CDF(X) and PMF(X). In our use of these formulae, they will be updated from time to time (described in Section 6.9) as additional data samples become available. Thus, we expect the distributions to become more accurate over the time in which they are used.

Note that there are many value of X such that PMF()=0 in this example. This effect occurs because of a small sample size. One can deal with this situation in a couple of different ways. First, one might apply a smoothing function to the distribution and then use the smoothing function to estimate the probability for a given value of X. For example, the method called cubic spline uses a series of unique third degree polynomials to fit between sets of m points, m≥2, of the whole data points, with the constraints that the curve obtained is continuous and appears smooth [3]. Alternatively, one could simply use the discrete CDF and PMF produced by the equations above. The latter becomes increasing accurate as the number of samples increases. Whether one uses a smoothing function or not is irrelevant to the research discussed here. We simply use the fact that approximating CDF's and PMF's can be calculated from time to time as increasing history is accumulated. For simplicity, in our experiments, we just use the raw CDF and PMF produced.

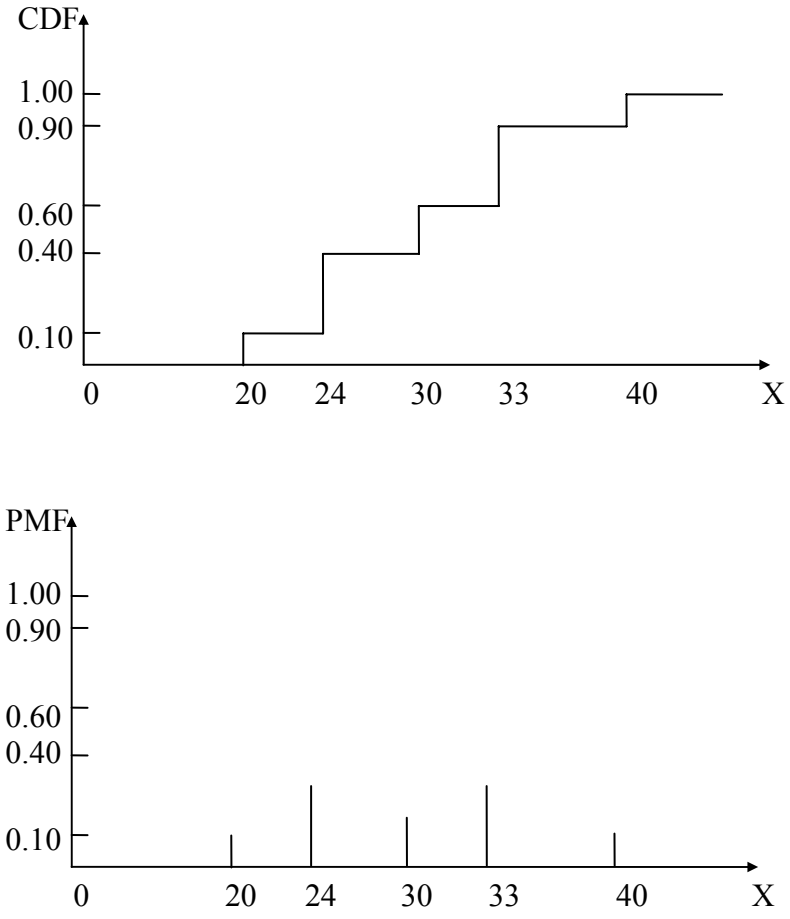


Fig. 5.1. An Example of Using EDF.

Based on the EDF, we can estimate the probability of the information production at a certain time t_p and the probability of the information need at a certain time t_n. For each of these, this approach can be formulated as:

1. Initially collect a small amount of data x_1, ..., x_k.
2. Calculate CDF(X).

3. When an additional value of X is collected and added to the data pool, CDF(X) will be adjusted when this process goes on.

When this process is iterated for many times, the collected data will be closer to "true" (or population) distribution and the prediction will be more accurate[12].

Ultimately, the probability distributions computed by EDF will be used to calculate utility when some time parameters in the utility function are unknown (the utility is introduced in Section 6.4).

5.4. Data Acquisition

Applying EDF raises three questions: 1) What kind of previous data do agents want to gather and how to initialize the system? 2) How does the agent acquire previous data? 3) By which format the previous data will be conveyed?

5.4.1. Source of History and System Initialization

Agents can gather previous data on information production- and need-time intervals in various ways. They can use the data provided by domain experts, or historical data collected dynamically during the execution of a plan. Agents must expend extra effort, thus acquiring knowledge from the domain experts, to obtain the data by the first way. These efforts limit the system's dynamic capabilities; therefore, we use the historical data, which can be gathered dynamically during the teamwork process.

[12] A famous application of this theory in statistics is so called bootstrapping method, which makes statistical inference based on the re-sampling of current sample set [93].

However, there may be a problem when using EDF at the beginning of teamwork, because agents have no sample data. One way of dealing with this problem might be to generate random numbers to initialize these times. However, this solution lacks regularity, so it may be impractical. Another way might be to develop a rule to guide agents at the beginning. For example, the rule could be that the provider and needer are obligated to communicate with each other and attach their historical data at the first several rounds of production and need rather than using DIP to predict needs and productions. However, this solution lacks flexibility and may create many messages which against the major goal of Proactive Communication.

The solution we adopt is to run the system in a trial mode. We will collect data from previous test runs and use these data to initialize time intervals. By this way, agents are able to predict time points of productions or needs since the system starts. Gradually the initialized data will be extended and the prediction will be based mostly on the data from actual run.

5.4.2. Acquisition of History

Under the approach taken here, agents need to have a history of the production- or need- time intervals of others, so they can estimate distributions of the production or need. This raises the question of how to obtain these time intervals. Empirical data [134] show that the cost of a message may be approximated by C+K**size*, where C is a base cost of sending a message, *size* is the message size in bytes, and K is a parameter which adjusts the effect of the message size to the message cost. Typically C is much larger than K and the cost for a small amount of additional information is almost

negligible, allowing the agents to attach historical information on their production- or need-time intervals to each message sent to the receiver. This suggests that the historical data can be sent opportunistically along with the information exchanged between providers and needers. We propose that needers attach a history of information need time intervals to every message they send; similarly, that providers attach a history of information production time intervals to every message they send. Agents certainly will not attach the whole history to every ask or tell message. They only attach time intervals from the last sent to the receiver to the latest one. These historical data will allow them to approximate the probability of a piece of information being needed or produced at certain time point by a specific needer or provider.

5.4.3. Message Format to Convey History

The message format is based on Knowledge Query and Manipulation Language (KQML) [37]. The syntax of KQML is based on a balanced parenthesis list. The initial element of the list is the performative and the remaining elements are the performative's arguments including message content, sender, receiver, and historical data of information production or need. An example message *ProactiveTell*ed about a wumpus w1's location from the carrier Ca to the fighter F1 looks like this:

((performative *ProactiveTell*)
(message (location w1 27 58))
(sender Ca)
(receiver F1)
(data (27 38 40 33 47 39)))

The last pair in the example list contains lengths of time intervals of producing wumpus' location information from the last one sent to the fighter to the most recently generated one. The sender maintains records of the last sent data regarding each of the receivers. The receiver will update its data list after receiving a new message.

5.5. Important Issues

Two issues require consideration to make DIP complete. In the following, we analyze them and propose some possible solutions. In Section 7.2.4, we investigate them further and apply some specific algorithms to experiments.

5.5.1. Preventing the Provider from Having History Starvation

During the teamwork process, it is possible that the provider may cease to receive need history updates, which occurs when the needer stops asking for the information. This case results from the essence of proactivity — agents always assist each other proactively, rather than passively waiting to be asked. For example, when proactivity is fully enabled, the needer may increasingly depend upon receiving information from proactive tells and the number of ask messages gradually decrease to zero, in which case the provider would receive no new historical data on need times. However, the EDF approach depends on the sample data to increase the accuracy of the approximation. If the provider ceases to receive historical data, the CDF and PMF will not be adjusted. In order to ensure that the EDF mechanism continues to function, we set a time threshold for the provider and for the needer. If neither hears from the other within the threshold, it must communicate with the other and attach its historical data.

5.5.2. Preventing Communication Deadlock

Deadlock will result if both needer and provider wait indefinitely for communication from the other. Since when needing or producing an information item, an agent may either contact the other or wait for the other to contact it (see Section 6.2 for detail), there is a possibility that they both decide to wait. In investigating this case, we found it similar to the history starvation case; thus, if agents do not get any information from others in a long time, we adopt the same approach of using a time threshold to prevent deadlock.

5.6. Summary

In DIP approach, agents are able to utilize previous data about information production and need. In order to provide CDF and PMF necessary for estimating the values of the utility function, we suggest transmitting data on the times of information production or need along with any messages that are sent among agents, and then using EDF methods to approximate CDF and PMF. The distributions calculated can help agents make better communication decisions in two ways: first, agents can proactively tell information to agents if they expect another agent to need it in the near future, thereby reducing the number of asks; second, agents can ask for needed information actively from specific providers if they do not expect a proactive tell in the near future.

CHAPTER VI

DECISION-THEORETIC PROACTIVE COMMUNICATION

6.1. Motivation and Overview

As one has seen, for generating Proactive Communication, the *relevance* of information to another agent can be inferred by reasoning about plans and actions of the other agent. Specifically, preconditions of these plans and actions constitute the information relevant to the other agent, which is needed by the agent in order to execute its plans and actions.

However determination of relevance is only one part of the requirement of Proactive Communication. The other part is to decide whether or not to proactively tell or actively ask for the information. Making communication decisions is difficult in that agents have different knowledge, and there are always unknown things existing on the needer's and the provider's sides. We develop agents' communication decision-making based on estimation of the probability distribution introduced in last section Dynamic Information Prediction. The purpose of predicting information production or need time is to help agents decide whether or not to send messages to active needers or providers, rather than always sending to them. An agent becomes active when it is selected to participate in executing a team plan. However, in order to know which agents are active, the original CAST sends a significant number of extraneous messages to maintain the shared mental model, where agents share execution states of teamwork processes [11]. And, when there are multiple active providers, a needer will repeatedly

ask them until it gets a reply [137]. The drawback is that the needer must wait for a reply if the one being asked does not have the information at that moment. Also, even if another provider has told the needer proactively (after the ask), the one asked still needs to reply when it has the information, since it doesn't know about the proactive tell. Hence the communication either takes longer or there is too much of it. Moreover, the original approach obeys a rigid rule, which says that information, which is needed more frequently than being produced, must be told proactively; otherwise it must be asked for actively. This rule explicitly states two policies, proactive tell and active ask, the original approach uses. It also implies a policy wait, i.e., for some information, agents have to wait until asked or told. However, information should not be communicated in such a strict way, and more options are desired to match reality.

Additionally, communication involves more complex issues not covered in the original CAST approach. First, communication can be valuable if it assists agents with timely and the newest information; it also carries communication cost and risks such as those in a hostile environment. Not modeling this value, cost and risk will limit our application on most practical systems. Hence, communication should be subjected to careful cost-risk-value analysis. Second, since an agent is a member of the team and it will accomplish the plans with other team members, its utility depends not just on its own communication decision, but also on the decision of its teammates. To reconcile decision interactions in the team, an agent should have a method of estimating others' decisions and considering how these decisions impact its own. Third, information changes dynamically in the environment, and the degree of use of the information may

be different too. For some information, agents must consume all changes (e.g., new enemy target identified), while for other information agents do not necessarily have to process each change (e.g., current location of friendly aircraft). Agents need to check every production of the first type information, while the check to the second type information depends on agents' needs. A more comprehensive solution must be developed to deal with different types of information.

In a word, a better communication solution is desired to reduce unnecessary messages between needers and providers, and to make decisions under uncertainty according to cost, risk and value of information the communication convey.

We propose Decision-Theoretic Proactive Communication (DTPC), by which agents communicate in an optimal way using a decision-theoretic approach. The decision-theoretic approach concentrates on identifying the "optimal" policy [93], where the notion of "optimal" has a number of different meanings, the most common of which is "that which maximizes the utility," in this case, of communication. We incorporated cost, risk and value into the decision-making and expand a set of policies that needer and provider will use. The decision-making generally involves computing the cost, risk and value of each policy, and choosing the one with maximum utility. Moreover, since communication involves needer and provider, and they keep interacting with each other during the teamwork process, their decisions may be interdependent. When making decisions, it is necessary for them to take the decisions of their counterparts into account and communicate in a way benefiting the team. These features are bases of DTPC.

By DTPC, agents are equipped with a set of communication policies from which they must choose when making decisions. To quantify agents' decisions, we have developed a generic utility function that focuses on representing the information production and need of team members. After evaluating the utility of each policy, agents will identify the optimal policy, which maximizes the utility of communication.

Two difficulties exist in agents' decision-making. First, agents cannot compute exact values of the utility since some parameters cannot be known precisely. Hence, they calculate the utility function by using estimated values of these parameters. Second, agents' decision-making is interdependent, so when evaluating a policy, agents must consider their counterparts' decisions, which also need to be estimated.

The following sections first define policies and time points for different situations of decision-making, followed by introductions to a generic utility function and multi-agent communication processes and finally a set of algorithms which handle the communication processes.

6.2. Policies and Time Points

We made the following assumptions about information production and need.

- The time interval from one production of information *I* to the next is a discrete random variable.
- A needer does not generate a new need for *I* until after the previous need has been satisfied.
- The time at which a need occurs is the time at which the transition from no need for *I* to a point of time where there is such a need is made.

- The needs will be continued in the time interval through which the needer waits for a value to be obtained either by observation or being told by the provider.
- If there is a need and a newly produced value is received, the value is immediately used.
- Once a value for *I* is used by a needer, it may not be reused.
- The time interval from the satisfaction of one need to the occurrence of the next need for *I* is a discrete random variable.

The provider will face two situations when making decisions. In situation PA, it produces a new piece of information. In situation PB, it receives a request for a piece of information. The needer also has two situations to consider. In situation NA, it needs a piece of information. In situation NB, it receives a piece of information, which may be either a reply from the provider whom it has actively asked or a proactive tell from the provider.

In order to make their communication decisions at each situation, agents need to consider the relationship between the time at which information is needed and the time at which it is produced. The various policies involve using the information produced at different times or satisfying needs at different times. Thus, to describe the range of possibilities encompassed by the different communication policies adequately, several different points in time must be defined. For clarity, we define sets of policies and sets of relevant points in time, for the needer and for the provider in different

situations. Any time an agent makes a decision, we assume it only chooses one policy and acts accordingly.

For clarity, we assume the system consists of two agents. This is agent *a,* a provider, and agent *b,* a needer.

6.2.1. Situation PA: Provider Produces a Value for *I*

Fig. 6.1 shows situation PA. Let $T^0_{a,P}$ be the time at which agent *a* produces a value for *I*; we consider this to be the current time. Also let $\{T^1_{a,P}, T^2_{a,P}, \ldots\}$ denote the (ordered) set of times at which agent *a* will produce *I* in the future, which are unknown at the current time. There is additional information available to the provider. The provider knows the times of each of the values it has produced and which of these were sent to the needer. Let $T^{ls}_{a,P}$ be the time of the last value for *I* the provider sent to the needer. Let $T_{b,N}$ be the time of a need after $T^{ls}_{a,P}$, which is (probably) unknown to the provider, too. It is possible, though, that some of the values for *I* the provider sent were unused. It is also possible that, in the time interval $(T^{ls}_{a,P}, T^0_{a,P}]$, the provider produced one or more value for *I* and did not send them to the needer. If there are any such values, let $T^{ns}_{a,P}$ be the largest time at which the provider generated a value for *I* that it did not send. The ordering constraints of these time points are as follows:

$$T^{ls}_{a,P} < T^{ns}_{a,P} < T^0_{a,P} < T^1_{a,P} < T^2_{a,P} < \ldots,$$

$$T^{ls}_{a,P} < T_{b,N}\,.$$

For these time points unknown, as illustrated in Fig. 6.1, $T_{b,N}$ could be any time point after $T^{ls}_{a,P}$ and $T^{1}_{a,P}$ could be any time point after $T^{0}_{a,P}$.

These time points will be used in utility function. To deal with the uncertainty brought by the unknown time points, we take advantage of two kinds of knowledge. First we know their low bound. For example, $T^{1}_{a,P}$ must be greater than $T^{0}_{a,P}$ and $T_{b,N}$ must be greater than $T^{ls}_{a,P}$. Second, we can approximate their distributions by EDF approach. By this way, we can deal with uncertainty.

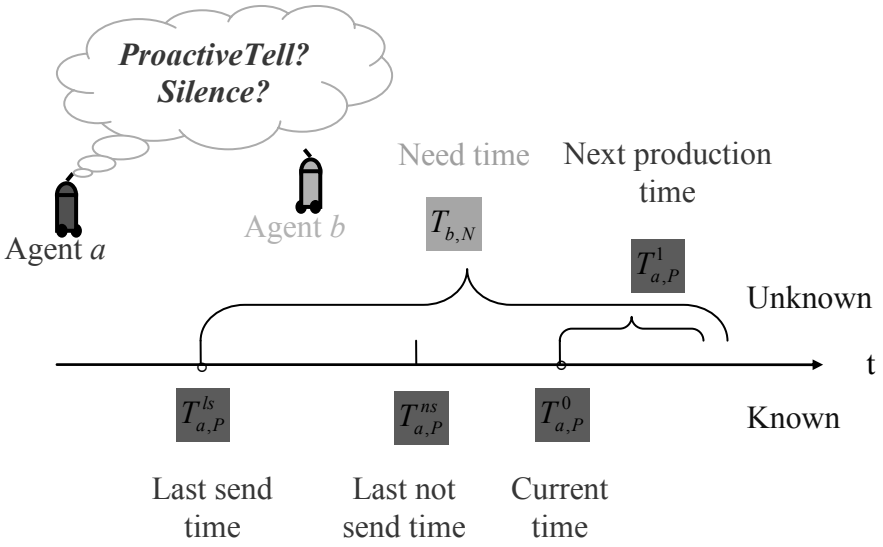


Fig. 6.1. Situation PA: Provider Produces *I*.

Agent *a* has two policies to choose on this situation:

ProactiveTell: The agent proactively provides *I*;

Silence: The agent does not provide *I*.

Agent *a* will either *ProactiveTell* the value just produced at $T^0_{a,P}$ to agent *b* or keep *Silence*. The difficulty of decision-making is that agent *a* may not know the exact time when the needer's need arises, i.e. $T_{b,N}$. Therefore, if agent *a* decides to *ProactiveTell* the just-produced value for *I*, then the value provided may not be the most current because new values may have been produced by the time agent *b* really needs *I*. However, if agent *a* decides to keep *Silence*, agent *b* may be unable to get the information in time if a need was already raised. Therefore, timeliness and currency[13] of the information provided should be a major consideration of decision-making.

6.2.2. Situation PB: Provider Receives a Request about *I*

Situation PB is shown in Fig. 6.2. Let $T_{b,q}$ be the time at which agent *b* requests *I*; we consider this to be the current time. Let $T^{q0}_{a,P}$ be the latest production time, before the request time $T_{b,q}$, and let $T^{q1}_{a,P}$ be the next production time, which is unknown, following the request time $T_{b,q}$. Obviously, the order of these time points is:

$$T^{q0}_{a,P} \leq T_{b,q} < T^{q1}_{a,P}.$$

[13] While timeliness and currency sound somewhat similar, there is a distinct difference. Timeliness refers to the delay between a need arising and the need being fulfilled, while currency refers to whether one is using current or old information.

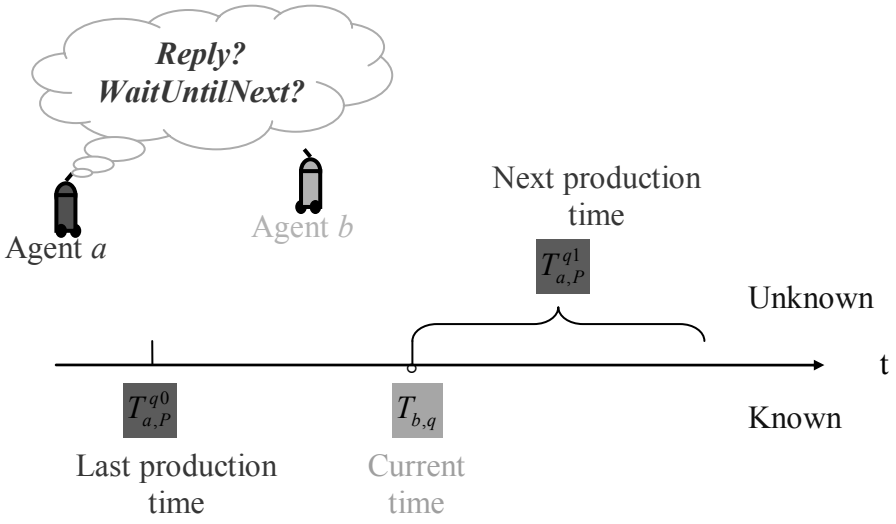


Fig. 6.2. Situation PB: Provider Receives a Request about *I*.

Agent *a* has two policies to choose on this situation:

Reply: The agent provides most recent value for *I*;

WaitUntilNext: The agent waits until next production of *I* and then provides *I*;

Agent *a* will either *Reply* the value last produced at $T_{a,P}^{q0}$, or *WaitUntilNext* production time $T_{a,P}^{q1}$ and reply the value produced at $T_{a,P}^{q1}$. The major consideration is still the timeliness and the currency of information provided. The last produced value is timely but may lost currency if a new value will be produced soon, while the new value may not be timely. Again, the utility function should address these issues.

6.2.3. Situation NA: Needer Has a Request about *I* Arise

Situation NA is shown in Fig. 6.3. Let $T^0_{b,N}$ be the time at which agent *b*'s most recent need for *I* arises; we consider this to be the current time. Let $T^{a0}_{a,P}$ be the time at which agent *a* most recently produced *I* and $T^{a1}_{a,P}$ be the time at which agent *a* produces *I* next. Let $T_{b,r}$ be the time at which agent *b* most recently received a value for *I* from agent *a*. The relationships among these points are as followings:

$$T_{b,r} \le T^{a0}_{a,P} \le T^0_{b,N} < T^{a1}_{a,P}.$$

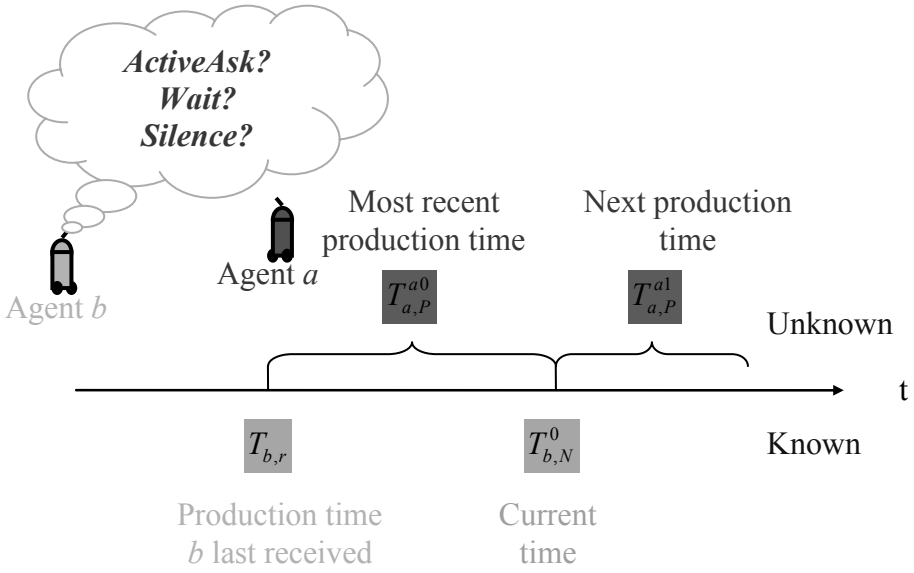


Fig. 6.3. Situation NA: Needer Has a Request about *I* Arise.

Agent *b* has three policies to choose on this situation:

Silence: The agent does not ask for *I* and uses the most recent value it has;

ActiveAsk: The agent actively asks for *I*;

Wait: The agent waits to be told *I* proactively.

If agent *b* *ActiveAsk*s for the information, the information it uses is always the most recent. But this process costs more messages and brings high risk. If the needer *Waits*, it cannot get the timely information. If the needer keeps *Silence*, i.e., it uses the value last received at time $T_{b,r}$, this value may not be the most recent because it may be changed during the time interval $[T_{b,r}, T_{a,P}^{a0}]$. Again, these considerations must be included in utility function.

6.2.4. Situation NB: Needer Receives *I*

Situation NB is shown in Fig. 6.4. In this situation, agent *b* receives a piece of information *I*, which may be either a reply from agent *a* whom it has *ActiveAsk*ed or a *ProactiveTell* from agent *a*. Agent *b*'s choice is deterministic, thus *Accept*s *I*. However, agent *b* may use *I* or not. If *I* is a reply to the *ActiveAsk* sent by agent *b*, agent *b* definitely will use *I*; if *I* is sent by the *ProactiveTell*, use of *I* happens if agent *b* is *Wait*ing for *I* or agent *b* will hold *I* and use it later with a *Silence* decision. Meanwhile, agent *b* will neither notify agent *a* about the acceptance of *I* nor the use of *I*. These uncertainties and unknown knowledge bring more difficulties to agent *a*'s estimation of the time of the need.

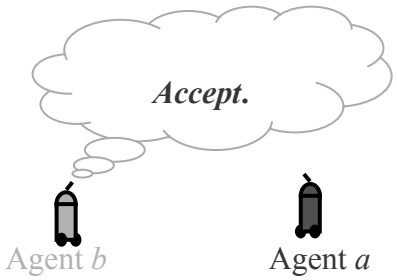


Fig. 6.4. Situation NB: Needer Receives *I*.

In summation, Table 6.1 lists communication situations which an agent will face and policies which are available for these situations.

Table 6.1. Situations and Policies.

Situation	Policy
PA: provider produces *I*	*ProactiveTell* *Silence*
PB: provider receives a request for *I*	*Reply* *WaitUntilNext*
NA: needer has a need for *I* arise	*ActiveAsk* *Silence* *Wait*
NB: needer receives *I*	*Accept*

6.3. DTPC Model

Part of decision-making is to evaluate each of the possible policies for each of the situations described above. We need to generalize the notion of situation slightly to

include the specific item of information *I* with respect to which it occurs. Let t be the time of occurrence of a situation, we denote a situation by S_t = (SU, *I*) for SU ∈ {PA, PB, NA, NB} and *I* ∈ set of information items. The policy, denoted δ, used to respond an situation S_t is also relevant. There are two time points closely related to information need and production: t_n the time at which a need for *I* occurs, and t_p the production time of a value for *I* which is provided for the need at t_n. Because obtaining *I* may involve sending messages, these messages are also part of our model. Then, letting E denote a finite set of states, we define our model DTPC as:

$$DTPC = <E, \{S_t\}, \{\delta\}, \{t_n\}, \{t_p\}, M, U>$$

where:

- E = {e} is a finite set of states. Each state e = (ε_a, ε_b) where $\varepsilon_j \in E_i$, i = {*a*, *b*}, are the local states of the corresponding agents.
- $\{S_t\}$ is the set of possible situations occurring at t.
- {δ} is a finite set of policies.
- $\{t_n\}=\mathcal{N}$ (the natural numbers) is the set of times at which a need may occur.
- $\{t_p\}$=N is the set of production times of *I* which are provided for the needs at $\{t_n\}$.
- M is the set of messages. A special message that belongs to M is the null message which is denoted by φ. This message is chosen by an agent that does not want to communicate to the other agents.
- U is a utility function assigning a value to the use of a specific policy δ.

6.4. Utility Function

We assume the information needer and provider have the same utility function; because as they are cooperative in a team, we consider the utility function to represent the utility gained by the team when a particular needer uses a particular item of information at a particular time. Consequently, needers and providers have the same utility function. However, because the needer and provider each evaluate the utility function based upon their individual knowledge, they are likely to obtain different values for it. In terms of our decision theoretic approach, we must therefore consider the evaluation of the utility function separately for the needer and provider.

6.4.1. Defining the Utility Function

The utility function is a mapping from agent's internal states, the current situation, a communication policy associated with the situation, time points t_n and t_p and messages in M to a real number:

$$U: e \times S_t \times \delta \times t_n \times t_p \times \{m\} \rightarrow \Re$$

where $\{m\}$ is the set of messages used by the policy δ.

Three terms are generally included in the utility function. The first is values gained from the information delivered. The uncertainties existing in the environment normally lead to the unfixed time durations of information production and need, which further lead to uncertainty in the value gained by communicating the information. The second term is the cost of sending a message. The third is the potential communication risk due to things like unwanted revelation of the information, such as being overheard by an enemy on a battlefield. We define utility as the difference between the value

gained by having the information, and the cost of sending the messages and the risk of communication:

$$U(e, S_t, \delta, t_n, t_p, \{m\})$$

$$= V(S_t, \delta, t_n, t_p) - C(\{m\}) - R(e, S_t, \delta).$$

where V denotes a value function, C denotes a cost function, and R denotes a risk function.

6.4.2. Identifying Information Production and Need Time in the Utility Function

In the utility function U, for a given decision point involving situation S_t, the parameters e and {m} are fixed, when U is evaluated for a specific policy δ which is associated with S_t. On the other hand, parameters t_n and t_p vary for different policies and either or both of them may be unknown at the current time t.

In Table 6.2, we identify t_p and t_n for different policies. P and N indicate policies for needer or provider; for example, P.*ProactiveTell* denotes provider's policy *ProactiveTell*. T_o is a time cutoff which replaces the length of the time an agent may wait in case the communication deadlock (introduced in Section 5.5.2) occurs. Based on this, later in our algorithm of decision-making, a needer and provider pair is forced to communicate if they kept silent for a period of time T_o (see Section 6.8).

For some policies, the time point used for a parameter depends upon the counterpart's response, which is unknown to the decision maker. For example, if a needer *ActiveAsk*s for *I*, it does not know whether the provider will *Reply* or *WaitUntilNext*; if the needer *Wait*s, it does not know whether the provider will keep *Silence* or *ProactiveTell*. The needer, therefore, needs to find a way to estimate which

policy the provider will choose and how this choice impacts t_p. To solve this problem, agents are asked to think as their counterparts do. We assume that both needer and provider know the other's possible policies and estimation process. When making a decision, each will go through the estimation process of its counterpart to identify the policy the counterpart will choose. This process, of course, will be based on each one's own information. The time point of the estimated policy will be used to fill in the parameters in Table 6.2.

Table 6.2. Identifying Parameters for Policies.

Policy	Parameter	
	t_n	t_p
P.*ProactiveTell*	$T_{b,N}$	$T^0_{a,P}$.
P.*Silence*	$T_{b,N}$	A production time between [$T^0_{a,P}$, $T^0_{a,P}$+T_o].
P.*Reply*	$T_{b,q}$	$T^{q0}_{a,P}$.
P.*WaitUntilNext*	$T_{b,q}$	$T^{q1}_{a,P}$.
N.*Silence*	$T^0_{b,N}$	$T_{b,r.}$
N.*ActiveAsk*	$T^0_{b,N}$	$T^{a0}_{a,P}$, if *a Reply;* $T^{a1}_{a,P}$, if *a WaitUntilNext.*
N.*Wait*	$T^0_{b,N}$	$T^{a1}_{a,P}$, if *a ProactiveTell;* A production time between [$T^0_{b,N}$, $T^0_{b,N}$+T_o], if *a Silence.*

The precise estimation of values for the time points varies from one situation and policy to another. These will be developed in conjunction with the development of the evaluation formulae for the different situations and policies.

In the next two sections, we define the cost and the value functions in the utility function. Particularly, we focus on analyzing t_p and t_n which are given values by the time points listed in Table 6.2, and the use of the distributions of t_p and t_n computed by EDF introduced in Section 5.2. We defer the risk function to the implementation, because risk normally is domain-dependent. For example, in the Multi-Agent Wumpus World, the risk of communication depends on the distance between wumpus and information sender and hearing radius of wumpus (refer to Section 7.2.3.1 for detail).

6.5. Cost Function

In cost function C, policy δ determines the number of messages to be sent. For example, policy *ProactiveTell* indicates one message to be sent while *Silence* means no message will be sent. The cost of sending the set of messages $\{m\}$ is assumed to be:

$$C(\{m\}) = \begin{cases} 0 & \text{if } \{m\} = \varphi \\ k_0 + k_1 \times \text{len}(\{m\}) & \text{otherwise} \end{cases}$$

where $\text{len}(\{m\})$ is the length of $\{m\}$, and k_0 and k_1 are coefficients.

6.6. Value Function

We measure the value gained by having *I* by two factors: *correctness* of the information to the need and *timeliness* of the fulfillment of the need. The rationale for considering these two factors is that there are different things that can affect the value of a given policy. In terms of timeliness, there may be a value associated with how

quickly a need can be satisfied, i.e., the sooner, the better. In terms of correctness, there may be a value associated with using the most recent value for *I*. For example, if the information *I* used to satisfy a need at time t_n was produced at an earlier time, t_0, it is possible that there is a more recent value for *I* produced at some time after t_0. However, since the information produced at time t_0 is immediately available, using this information may still have some values, although less than the use of more recently produced information. The highest value should accrue from using the newest information.

We measure the value gained by having *I* by two factors: *currency* of the information to the need and *timeliness* of the fulfillment of the need. The rationale for considering these two factors is that there are different things that can affect the value of a given policy. In terms of timeliness, there may be a value associated with how quickly a need can be satisfied, i.e., the sooner, the better. In terms of currency, there may be a value associated with using values other than the most recent for *I*. For example, if the information *I* used to satisfy a need at time t_n was produced at an earlier time, t_0, it is possible that there is a more recent value for *I* produced at some time after t_0. However, since the information produced at time t_0 is immediately available, using this information may still have some values, although less than that of the use of more recently produced information.

A common payoff function for the case like our problem which has multiple factors is to make a linear combination of these factors [29]. Based on this, we define the value function in term of the probability of using the most recent information vs.

the payoff of using the old information (value for *I* may have changed since last update).

$$V(S_t, \delta, t_n, t_p)$$

$$=T_s(t_n, t_p)\times P(S_t, \delta, t_n, t_p) + T_f(t_n, t_p)\times(1-P(S_t, \delta, t_n, t_p)),$$

where T_s denotes the reward, in terms of timeliness, of successfully using the most recent information, T_f denotes the reward, of using other than the most recent information, and P denotes the probability of using the most recent information.

We will define these three functions in a general way that encompasses a number of possible situations. T_s will be chosen to have a maximum value when information is immediately available and degrades as stale information is used. T_f will be chosen to reflect the value of using old information. P will be chosen to have a maximum value of 1 if information being used has not changed since the most recent production and have a minimum value of 0 if the information has changed.

6.6.1. Timeliness Function

In this section, we define T_s and T_f. First, the timeliness T_s of satisfying the need requires considerations of several cases:

1. The needer uses a value it already has. This means that $t_p<t_n$, and the value should be maximum because the need is immediately satisfied.
2. The needer asks a provider for a value and the provider returns a value it has previously obtained. This means that we can again consider $t_p<t_n$, the need is immediately satisfied, and again the value should be maximum.

3. The needer waits for a provider to proactively tell the information. In this case $t_p > t_n$ and the need is not immediately satisfied.
4. The needer asks a provider for the information and the provider waits until its next production of information to return a value. Again, in this case $t_p > t_n$, the need is not satisfied immediately, and one can expect a lower timeliness value.

The considerations can all be taken into account if the timeliness component of the value function is maximum for $t_p \leq t_n$ and decreases as t_p becomes greater than t_n. This, in turn, can be handled by first defining a time difference function d:

$$d(t_p, t_n) = \max(0, t_p - t_n).$$

We then define a non-increasing function f_s:

$$f_s(x) \text{ s.t. } 0 < f_s(y) \leq f_s(x) \text{ if } y \geq x.$$

f_s may take various forms. For example, it might decrease exponentially, or it might be constant for a length of time and zero thereafter, indicating that the information must be consumed in a finite length of time or it is useless. We leave f_s unspecified for the high level development. In general, f_s is required to have these properties:

1. For cases 1& 2, f_s is a max, (but the information may be stale, which is captured in the currency).
2. For cases 3 & 4, the most recent value is used (so currency is a max), but there is degradation due to waiting.

Later, we determine a specific one in our experiments (see Section 7.2.3.3). Finally, we use f_s to represent T_s:

$$T_s(t_n, t_p) = f_s(d(t_p, t_n)).$$

Next we consider T_f, the reward of timeliness under the condition of using old information. In some circumstances, old information still has value. For example, if an item (say an enemy troop location) has not been processed, it may still contain valuable information (the enemy troop is not far away from the previously reported location), although this case might reduce to a more simplistic decision algorithm (always send *I* to a needer). Thus, at the highest level, we represent T_f by a function $T_f = f_f()$ that expresses the pertinent factors. There are many forms that f_f could take for different types of *I*; this provides flexibility of defining T_f based on various focuses of different domains. For example, T_f can be a time discount function similar to T_s, if there is a value to use old information but the value decreases with the age of the information; or T_f can be a constant, implying that there is a fixed reward for using old information; or T_f can be zero if old information is completely useless for agents to make their further decision.

6.6.2. Currency Function

The general idea we will use for developing a model of currency is that value for *I* at time t_p should not change between t_p and the time it is used to satisfy the need at t_n. Let t_u be the time at which *I* is used by the needer for the need at time t_n. It will be useful to note that $t_u = \max(t_p, t_n)$, because if the value for *I* was produced before the

need arises, the value will not be used until the need arises, and if the value for *I* was produced after the need arises, the value will not be used until it is produced.

We consider the probability that the value does not change in time interval (t_p, t_u] and use that as the basis for measuring currency. There are still difficulties, as we may not know t_n and t_p and consequently t_u may be unknown too. However, from the EDF process, we do have probability mass functions on the times between successive occurrences of a need for *I* or production of new values for *I*. Using these distributions and the event time t as a reference point, we determine for each situation and policy estimates of the need and production times, and define a function P as:

$$P(S_t, \delta, t_n, t_p) = Pr(\neg\exists\, \tau \in Int(t_p, t_u] \ni I_P(\tau) \mid S_t \wedge \delta),$$

where $Int(t_p, t_u]$ denotes the interval between the two time points, noting that the time order is unspecified; $I_P(\tau)$ denotes the production of a value for *I* at time τ. P is the conditional probability that no other value is produced during $Int(t_p, t_u]$, conditional on a policy δ which is chosen in situation S_t at time t. For simplicity, we abbreviate this conditional event as NOPRODUCE(t_p, t_u), so P is abbreviated as:

$$Pr(NOPRODUCE(t_p, t_u)).$$

6.7. Calculating Probability of Currency

The currency function needs extensive probability calculations which include estimating the production time t_p and the need time t_n. For different situations, agents' knowledge of the various points in time t_n and t_p differ. In particular, in some situations a time is known exactly (e.g., the needer knows the time at which it needed a value) and in other situations, an agent can only estimate one or both of the times based upon

the known information. The knowledge of these time points is used in estimating the currency for each policy on each situation. We analyze them one by one below.

6.7.1. Situation PA: Provider Produces a Value for *I*

Recall that at the beginning of this section, we introduced a set of time points for each situation of decision-making. It is necessary to revisit them here. Fig. 6.5 redraws the time points and their relations for situation PA.

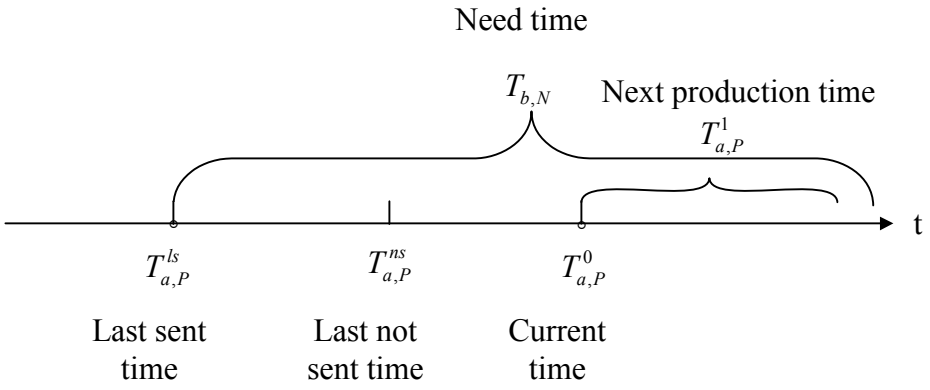


Fig. 6.5. Time Points for Situation PA.

In this situation, the provider just produced a value for *I* at time t=$T^0_{a,P}$. We denote a need for *I* at time t_n as $I_N(t_n)$ and the production of a value for *I* at time t_p as $I_P(t_p)$. The provider is making decisions on whether or not to provide $I_P(T^0_{a,P})$ to the needer. In order to identify t_n, the time of the need in question, we observe first that $t_n > T^{ls}_{a,P}$. This is because if there were a need before $T^{ls}_{a,P}$, the needer would either use a value it already had, or *Wait* or *ActiveAsk*, in which case $I_P(T^{ls}_{a,P})$ could be either a

ProactiveTell or a response to the *ActiveAsk*. By any choice, the most recent need before $T^{ls}_{a,P}$ would be satisfied and hence $t_n > T^{ls}_{a,P}$. Multiple needs may have arisen since $T^{ls}_{a,P}$, the first of which might use the information supplied at $T^{ls}_{a,P}$ through use of the *Silence* policy. Since we assume that the needer does not have a new need arise until after it has completed servicing a previous need, and, since we assume that there is only one provider, the frequency of new needs cannot exceed the frequency at which the provider provides values for *I*, though delay in satisfying a need could become large. Hence, we assume that in evaluating currency for this case, we need only consider the first need that arises after $T^{ls}_{a,P}$, we denote this time $T_{b,N}$ and take $t_n = T_{b,N}$. $T_{b,N}$ may be in the future and the time may be unknown to the provider, and we will have to consider the different policies that the needer might have made at time $T_{b,N}$.

As we shall see below, identifying t_p is not quite straightforward. In particular, it is not always appropriate to take $t_p = T^0_{a,P}$. We must consider two cases: 1) the provider uses policy *ProactiveTell* and 2) the provider uses policy *Silence*.

6.7.1.1. PA – ProactiveTell Is Used

In this case, $I_P(T^0_{a,P})$ will be provided for the need at $T_{b,N}$, hence $t_p = T^0_{a,P}$. The probability P is specified to:

$$\Pr(\text{NOPRODUCE}(T^0_{a,P}, t_u)),$$

where t_u is unknown because $I_P(T^0_{a,P})$ may or may not be used by the needer and depends on $T_{b,N}$ which is unknown. We then divide the problem into two sub-cases and t_u will be identify in case by case analysis on late this section,

Sub-case1: there is a need at $T^0_{a,P}$, i.e. $T_{b,N} \le T^0_{a,P}$;

Sub-case2: there is not a need at $T^0_{a,P}$, i.e. $T^0_{a,P} < T_{b,N}$.

Based on the law of total probability [149], P can be evaluated as:

$$\Pr(\text{NOPRODUCE}(T^0_{a,P}, t_u) \mid T_{b,N} \le T^0_{a,P}) \times \Pr(T_{b,N} \le T^0_{a,P}) +$$

$$\Pr(\text{NOPRODUCE}(T^0_{a,P}, t_u) \mid T^0_{a,P} < T_{b,N}) \times \Pr(T^0_{a,P} < T_{b,N}).$$

This shows that we need the probability that $T_{b,N}$ occurs in each of the relevant intervals, and given that it does, we need to examine function P through each of the possible decisions the needer might make. Some of the needer's decision, such as *ActiveAsk*, depends on the provider's responding decisions, such as *Reply* or *WaitUntilNext*. For this case, we also need to evaluate the provider's possible responding decisions. Thus the provider must estimate which policy the needer will choose from *Silence*, *ActiveAsk* and *Wait* and what the provider will response if *ActiveAsk* has been chosen by the needer.

$\Pr(T_{b,N} \le T^0_{a,P})$ and $\Pr(T^0_{a,P} < T_{b,N})$ can be calculated because we known $T^0_{a,P}$ and distributions of the unknown time point, thus $T_{b,N}$. Appendix A provides calculations to these two probabilities.

What left in P is Pr(NOPRODUCE($T^0_{a,P}$, t_u)) conditional on two sub-cases $T_{b,N} \leq T^0_{a,P}$ and $T^0_{a,P} < T_{b,N}$.

6.7.1.1.1. Sub-case1 $T_{b,N} \leq T^0_{a,P}$

In this case, the provider must consider each of the decisions the needer could make at $T_{b,N}$, to estimate if the needer will use $I_P(T^0_{a,P})$. If the needer obtains a response either by an *ActiveAsk* or a *Wait*, the needer will immediately use $I_P(T^0_{a,P})$ and hence $t_u = T^0_{a,P}$. Given $t_u = T^0_{a,P} = t_p$, obviously there no other value has been produced between t_u and t_p. Therefore the probability of NOPRODUCE equals 1 if the needer did not decide to keep *Silence* at $T_{b,N}$. However, if the needer kept *Silence*, it will use the last value for *I* sent at $T^{ls}_{a,P}$ by the provider. Then $t_u = T^{ls}_{a,P}$ and hence there is another production between the time interval $[t_u, t_p)$. Thus, the needed probability reduces to

$$\Pr(\text{NOPRODUCE}(T^0_{a,P}, T^0_{a,P}) \mid T_{b,N} \leq T^0_{a,P})$$

$$= 1 - \Pr(\text{needer decides to keep } \textit{Silence} \text{ at } T_{b,N}).$$

Next we calculate Pr(needer decides to keep *Silence* at $T_{b,N}$). First the provider needs to estimate unknown $T_{b,N}$. Though $T^{ls}_{a,P} < T_{b,N}$ (see Section 6.7.1), the base for estimating $T_{b,N}$ varies and the provider may or may not know the base. For example, if the needer chose *Silence* for the most recent need before $T^{ls}_{a,P}$, the base for estimating $T_{b,N}$ was the time at which the most recent need before $T^{ls}_{a,P}$ raised. Denote this time

$T_{b,N}^{-1}$. In this case, the provider won't know $T_{b,N}^{-1}$. While in the case that the needer *ActiveAsk* or *Wait* at the most recent need before $T_{a,P}^{ls}$, this need can be satisfied by $I_P(T_{a,P}^{ls})$, so in this case the base is $T_{a,P}^{ls}$ and the provider does know $T_{a,P}^{ls}$. It can be seen that the unknown $T_{b,N}$ increases the uncertainty on predicting the needer's decision makings.

To seek a reasonable and computationally feasible solution, we take advantage of the average lengths of time between information productions or needs which can be obtained from EDF process. We use the average length of information production or need as approximations in the future.

Let τ_n be the average length of time between needs of *I* by the needer. It is easily to get that $T_{b,N}^{-1} < T_{a,P}^{ls} < T_{b,N}$. One could use the expected value of where $T_{a,P}^{ls}$ would lie in the interval $(T_{b,N}^{-1}, T_{b,N})$, which under reasonable assumptions would be half way in between them. Then, we use $T_{a,P}^{ls} + \tau_n/2$ as an estimate for $T_{b,N}$. In the current sub-case, though, we are considering $T_{b,N} \leq T_{a,P}^{0}$, and there is no guarantee that $T_{a,P}^{ls} + \tau_n/2 \leq T_{a,P}^{0}$. Thus, we will use

$$T_{b,N} = \min(T_{a,P}^{0}, T_{a,P}^{ls} + \tau_n/2).$$

To estimate the needer's decision, the provider, using its own knowledge, will go through the needer's decision process and choose a policy which has maximum utility. This evaluation can be calculated as follows:

$$\Pr(\text{needer decides to keep } \textit{Silence} \text{ at } T_{b,N})$$

$$=\Pr(U(e, NA, \textit{Silence}, T_{b,N}, T^{ls}_{a,P}, \{m\}) > \text{Max}(U(e, NA, \textit{ActiveAsk}, T_{b,N}, T^{0}_{a,P}, \{m\}),$$

$$U(e, NA, \textit{Wait}, T_{b,N}, T^{0}_{a,P}, \{m\}))).$$

The solution to this will involve evaluating the utility function under each of the possible policies the needer might make at $T_{b,N}$, the time the need arises. The evaluation of all parts of the utility function except currency is straightforward. In the case of currency, the needer must associate time t_n and t_p with the time $T_{b,N}$. Fortunately, this is rather straightforward, given the conditions already known. The evaluation of currency for the needer under the condition that a need has arisen is given in Section 6.7.3. Since, as shown in Section 6.7.3, with substitutions for t_p and t_n, the expression for the utility function is deterministic, the utility can be determined for each possible policy. Having these utilities, the needer's decision is deterministic, thus it will use the policy which has the max utility. Therefore the provider's estimate of probabilities of the needer's choices is computable. Moreover, since the needer will only choose one policy at one decision point, so these probabilities equal either 1 or 0. For example, if the utility of *Silence* is the highest, then Pr(needer decides to keep *Silence* at $T_{b,N}$) = 1; otherwise it equals 0.

6.7.1.1.2. Sub-case 2 $T^{0}_{a,P} < T_{b,N}$

In this case, the value provided is not immediately used. Occurrence of the use of $I_P(T^{0}_{a,P})$ depends upon the decision the needer will make at $T_{b,N}$ and the provider's

responding decision at $T_{b,N}$. Since combinations of cases must be considered, we consider distinct four events:

$$\sum_{n=1}^{4} \Pr(\text{NOPRODUCE}(T_{a,P}^{0}, t_u) \mid T_{a,P}^{0} < T_{b,N} \wedge E_n) \times \Pr(E_n),$$

where E_n, n=1,..,4, denote the following events:

E_1: needer decides to *Wait* at $T_{b,N}$;

E_2: needer decides to keep *Silence* at $T_{b,N}$;

E_3: needer decides to *ActiveAsk* at $T_{b,N}$ $\wedge$ provider decides to *Reply* at $T_{b,N}$;

E_4: needer decides to *ActiveAsk* at $T_{b,N}$ $\wedge$ provider decides to *WaitUntilNext* at $T_{b,N}$.

We first consider $\Pr(\text{NOPRODUCE}(T_{a,P}^{0}, t_u) \mid T_{a,P}^{0} < T_{b,N} \wedge E_n)$. Since these probabilities turn out to be zero for some cases, $\Pr(E_n)$ does not need to be calculated for these cases.

6.1.1.1.2.1. Calculating $\Pr(\text{NOPRODUCE}(T_{a,P}^{0}, t_u) \mid T_{a,P}^{0} < T_{b,N} \wedge E_n)$

- *E_1:needer decides to Wait at $T_{b,N}$*

  In this case, the needer will wait until the next time the provider sends it a value for *I*. Since $T_{a,P}^{0} < T_{b,N}$, the needer certainly will not use $I_P(T_{a,P}^{0})$ and then $I_P(T_{a,P}^{0})$ is not relevant. In other words, the needer will use

another production to fulfill the need raised at $T_{b,N}$. Therefore the probability that no other value will be produced between t_p and t_u is zero.

$$\Pr(\text{NOPRODUCE}(T^0_{a,P}, t_u) \mid T^0_{a,P} < T_{b,N} \wedge E_1) = 0.$$

- *E_2: needer decides to keep Silence at $T_{b,N}$*

  In this case, the needer will use the most recent value it has for *I*. So it will use $I_P(T^0_{a,P})$ when $T^0_{a,P} < T_{b,N} < T^1_{a,P}$:

$$\Pr(\text{NOPRODUCE}(T^0_{a,P}, t_u) \mid T^0_{a,P} < T_{b,N} \wedge E_2)$$

$$= \Pr(T^0_{a,P} < T_{b,N} < T^1_{a,P}).$$

  $\Pr(T^0_{a,P} < T_{b,N} < T^1_{a,P})$ is calculated in Appendix A.

- *E_3: needer decides to ActiveAsk at $T_{b,N}$ $\wedge$ provider decides to Reply at $T_{b,N}$;*

  In this case, the needer will use the most recent value for *I* produced before $T_{b,N}$, This case is similar to the last case E_2, i.e. the needer will use $I_P(T^0_{a,P})$ when $T^0_{a,P} < T_{b,N} < T^1_{a,P}$. Therefore we have:

$$\Pr(\text{NOPRODUCE}(T^0_{a,P}, t_u) \mid T^0_{a,P} < T_{b,N} \wedge E_3)$$

$$= \Pr(T^0_{a,P} < T_{b,N} < T^1_{a,P}).$$

  Again $\Pr(T^0_{a,P} < T_{b,N} < T^1_{a,P})$ is calculated in Appendix A.

- *E_4: needer decides to ActiveAsk at $T_{b,N}$ $\wedge$ provider decides to WaitUntilNext at $T_{b,N}$.*

  Since $T^0_{a,P} < T_{b,N}$, the provider will reply the value for *I* at or after $T^1_{a,P}$ and the needer will use this value. Then $I_P(T^0_{a,P})$ will not be used and hence it is not relevant.

  $$\Pr(\text{NOPRODUCE}(T^0_{a,P}, t_u) \mid T^0_{a,P} < T_{b,N} \wedge E_4) = 0.$$

6.1.1.1.2.2. Calculating Pr(En)

From the above analysis, only two probabilities $\Pr(E_2)$ and $\Pr(E_3)$ must be determined because the factors on the other event probabilities are zero.

- *Pr(E_2): Pr(needer decides to keep Silence at $T_{b,N}$)*

  Pr(needer decides to keep *Silence* at $T_{b,N}$)

  =Pr(U(e, NA, *Silence*, $T_{b,N}$, t_p, {m}) > Max(U(e, NA, *ActiveAsk*, $T_{b,N}$, t_p, {m}), U(e, NA, *Wait*, $T_{b,N}$, t_p, {m}))).

  where t_p will be replaced, for a given policy, by the value that policy calls for.

  The time t_p is important in predicting the needer's decisions at $T_{b,N}$. To estimate t_p for each policy, the provider must predict its own decisions in the future. These decisions, in turn, closely depend on $T_{b,N}$. For example, in the case of a needer policy of δ = *Silence*, the needer will use the most

recent value for *I* it has. This value is the last value sent by the provider before $T_{b,N}$. So the provider needs to predict the number of production times between $T^0_{a,P}$ and $T_{b,N}$. The last sent value could be produced at any of these time points, so the provider also must estimate decisions it will make, *ProactiveTell* or *Silence*, on every production time. Since $T_{b,N}$ is unknown, the length of time between $T^0_{a,P}$ and $T_{b,N}$ is undetermined and hence the number of production time in between is undetermined. It can be see that the unknown $T_{b,N}$ and production time increases the uncertainty on predicting future decision makings. As before, we could take advantage of the average lengths of time between information productions or needs which can be obtained from EDF process.

In the following, we estimate $T_{b,N}$ and identify t_p for each three possible policies. Once $T_{b,N}$ and t_p are fixed, utility for every policy can be computed and then the needer's decision can be estimated.

We still use $T^{ls}_{a,P} + \tau_n/2$ as an estimate for $T_{b,N}$. In the current sub-case, though, we are considering $T_{b,N} > T^0_{a,P}$, and there is no guarantee that $T^{ls}_{a,P} + \tau_n/2 > T^0_{a,P}$. Thus, we will use

$$T_{b,N} = \max(T^0_{a,P}+1,\ T^{ls}_{a,P} + \tau_n/2).$$

Let τ_p be the average length of time for producing a new value for *I*. We use the following estimation for a future production time $T^i_{a,P}$:

$$T^i_{a,P} = T^0_{a,P} + \mathrm{i} \times \tau_p,\ \mathrm{i} \geq 1,$$

Having these estimations, we can specify i and then fix t_p for each policy. We consider three needer policies separately.

Needer δ = Silence

In this case, t_p is the time of the last value for sent *I* before $T_{b,N}$. Since $T^1_{a,P} \leq T_{b,N}$, the provider should not delivery a value for *I* produced before the most recent production time before $T_{b,N}$, because otherwise this provided value is out-of-date for the need raises at $T_{b,N}$. On this basis, we assume t_p equals the most recent production time before $T_{b,N}$. We define a function Z:

$$\mathrm{Z} = \left\lfloor \frac{T_{b,N} - T^0_{a,P}}{\tau_p} \right\rfloor.$$

Z returns 0 or a positive integer, meaning the number of productions during $T_{b,N}$ and $T^1_{a,P}$. Let:

$$t_p = T^0_{a,P} + \mathrm{Z} \times \tau_p. \qquad (6\text{-}1)$$

Needer δ = ActiveAsk

In this case, t_p depends upon the provider's decision at $T_{b,N}$. So the provider needs to predict its respondent decisions, *Reply* or *WaitUntilNext*, to the *ActiveAsk* received at $T_{b,N}$.

Pr(the provider decides to *Reply* at $T_{b,N}$)

=Pr(U(e, PB, *Reply*, $T_{b,N}$, t_p, {m}) > U(e, PB, *WaitUntilNext*, $T_{b,N}$, t_p, {m}))

where t_p of *Reply* is the production time just before $T_{b,N}$. It is exactly Eq. (6-1), i.e. $t_p = T^0_{a,P} + Z \times \tau_p$, and t_p of *WaitUntilNext* is the production timejust after $T_{b,N}$:

$$t_p = T^0_{a,P} + (Z+1) \times \tau_p \qquad (6\text{-}2)$$

Having these estimations, Pr(the provider decides to *Reply* at $T_{b,N}$) can be calculated by computing the utility for each of the possible policies. Therefore the utility can be determined for each of the possible policies. Again since the provider will make one responding decision at one per request, so Pr(the provider decides to *Reply* at $T_{b,N}$) equals either 1 or 0.

After estimating the provider's responding decisions, t_p can be fixed. Thus if the provider *Reply*, t_p is calculated by Eq. (6-1); if the provider *WaitUntilNext*, t pos calculated by Eq. (6-2).

Needer δ = *Wait*

In this case, t_p is the next *ProactiveTell* time after $T_{b,N}$. Since there is a need, the provider should not let the needer wait long, because otherwise the value for *I* provided is not timely to the need which has already risen. Therefore we assume the next *ProactiveTell* time is the next production time after $T_{b,N}$. Thus, we use Eq. (6-2) as estimate for t_p. Once t_p is estimated, the provider is able to evaluate the utility function under each of the possible policies the needer might make at $T_{b,N}$.

From above analysis, one can see that, to calculate the probability of the needer's single decision, the provider needs to compute utilities for all possible needer's policies and provider's responding policies. Then in the future if the probability for one needer's policy can be computed, then the probability for other needer's policies and the provider's responding policies are also computable.

- *Pr(E_3): Pr(needer decides to ActiveAsk at $T_{b,N}$ $\wedge$ provider decides to Reply at $T_{b,N}$)*

  Pr(needer decides to *ActiveAsk* at $T_{b,N}$ $\wedge$ provider decides to *Reply* at $T_{b,N}$)

  =Pr(needer decides to *ActiveAsk* at $T_{b,N}$)

  ×Pr(provider decides to *Reply* at $T_{b,N}$ | needer decides to *ActiveAsk* at $T_{b,N}$)

In this equation, Pr(needer decides to *ActiveAsk* at $T_{b,N}$) is calculable. This is because in the last case E_2 which computes Pr(needer decides to keep *Silence* at $T_{b,N}$), we calculate the utility for each of the needer's possible policy: *ActiveAsk*, *Silence* and *Wait*. Based on this, Pr(needer decides to *ActiveAsk* at $T_{b,N}$) equals 1 if *ActiveAsk* has the maximum utility, or 0 otherwise. The probability Pr(provider decides to *Reply* at $T_{b,N}$ | needer decides to *ActiveAsk* at $T_{b,N}$) is calculated in E_2 the case of *Needer* δ=*ActiveAsk*. So this probability is computable and equals either 1 or 0.

6.7.1.2. PA – Silence Is Used

Above we presented the calculation of currency for policy *ProactiveTell* in situation PA, which is when a provider produces a value for *I*. This section we consider the other provider's policy *Silence* in situation PA.

In this case, the value for *I* at $T^0_{a,P}$ will not be provided proactively. t_p thus depends upon the needer's decision at t_n and, if needed, the provider's responding decision at t_n. P equals:

$$\Pr(\text{NOPRODUCE}(t_p, t_u)),$$

where t_p and t_u will be identified in two sub-cases:

Sub-case1: there is a need at $T^0_{a,P}$, i.e. $T_{b,N} \leq T^0_{a,P}$;

Sub-case2: there is not a need at $T^0_{a,P}$, i.e. $T^0_{a,P} < T_{b,N}$.

P can be evaluated as:

$$\Pr(\text{NOPRODUCE}(t_p, t_u) \mid T_{b,N} \le T^0_{a,P}) \times \Pr(T_{b,N} \le T^0_{a,P})$$

$$+\Pr(\text{NOPRODUCE}(t_p, t_u) \mid T^0_{a,P} < T_{b,N}) \times \Pr(T^0_{a,P} < T_{b,N}).$$

In this form, $\Pr(T_{b,N} \le T^0_{a,P})$ and $\Pr(T^1_{a,P} < T_{b,N})$ have been done in Appendix A. So we consider $\Pr(\text{NOPRODUCE}(t_p, t_u)$ for the two sub-cases $\Pr(T_{b,N} \le T^0_{a,P})$ and $\Pr(T^0_{a,P} < T_{b,N})$.

6.7.1.2.1. Sub-case1 $T_{b,N} \le T^0_{a,P}$

This case means that the needer must not decide to *ActiveAsk* at $T_{b,N}$, because otherwise the provider would be obligated to provide the value at $T^0_{a,P}$ and could not choose *Silence*. Thus, the needer either decided to *Wait*, or keep *Silence*. If it decided to *Wait*, t_p depends upon the decision the provider will make at the production time after $T^0_{a,P}$. For example, if the provider decide to *ProactiveTell* at $T^1_{a,P}$, then $t_p = T^1_{a,P}$, otherwise if the provider decide to keep *Silence* then t_p again depends upon the provider's decision at $T^2_{a,P}$ and so on. However, whatever t_p would be, since the needer is waiting, it definitely will use the value produced at t_p, i.e. $t_u = t_p$. Obviously there is no other value produced between t_u and t_p, so $\Pr(\text{NOPRODUCE}(t_p, t_u))=1$.

However, if the needer decided to keep *Silence*, it will use the value for *I* produced at $T^{ls}_{a,P}$, then $t_p = T^{ls}_{a,P}$ and $t_u = T_{b,N}$. $\Pr(\text{NOPRODUCE}(T^{ls}_{a,P}, T_{b,N}))$ is equal to the probability that there is no values for *I* produced between $T^{ls}_{a,P}$ and $T_{b,N}$ that the provider did not send to the needer. We consider two cases.

If the provider knows that there were no values produced between the last one it sent to the needer and the current time, then

$$\Pr(\text{NOPRODUCE}(T^{ls}_{a,P},\ T_{b,N}) \mid T_{b,N} \le T^{0}_{a,P}) = 1.$$

Otherwise, in order for there to have been at least one value for *I* produced and not sent out between $T^{ls}_{a,P}$ and $T^{0}_{a,P}$, $T_{b,N}$ must be less than $T^{ns}_{a,P}$ in the interval $(T^{ls}_{a,P}, T^{ns}_{a,P}]$. The provider can then compute:

$$\Pr(\text{NOPRODUCE}(T^{ls}_{a,P},\ T_{b,N}) \mid T_{b,N} \le T^{0}_{a,P})$$

$$= \Pr(T_{b,N} \in (T^{ls}_{a,P}, T^{ns}_{a,P}])$$

$$= \text{CDF}_{b,N}(T^{ns}_{a,P} - T^{ls}_{a,P} - 1).$$

where $\text{CDF}_{b,N}$ means the cumulative density function of agent *b*'s need time interval. Since it is the provider doing the evaluation, the provider will use its estimate of the CDF based upon the information it has received from the needer over the past history to calculate this probability.

6.7.1.2.2. Sub-case 2 Pr($T^{0}_{a,P} < T_{b,N}$)

In this case, t_p depends upon the decision the needer will make at $T_{b,N}$ and the provider's decision at $T_{b,N}$. We consider the same combinations of cases that have been defined in the case *ProactiveTell*.

$$\sum_{n=1}^{4} \Pr(\text{NOPRODUCE}(t_p, t_u) \mid T^{0}_{a,P} < T_{b,N} \wedge E_n) \times \Pr(E_n).$$

We first consider $\Pr(\text{NOPRODUCE}(t_p, t_u) \mid T^{0}_{a,P} < T_{b,N} \wedge E_n)$ and then identify which $\Pr(E_n)$ needs to be determined.

- *E_1:needer decides to Wait at $T_{b,N}$*

  In this case, t_p is the next *ProactiveTell* time after $T_{b,N}$, so $t_u = t_p$. There is no other production between t_p and t_u, and the probability equals 1:

  $$\Pr(\text{NOPRODUCE}(t_p, t_u) \mid T^0_{a,P} < T_{b,N} \wedge E_1) = 1.$$

- *E_2: needer decides to keep Silence at $T_{b,N}$*

  In this case, t_p is the production time of the most recent value for *I* the needer has. So $t_u = T_{b,N}$. To determine t_p, we can calculate Z, as in Eq. (6-1), to find the number of productions between $T^0_{a,P}$ and $T_{b,N}$. If Z = 0, meaning $T^0_{a,P} < T_{b,N} < T^1_{a,P}$, then $t_p = T^{ls}_{a,P}$. $\Pr(\text{NOPRODUCE}(T^{ls}_{a,P}, T_{b,N}) \mid T^0_{a,P} < T_{b,N} \wedge E_1) = 0$ because there is at least one production time, $T^0_{a,P}$, in between. If $Z \geq 1$, meaning $T^1_{a,P} \leq T_{b,N}$, then $\Pr(\text{NOPRODUCE}(t_p, T_{b,N}) \mid T^0_{a,P} < T_{b,N} \wedge E_1) = 1$, because t_p is the most recent production time before $T_{b,N}$.

- *E_3: needer decides to ActiveAsk at $T_{b,N}$ $\wedge$ provider decides to Reply at $T_{b,N}$*

  In this case, $t_u = T_{b,N}$ and t_p is the most recent production time before $T_{b,N}$. There is no other production between t_p and t_u:

  $$\Pr(\text{NOPRODUCE}(t_p, t_u) \mid T^0_{a,P} < T_{b,N} \wedge E_3) = 1.$$

- *E_4: needer decides to ActiveAsk at $T_{b,N}$ ∧ provider decides to WaitUntilNext at $T_{b,N}$.*

In this case, t_p is the next production time after $T_{b,N}$, and t_u=t_p.

There is no other production between t_p and t_u:

$$\Pr(\text{NOPRODUCE}(t_p, t_u) \mid T^0_{a,P} < T_{b,N} \wedge E_4) = 1.$$

From above analyses, either $\Pr(\text{NOPRODUCE}(t_p, t_u) \mid T^0_{a,P} < T_{b,N} \wedge E_n)=1$ for all E_n, or it equals 1 for all except for E_2, in which case $E_2 = 0$ (from the above argument); the choice depends on Z, which the provider can know. For the former case, $\sum_{n=1}^{4} \Pr(\text{NOPRODUCE}(t_p, t_u) \mid T^0_{a,P} < T_{b,N} \wedge E_n) \times \Pr(E_n)=1$. For the latter, we need to calculate $\Pr(E_2)$, Pr(needer decides to keep *Silence* at $T_{b,N}$). If $\Pr(E_2)=1$, meaning $\Pr(E_1)$, $\Pr(E_3)$ and $\Pr(E_4)$ are all equal 0, then

$$\sum_{n=1}^{4} \Pr(\text{NOPRODUCE}(t_p, t_u) \mid T^0_{a,P} < T_{b,N} \wedge E_n) \times \Pr(E_n)=0.$$

If $\Pr(E_2)=0$, meaning one of $\Pr(E_1)$, $\Pr(E_3)$ and $\Pr(E_4)$ must equals 1, then $\sum_{n=1}^{4} \Pr(\text{NOPRODUCE}(t_p, t_u) \mid T^0_{a,P} < T_{b,N} \wedge E_n) \times \Pr(E_n)=1$.

The calculation of $\Pr(E_2)$ is similar to what has been done in the previous case of *ProactiveTell*. $T_{b,N}$ is estimated to be $\max(T^0_{a,P}+1,\ T^{ls}_{a,P} + \tau_n/2))$. t_p is specified by considering each of the needer's possible decisions at $T_{b,N}$, *Silence*, *Wait* and *ActiveAsk*, and the provider's responding decisions, *Reply* and *WaitUnitilNext*, to *ActiveAsk*. There is only one difference between the previous case and the present one.

For the present case, since the provider won't provide $I_P(T^0_{a,P})$, so for the case of needer's δ= *Silence* policy, if $T^0_{a,P} < T_{b,N} < T^1_{a,P}$, $t_p = T^{ls}_{a,P}$ ($t_p = T^0_{a,P}$ for the previous case). After estimating t_p, the utility for each of the needer's possible policies is determined and then the needer's decision can be determined.

6.7.2. Situation PB: Provider Receives a Request about *I*

In this situation, the provider receives a request from the needer at time $T_{b,q}$. For clarity, Fig. 6.6 redraws time points for this situation below.

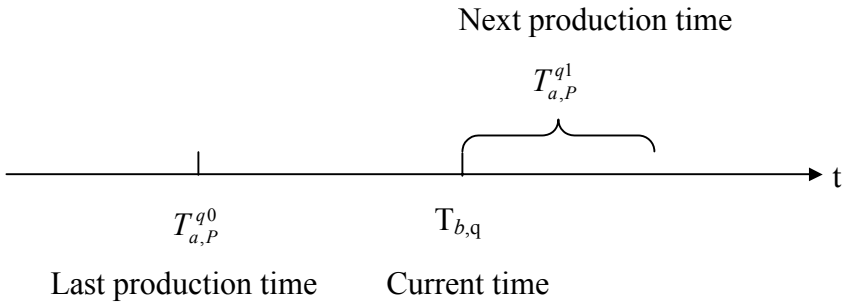


Fig. 6.6. Time Points for Situation PB.

There is a need at the current time $t = T_{b,q}$, then $t_n = T_{b,q}$. The provider will either *Reply* with the most recent value produced or *WaitUntilNext* production. Since there is a need, either value which is about to be provided will be used by the needer. P equals:

$$\Pr(\text{NOPRODUCE}(t_p, t_u)),$$

where t_p and t_u will be identified in the policy being chosen.

In the case of *Reply*, the most recent value for *I* produced at $T_{a,P}^{q0}$ will be provided for the need at $T_{b,q}$. So $t_p = T_{a,P}^{q0}$. Because there is a need, the needer definitely will use $I_P(T_{a,P}^{q0})$ when receives it, so $t_u = T_{b,q}$. By definition the most recent value means no other value has been produced since $T_{b,q}$, hence there was no other value for *I* produced between $T_{a,P}^{q0}$ and $T_{b,q}$. Therefore:

$$\Pr(\text{NOPRODUCE}(T_{a,P}^{q0}, T_{b,q})) = 1.$$

In the case of *WaitUntilNext*, next value for *I* which will be produced at $T_{a,P}^{q1}$ will be provided for the need at $T_{b,q}$. So $t_p = T_{a,P}^{q1}$ and $t_u = T_{a,P}^{q1}$. Therefore:

$$\Pr(\text{NOPRODUCE}(T_{a,P}^{q1}, T_{a,P}^{q1})) = 1.$$

Therefore currency function equals 1 for both policies, *Reply* and *WaitUntilNext*, in situation PB.

6.7.3. Situation NA: Needer Has a Request about *I* Arise

Fig. 6.7 shows time points for this situation.

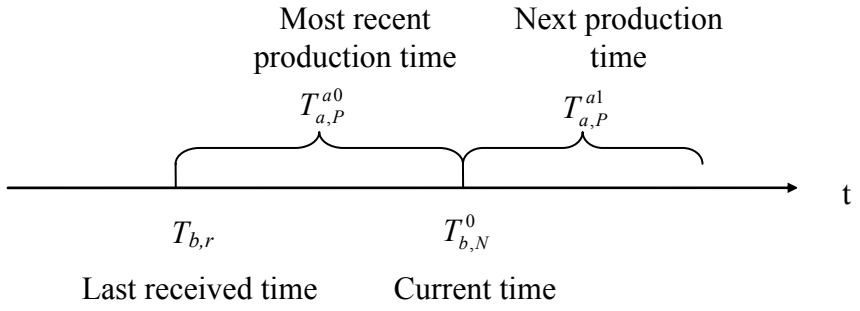


Fig. 6.7. Time Points for Situation NA.

This is a need for *I* at the current time t = $T^0_{b,N}$, so $t_n = T^0_{b,N}$. When the needer needs *I* at $T^0_{b,N}$, it has three policies to choose from: *Silence*, *ActiveAsk* and *Wait*. If the needer *ActiveAsk*s the provider or *Waits* for a *ProactiveTell* from the provider, it will always use the most recent value for *I*. However, if the needer keeps *Silence*, i.e., it uses the value last received at time $T_{b,r}$, it is possible that the value for *I* has been changed since then. We consider each of these sub-cases below.

P equals:

$$\Pr(\text{NOPRODUCE}(t_p, t_u)),$$

6.7.3.1. NA – ActiveAsk Is Used

If *ActiveAsk* is chosen, the value for *I* provided for the need at $T^0_{b,N}$ will be either $T^{a0}_{a,P}$ or $T^{a1}_{a,P}$, depending on the provider's response (*Reply* or *WaitUntilNext*). If the provider does *Reply*, it will send the most recent value it has produced, then $t_p = T^{a0}_{a,P}$. Since there is a need, the needer will immediately use $I_P(T^{a0}_{a,P})$ so $t_u = T^0_{b,N}$. There was no other value produced between $T^{a0}_{a,P}$ and $T^0_{b,N}$. On the other hand, if the provider *WaitUntilNext*s, $t_p = T^{a1}_{a,P}$ and $t_u = t_p = T^{a1}_{a,P}$. It is also the case that there has no value produced in the interim. Thus, in either case the probability for no other value produced between t_p and t_u is one:

$$\Pr(\text{NOPRODUCE}(t_p, t_u)) = 1.$$

6.7.3.2. NA – Silence Is Used

When *Silence* is used, the most recent value for *I* that the needer has will be used for the need at $T^0_{b,N}$, so $t_p = T_{b,r}$. This value will be used at $T^0_{b,N}$, then $t_u = T^0_{b,N}$. The probability that no other value was produced between $T^0_{b,N}$ and $T_{b,r}$ equals to the probability of $T_{b,r}=T^{a0}_{a,P}$, which means that $T_{b,r}$ is the time at which the most recent value was produced. One can then compute:

$$\Pr(\text{NOPRODUCE}(t_p, t_u))$$

$$=\Pr(T_{b,r}=T^{a0}_{a,P})$$

$$=1-\Pr(T_{b,r}<T^{a0}_{a,P})^{14}$$

$$=1-\Pr(T_{b,r}<T^{a0}_{a,P}\leq T^0_{b,N})$$

$$=1-\text{CDF}_{a,P}(T^0_{b,N}-T_{b,r}-1)$$

where $\text{CDF}_{a,P}$ means the cumulative density function of agent *a*'s production time interval. Since it is the needer doing the evaluation, the needer will use its estimate of the CDF based upon the information it has received from the provider over the history to calculate this probability.

6.7.3.3. NA – Wait Is Used

If *Wait* is adopted, t_p is some production time of the value for *I* at which the provider will *ProactiveTell* in the future. This value will not be used until being produced, thus $t_u=t_p$. Obviously, there is no value produced between t_p and t_u:

---

[14] $T_{b,r}$ cannot be greater than $T^{a0}_{a,P}$ since the latter is the most recent production time.

$$\Pr(\text{NOPRODUCE}(t_p, t_u)) = 1.$$

6.7.4. Situation NB: Needer Receives a Value for *I*

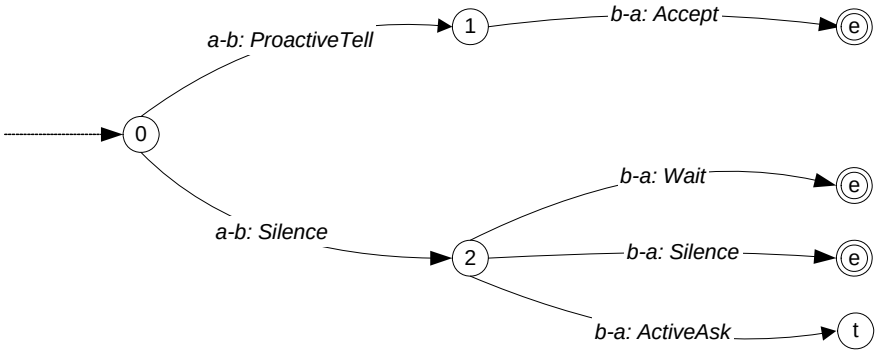
On this situation, the needer will *Accept* the value for *I* sent from the provider. The decision is deterministic, so we do not consider this case.

6.8. Decision-Making Processes

Figs. 6.8-6.11 show finite state diagrams representing the communication process of getting and telling (respectively) an information item. Each node represents a decision point. As one proceeds through the graph, the nodes represent alternating decisions by the needer and the provider. The nodes marked "e" are special in the sense that they represent the receipt of the information, or a timeout condition (explained below). The nodes marked "t" denoting a transfer from one node to another. For example, in situation PA of Fig. 6.8, agent *a* may receive an *ActiveAsk* from agent *b* when deciding to keep *Silence* to agent *b*. In such a case, the state will transfer to the start state of situation PB, the situation where *a* receives a request from agent *b*. In this case, agent *a* needs to update its data about agent *b*'s need time and decide if to *Reply* agent *b* right away or *WaitUntilNext* production. By either decision which *a* will make, agent *b* is able to receive an information item, so an "e" node is reached and the decision making ends.

Since some of the decisions that can be made involving waiting an arbitrary length of time for another agent to do something, the possibility of infinite waits arises. To circumvent this, we use a heuristically chosen loop breaking algorithm. If a needer does not get the information during a time cut-off, the needer adopts policy *ActiveAsk*.

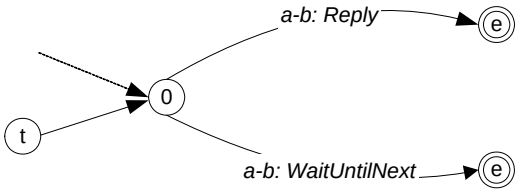
We choose the needer to do this, because it very likely that the needer has few chances to start communication if proactivity is fully explored. So letting the needer break time-out can increase chances of sending the needer's history data to the provider.



a: provider *b*: needer e: end t: transfer

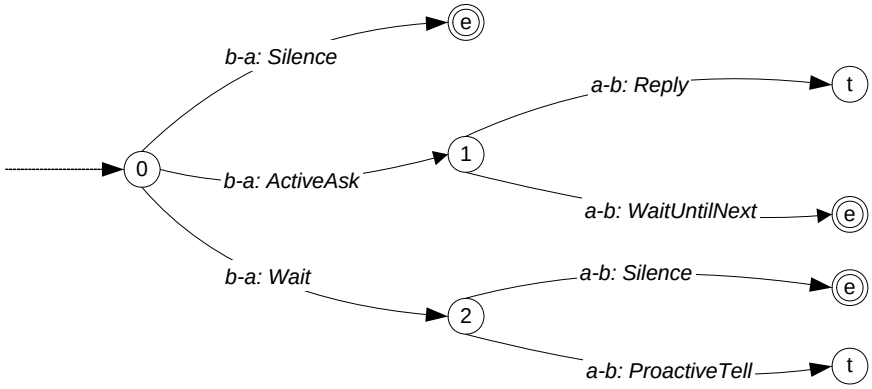
Situation PA: Provider produces a new piece of information

Fig. 6.8. Decision-Making Process of Provider in Situation PA.



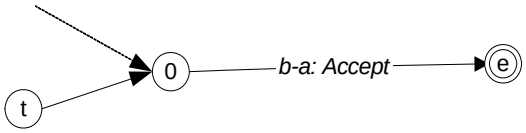
Situation PB: Provider receives a request for a piece of information

Fig. 6.9. Decision-Making Process of Provider in Situation PB.



Situation NA: Needer needs a piece of information

Fig. 6.10. Decision-Making Process of Needer in Situation NA.



Situation NB: Needer receives a piece of information

Fig. 6.11. Decision-Making Process of Needer in Situation NB.

6.9. Decision-Theoretic Proactive Communication

DTPC (Decision-Theoretic Proactive Communication) is the overall process for managing communication. It has three parts: an algorithm for selecting a policy, algorithms for providing information and algorithms for getting needed information.

Fig. 6.12 shows the algorithm for selecting a policy. The *identify* function identifies parameters (t_p and t_n) that will be used appropriately, based on the values listed in Table 6.2. The *evaluate* function calculates utility of each policy. For each counterpart agent Ag_i, a policy δ_i with the maximum utility is added to *policyList*. By comparing each δ_i in *policyList*, a final $\delta_i^{`}$ with maximum utility is selected.

```
/*self is an agent who makes the decision;
  counterparts is an agent set about whom the decision is made;
  I is information that communication conveyed.
*/
selectPolicy(self, counterparts, I){
  policyList = null;

  ∀ Agi ∈ counterparts
        ∀ policy δi
              identify(self, Agi, δi, I);
              U(δi)=evaluate(self, Agi, δi, I);
        select one δi with maximum U;
        add δi to policyList;

  select one δi` with maximum U from policyList;
  return δi`;
}
```

Fig. 6.12. A Policy Selection Algorithm.

Figs. 6.13 and 6.14 show algorithms for providing a piece of information to a needer, or getting a piece of information from a provider. Generally, agents select a policy that has maximum utility and act corresponding to that and their counterpart's response. T_o is a time cutoff [15] used by needer to guarantees that the system does not go into a waiting forever state.

Function *updateSelfData* updates the decision maker's (provider or needer) information production or need time intervals. *updateSelfData* is executed when the decision maker produces/needs an information item. Function *updateOtherData* updates the decision maker's knowledge about counterpart's production or need time intervals which are from historical data (if any) attached in each message sent to the counterpart. Updating this data can help agents make better estimation for distributions of information production or need.

[15] Of course, if desired, one could use a different cutoff for each situation.

```
/*Executed when provider is in situation PA at time t.
  Let pendWUNList be a list of needers whose requests will be replied with
  WaitUntilNext production.
*/
provideNeededInfo(provider, needers, I, t){//needers is a needer set
  updateSelfData(provider, I, t);  //update provider's production time

  if (pendWUNList != null)        //there is pending WaitUntilNext reply(s)
      reply I to A0;              //A0 is the first needer on pendWUNList;
      updateOtherData(A0, I, t);
      remove A0 from pendWUNList;
      exit;

  δi` = selectPolicy(provider, needers, I);
  switch(δi`)
      case ProactiveTell:
              ProactiveTell needersi;
              updateOtherData(needersi, I, t);
              break;
      case Silence:
              Silence;
              break;
}

/*Executed when provider is in situation PB at time t.
*/
receiveRequest(provider, needer, I, t){
                                        //needer is a single agent who needs I
  δi` = selectPolicy(provider, needer, I);
  switch(δi`)
      case Reply:
              Reply needer;
              updateOtherData(needer, I, t);
              break;
      case WaitUntilNext:
               add needer to pendWUNList;
               break;
}
```

Fig. 6.13. Algorithms about Providing Information.

```
/*Executed when needer is in situation NA at time t.*/
getNeededInfo(needer, providers, I, t){
  set time cutoff To;
  waitTime = 0;
  boolean obtained=FALSE, waiting=FALSE;
  updateSelfData(needer,  I, t);   //update self need time
  δi` = selectPolicy(needer, providers, I);
  switch(δi`)
        case Silence:
              Silence;          //use most recent value it has
              break;
        case ActiveAsk:
              ActiveAsk providersi;
              if providersi sends Reply
                    receiveInfo(providersi, I, t);//transfer to situation NB
              else              //provider chose WaitUntilNext
                    Wait;
                    waiting = TRUE;
              break;
        case Wait:
              Wait;
              waiting = TRUE;
              break;
  if (waiting)
        while ((!obtained)&&(waitTime<To))
              waitTime++;
              if providersi sends WaitUntilNext reply
                    receiveInfo(providersi, I, t+waitTime);
                    obtained=TRUE;
              if a provider p Proactivetells I
                    receiveInfo(p, I, t+waitTime);
                    obtained=TRUE;
        if (!obtained)
              randomly select a provider q;
              ActiveAsk q;
}
/*Executed when needer is in situation NB at time t.*/
receiveInfo(provider, I, t){
      updateOtherData(provider, I, t);
}
```

Fig. 6.14. Algorithms about Getting Needed Information.

6.10. Summary

In this section, we have presented a method for achieving proactive communication using decision theory for determining the communication policy to be used. We have identified each situation that might (or might not) involve the exchange of information; we have identified the policies that could be selected. We have then introduced the general form of a utility function that can be used for the decision theoretic selection of the best policy. In order to do this, it is necessary to estimate the value of the utility function, as some of the independent variables cannot be precisely known by the evaluating agent. In addition, the decision-making process is interdependent between provider and needer, so estimation about each other's decision is also necessary for evaluating these variables.

CHAPTER VII

AN APPLICATION DOMAIN DESIGN AND EVALUATIONS

This section describes the design for an application domain and the experiments used to evaluate the Proactive Communication approach.

An applicable domain should meet following criteria:

- Messages are allowed to be sent.
- Communciation is assumed to have cost and risk.
- The team has a common goal which can be accomplished by executing a set of team plans. There are information needs among the team. Agents need to know some information to execute these team plans.
- There is uncertainty during teamwork. The uncertainty may be caused by agents holding incomplete knowledge about the time of information production and need and about the world.
- The teamwork process is characterized by stochastic properties. For example, due to random moves of objects in the world, durations of plan executions are random variables.

We have extended the classic Wumpus World problem [102] into a multi-agent version and used this as the application domain. Two evaluations were performed: a test of the effectiveness of observability, as the part of the overall approach; and a test of the effectiveness of the overall approach, which includes the empirical distribution

function method for predicting information production and need and the decision theoretic method on deciding a communication policy.

7.1. Evaluation of Observability

7.1.1. Multi-Agent Wumpus World

For this evaluation, the world is 20 by 20 cells and has 20 wumpuses, 8 pits, and 20 piles of gold.

The team is composed of 1 carrier and 3 fighters and is allowed to operate a fixed number of 150 steps. The team goal is to kill wumpuses and get the gold without being killed. The carrier is capable of finding wumpuses and picking up gold. We assume that the carrier is strong enough to carry all of the gold it finds. The fighters are capable of shooting wumpuses.

Every agent can sense a stench (from adjacent wumpuses), a breeze (from adjacent pits), and glitter (from the same position) of gold. When a piece of gold is picked up, both the glitter and the gold disappear from its location. When a wumpus is killed, both the stench and the wumpus' body are removed from the world. The environment simulation maintains object properties and agents' actions.

The agents may also have observabilities, while their observing radii may be different. Each agent has an individual knowledge base (KB) to save the beliefs it generates after observing the world and actions of other agents. The observabilities are encoded as rules in agents' KB. The inference engine used is JARE [60].

The agents are randomly located in the world and know each other's starting location. In the absence of any target information (wumpus or gold), all agents reason

about the world to determine their priority of potential movements. Basically, they move to locations not previously visited (when possible, though they may revisit a location if there are no reachable unvisited safe locations). If they are aware of a target location requiring action on their part (shoot wumpus or pick up gold), they move toward the target. In all cases, they avoid unsafe locations.

If the fighter senses other wumpuses while it is on the way to kill the wumpus about which the carrier has told it, it will kill them first. This is because the fighter has a limited range of vision, so the wumpuses it senses must be close and can be killed quickly.

7.1.2. Problem Analysis

In ODBC, Proactive Communication has two primary protocols named *O-Tell* and *O-Ask*. These protocols are used by each agent to generate inter-agent communication when information exchange is desirable.

Decisions about whether to use *O-Tell* or *O-Ask* (see Section 4.5) when observing an information item depends on the relative frequency of information need vs. production. For any piece of information *I*, we define two functions, f_C and f_N [135]. $f_C(I)$ returns the frequency with which *I* changes. $f_N(I)$ returns the frequency with which I is used by agents. We classify information into two types – static[16] and dynamic. If $f_C(I) \leq f_N(I)$, *I* is considered static information; if $f_C(I) > f_N(I)$, *I* is considered

[16]Here, static information includes not only information that never changes, but also information infrequently changed but frequently needed.

dynamic information. For static information we use *O-Tell* by providers, and for dynamic information we use *O-Ask* by needers[17].

In order to understand the Proactive Communication problem in the Multi-Agent Wumpus World domain, we present the team's plans, which are based on the team plans in [137], which were developed for the original CAST[18]. Fig. 7.1 shows the major part of the team plan:

```
(plan killWumpus()
        (process
          (seq
                (agent-bind ?ca (constraint (play-role ?ca carrier)))
                (DO ?ca (findWumpus)) ; carrier is assigned
                (agent-bind ?fi (constraint ((play-role ?fi fighter)
                                (closest-to-wumpus ?fi ?wumpusId))))
                                ;fighter who is closest to wumpus is assigned
                (DO ?fi (startKill))
)))
```

Fig. 7.1. An Example of Plans of the Multi-Agent Wumpus World.

Each agent has a copy of the team plan and will evaluate the pre-cond during the plan execution. The evaluation is based on the agent's own beliefs about the environment. The team plan does not explicitly state the communication that is to take place. Rather, the agents are to infer the necessary communication from their beliefs of the plan and the environment.

[17]In the next section 7.2, we address some statistical methods to calculate frequencies and hence will be able to provide more comprehensive proactive communication protocols.
[18] In the next experiment, we develop a new team plan for current CAST.

As one can infer from these plans, the key problems are:

1) Which kind of information will be communicated?
2) Who will need or produce the information?
3) Which information will be *O-Telled* and which will be *O-Asked*?

The answer to the first problem is that the conjunct that is part of the precondition of a plan or an action will be communicated in the team at the time when the conjunct is evaluated. For this example, the information is "wumpus location." In this evaluation, we encoded a domain-dependent role constraint, *closest-to-wumpus*, for selecting the fighter closest to the wumpus found (see Fig. 7.1). The selected fighter will be assigned the *startKill* plan and will kill the wumpus after arriving at the wumpus' location. Therefore an unknown conjunct that is part of a constraint (e.g., "fighter location") is another piece of information which is to be exchanged. However this kind of domain constraint is too specific to be generalized. After developing Dynamic Information Prediction and Decision Theoretic Proactive Communication, we are able to remove it from the domain and let the carrier decide who can be committed on the fly. The capability will be presented in our next evaluation to the overall Proactive Communication approach (see Section 7.2).

To determine who needs and who produces a given item of information, agents analyze the preconditions and effects of plans and actions, and generate a list of needers and a list of providers for every piece of information. The needers are agents who might need to know the information (e.g. the fighters), and the providers are agents who might know the information (e.g. the carrier and other fighters).

As to the third problem, the "wumpus location" is static information and the "fighter location" is dynamic information. Since the static information won't often change, agents use *O-Tell* to impart the static information they just learned if they believe other agents will need it. For example, the carrier *O-Tell*s the fighters the wumpus' location. Agents use *O-Ask* to request dynamic information if they need it and believe other agents have it. For example, fighters *O-Ask* each other about their locations.

7.1.3. Results and Analysis

Our goal is to evaluate effectiveness of agents' observabilities. Therefore we used two teams, a team has observability and a team does not have observability.

We report three experiments. The first explores how observability reduces communication load and improves team performance in multi-agent teamwork. The second focuses on the relative contribution of each type of belief generated from observability to the successes of CAST-O as a whole. Finally, the third evaluates the impact of observability on changing communication load with increase of team size.

7.1.3.1. Overall Effectiveness of Observability

Two teams are defined in below. Except for the observability rules, the conditions of both teams were exactly the same.

- Team A: The carrier can observe objects within a radius of 5 grid cells, and each fighter can sense objects within a radius of 3 grid cells.

- Team B: None of the agents have any sensing capabilities beyond the basic capabilities described at the beginning of the section.

We use measures of performance which reflect the number of wumpuses killed, the amount of communication used and the gold picked up. In order to make comparisons easier, we have chosen to have decreasing values indicate improving performance, i.e., smaller numbers of communication messages are better. To maintain this uniformity with some parameters of interest, we use the quantity of wumpuses left alive rather than the number killed. The experiments were performed on 5 randomly generated worlds. The results are shown in Table 7.1.

Table 7.1 shows that, as expected, Team A killed more wumpuses and found more gold than Team B. From other experiments, we have learned that the further the agents can, the more wumpuses they kill. It is interesting that the absolute number of communications is higher for Team A with observabilities than that of Team B, i.e., 33.8 vs. 28.8 for *O-Tell* and 77.4 vs. 67.6 for *O-Ask*. The number of *O-Tell*s in Team A were greater because the carrier, which is responsible for finding wumpuses and *O-Tell*ing their locations to fighters, has further vision than that of the carrier in Team B. Hence the carrier in Team A can sense more wumpuses. This feature leads to more *O-Tell*s from the carrier to the fighters in Team A. The number of *O-Tell*s can be reduced by the carrier's beliefs about the fighters' observability, i.e., if the carrier believes the fighters can sense the wumpus' location, it will not *O-Tell* the fighters. However, since the fighters' detection range is smaller than that of the carrier, the reduction cannot offset the number of extra *O-Tell*s. The reason for the increased number of *O-Ask*s in

Team A is that the more wumpuses team members find, the more likely it becomes that they send messages among themselves to decide who is closest to a particular wumpus. Although the number of the messages could be reduced by factors such as allowing the fighter to sense other fighters' locations and to sense other fighters killing a wumpus, the increase cannot be totally offset because of the fighters' short vision. Hence, it makes more sense to compare the average number of messages per wumpus killed. In these terms, the performance of Team A is much better than that of Team B, 2.23 vs. 5.9 for *O-Tell* and 5.09 vs. 13.6 for *O-Ask*. Hence, our algorithms for managing the observability of agents have been effective.

Table 7.1. Team Performance and Communication Amounts in Sample Runs.

Team A	T1	T2	T3	T4	T5	T6
Run 1	4	8	82	32	5.12	2.00
Run 2	5	9	76	35	5.06	2.33
Run 3	6	6	72	38	5.14	2.71
Run 4	5	7	80	32	5.33	2.13
Run 5	4	6	77	32	4.81	2.00
Average	4.8	7.2	77.4	33.8	5.09	2.23
Team B	T1	T2	T3	T4	T5	T6
Run 1	14	14	72	30	12.00	5.00
Run 2	16	16	62	27	15.5	6.75
Run 3	16	14	62	27	15.5	6.75
Run 4	14	15	72	30	12.00	5.00
Run 5	15	14	70	30	14.00	6.00
Average	15	14.6	67.6	28.8	13.6	5.90

T1: number of wumpuses left alive
T2: amount of gold left unfound
T3: total number of *O-Ask*s used
T4: total number of *O-Tell*s used
T5: average number of *O-Ask*s per wumpus killed
T6: average number of *O-Tell*s per wumpus killed

From this experiment, we learned two things. First, by introducing observabilities to agents, the amount of communication is increased slightly, because observability is a major means for an individual agent to obtain information about the environment and team members; the more information obtained by the agent, the more messages it conveys to help others. Second, observability can greatly decrease the number of communications when normalized by some measure of team performance, which, in this example, is the average number of communications per wumpus killed, denoted by ACPWK.

7.1.3.2. Effectiveness of Different Perspectives of Observability

The second experiment tested the contribution of different categories of belief generated from observability to the successful reduction of the communication. These beliefs are as follows:

1) belief1: beliefs about an observed property.
2) belief2: beliefs about an observed action whose pre-cond contains the information worth exchanging.
3) belief3: beliefs about an observed action whose effect contains the information worth exchanging.
4) belief4: beliefs about another agent sensing a property[19].

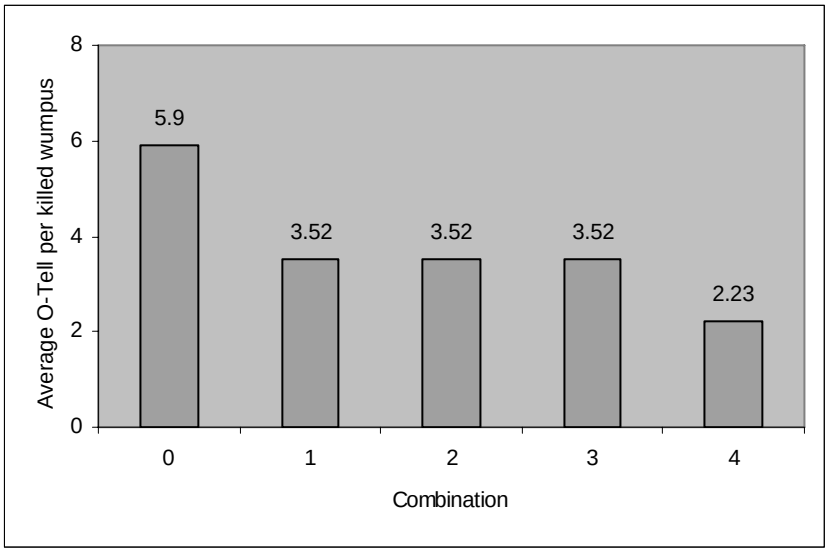
We test their contributions by combining them. We used Team A and Team B in this experiment and kept all conditions the same as those of the first experiment. We

[19] Currently, our OBPC algorithms involve only two parts, i.e., sender and receiver. They do not consider the third party communication such as agent *a* asks *b* to ask *c* for some information. Therefore, belief about what another believes about an observed action executed by the third agent is not included.

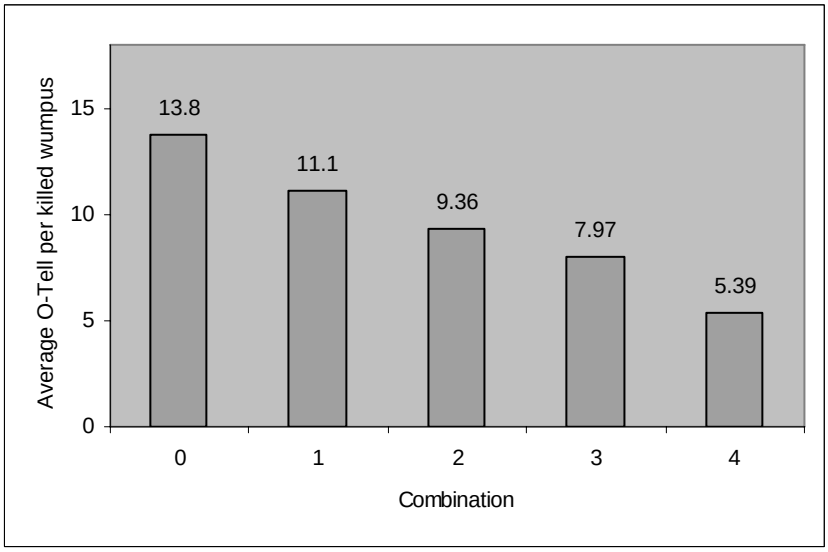
used Team B as a control condition against which to evaluate the effectiveness of different combinations of observability with Team A. We named Team B, without any of these beliefs, combination 0, since it involves none of the four beliefs. For Team A, we tested another 4 combinations of these beliefs to show the effectiveness of each, in terms of ACPWK. These combinations are:

0. Team B.
1. Team A with each agent's reasoning restricted to generating beliefs in category belief1. Then each agent believes properties it observes.
2. Team A with each agent's reasoning restricted to generating beliefs in categories belief1 and belief2. This allows the agent to reason what the doer believes the pre-cond of the observed action. This combination tests the effect of belief2.
3. Team A with each agent's reasoning restricted to generating beliefs in categories belief1, belief2 and belief3. Then the agent can reason what the doer believes the effect of the observed action. This combination tests how belief3 improves the situation.
4. Team A with all the ability to reason for all four belief categories. Then the agent addtionlly believes that the others sense some properties. This combination tests the effect of belief4 and shows the effectiveness of the beliefs as a whole.

Each combination was run in the five randomly generated worlds. The average results of these runs are presented in Fig. 7.2, in which one bar shows ACPWK for one combination.



(a) *O-Tell* protocol.



(b) *O-Ask* Protocol.

Fig. 7.2. Average Communication per Killed Wumpus in Different Combinations.

The first case, agents' belief1 (combination 1), is a major contributor to effective communication, for both *O-Tell* and *O-Ask*. As seen in (a), belief1 compared to combination 0 causes ACPWK to drop significantly for *O-Tell*, from 5.9 to 3.52. For *O-Ask*, in (b), ACPWK drops from 13.8 to 11.1.

The second case, belief2 (combination 2), does not produce any further reduction and hence is not effective for *O-Tell*, but belief2 does produce improvement for *O-Ask*. For *O-Tell*, when a provider senses an action, meaning the doer believes the precondition of the action, so the provider won't perform *O-Tell.* So for this example belief2 can be of little help in *O-Tell*. While for *O-Ask*, belief2 reduces ACPWK from 11.1 to 9.36, because with belief2, a needer will know who has a piece of information explicitly. Then it can *O-Ask* without ambiguity.

Third, for the similar reason that belief2 only works for *O-Ask*, belief3 (combination 3) contributes little to *O-Tell* but further decreases ACPWK to 7.97 for *O-Ask*.

Fourth, belief4 (combination 4) has a major effect on communications that applies to both protocols. It further drops ACPWK to 2.23 for *O-Tell* and to 5.39 for *O-Ask*. Belief4 is particularly important for *O-Tell*. For example, if the carrier believes that the fighters sense a wumpus' location, it will not tell them.

This experiment examined the contribution of each belief deduced from observability to the overall effectiveness of communication. The result indicates three things. First, belief1 and belief4 have a strong effect on the efficiency of both *O-Tell* and *O-Ask*. Second, belief2 and belief3 have weak influence on the efficiency of *O-*

Tell. Third, these beliefs work best together, because each of them provides a distinct way for agents to get information from the environment and other team members. Furthermore, they complement each other's relative weaknesses, so using them together better serves the effectiveness of the communication as a whole.

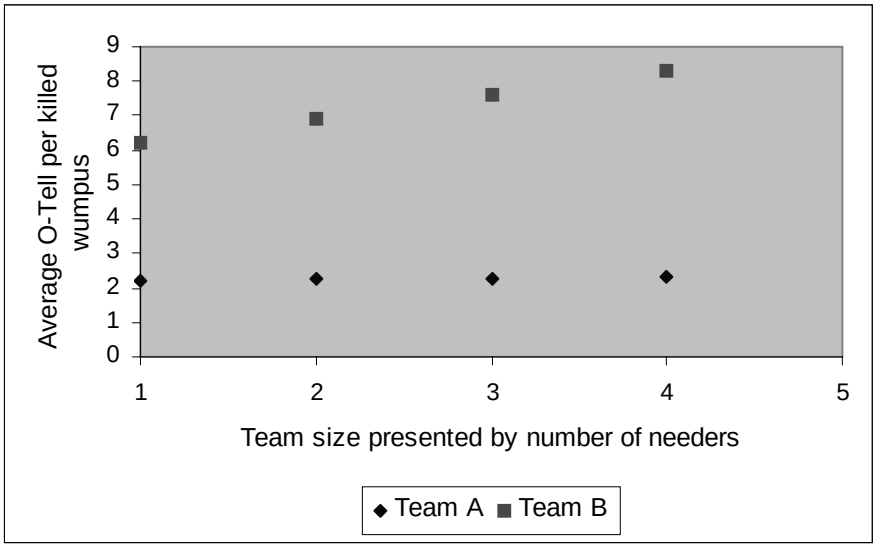
7.1.3.3. OBPC's Contributions to Team Scalability

We designed the third experiment to show how communication load changes with increased team size. *O-Ask* is directed to only one provider at certain time, while the *O-Tell* goes to all needers who do not have the information. So we assume that *O-Tell* brings more communication into play than *O-Ask*, and then we chose to test the *O-Tell* protocol. If the test results are good for *O-Tell*, we can expect that they are valid for *O-Ask* as well.

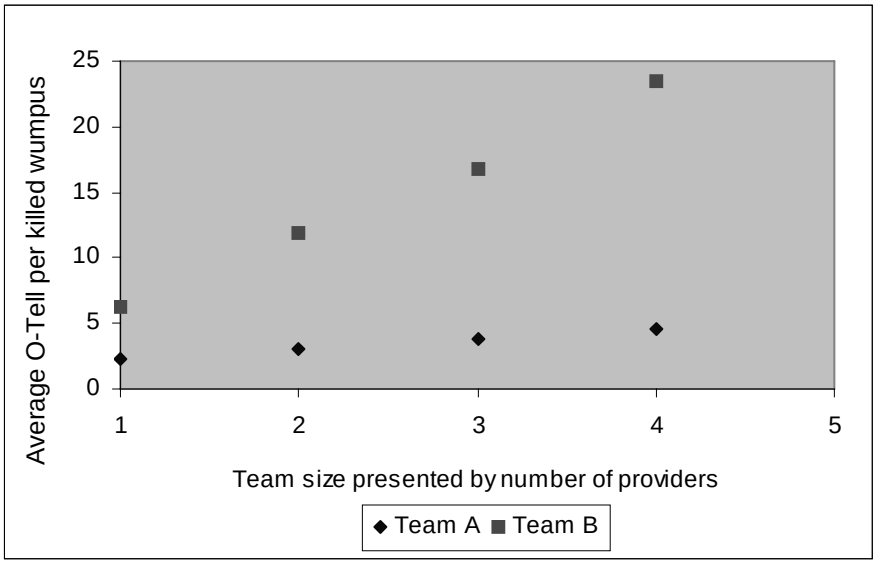
We used the same sensing capabilities for Teams A and Team B as in the first experiment. However, we increased the number of team members by 1, 2 and 3, in two tests that we ran. In the first test, we increased the number of needers, (i.e. fighters,) and kept the same number of providers, (i.e. carriers). In the second test, we did the opposite. In each test, for each increment and each team, we ran the five randomly generated worlds and used the average value of ACPKW produced in each world.

Fig. 7.3 shows the trend of ACPKW as a function of increasing team size. In (a), Team B has an obvious increase in ACPKW with increasing the team size. However, Team A the ACPKW remains the same. The trend can be attributed to two factors: first, the number of *O-Tell*s is held down because if the carrier believes the fighters can sense the wumpus, the carrier does not perform *O-Tell*; second, the more

fighters there are, the more wumpuses will be killed, which enlarges the numerator of ACPKW.



(a) Needers Increment.



(b) Providers Increment.

Fig. 7.3. The Comparison of *O-Tell* with Different Team Sizes.

In (b), increasing the number of providers breaks the constant trend in Team A and shows an increased ACPWK. However, comparing this increase to that of Team B, it is a moderate number. In Team B, incrementing the number of providers almost doubled the number of *O-Tell*s in every case. The communication load increased because different carriers duplicated the *O-Tell*s of other carriers. For example, each carrier always provides the wumpus' location to fighters when observing a wumpus. The carriers lack an effective way to predict when a piece of information is produced and by whom, which is one of main concerns for the empirical distribution function method for predicting information production and need and the decision theoretic method (see next Section 7.2). This experiment shows that ACPWK grows more slowly with increase of team size, in the team empowered with observability, which may indicate that observability will improve team scalability in some sense.

7.2. Evaluation of Proactive Communication

The experiments introduced in this section are overall evaluations of the Proactive Communication developed. We specify the form of utility function in the Multi-Agent Wumpus World domain. To show that Proactive Communication helps produce more effective interaction among agents in multi-agent teamwork, we design two other communication conditions, Always Tell and Always Ask. Experiments have been run under controlled by these three conditions. The results are presented that show the advantage of Proactive Communication in enhancing team performance as well as decreasing communication load.

To demonstrate Proactive Communication's ability of handling complex problems, we adjusted the existing Multi-Agent Wumpus World by adding more uncertainties and flexibility into it, as described in the next section.

7.2.1. Adjusted Multi-Agent Wumpus World

The world is 20×20 and has 4 wumpuses. The team is still composed of 1 carrier and 3 fighters. The team goal is to kill wumpuses. A complete MALLET team plan is given in Appendix B. The team is allowed to operate a fixed number of 5000 steps. We let the agents' observability region be an equilateral rhombus whose vertices are 7 (for carrier) or 1 (for fighters) in the X and Y directions from the agents' location.

We assume the agents know each other's initial location. At each time step, the carrier makes a random safe move to an adjacent location not previously visited within the last 50 steps if possible. If there is no such location, the carrier will visit the least recently visited adjacent safe location. At the start of a trail, each fighter remains at its initial location until it receives a wumpus location from the carrier. Similarly, after killing a wumpus, a fighter will stay at the location where it killed a wumpus until it receives a new wumpus' location from the carrier. The carrier chooses the first fighter the carrier evaluates, if two or more fighters have the same utility.

To facilitate estimating the time duration until the next information need (in this implementation, the time it takes for a fighter to kill a wumpus) and the next information production (the time of finding a wumpus), the fighters and carriers exchange information about the times at which they killed or found a wumpus. However, this exchange of information is included as part of tell or ask messages, and

is not sent separately. For example, for each wumpus found, the carrier records the time at which the wumpus was found and attaches such historical data to messages sent to the fighters. Fighters attach the times at which they received wumpus locations and killed the corresponding wumpuses with each active-ask. In addition, the location of the sender is also included in messages sent. So the carrier knows the fighters' present locations, which are the last wumpuses' locations the carrier told the fighters, and the fighters know the carrier's location (may be present or not) from the message sent by the carrier. Though the fighters may not have the carrier's present location since the carrier keeps moving all the time, the carrier may frequently contact the fighters and attaches its location if proactivity is fully enabled.

The wumpuses periodically jump to some other random location from time to time after their first appearance. There is no limit for how far they can jump (other than that they cannot jump outside of the world). The length of time they stay at their current locations is randomly generated from 1 to 40 steps (agents are assumed know this range). There is a new wumpus born at a randomly chosen location on the step after one has been killed, allowing us to maintain a constant number of wumpuses in the world.

When a wumpus jumps before the fighter arrives at the wumpus' location and kills it, we consider a new need to arise. The fighter will continue to move to the location at which it was told the wumpus was, and then stay at the wumpus' location and make a new decision.

Each wumpus is assigned a hearing rhombus of radius 8 when it is generated. The wumpuses have a probability (the same for all wumpuses) of hearing sounds (messages) within their radius. A wumpus does not always hear messages because it does not always focus on hearing (e.g., it sleeps some). However, once a wumpus hears a message (sent either by a carrier to a fighter or vice versa), it will be alerted. If the message is from the carrier to a fighter, the wumpus can tell whether it has been identified. If the message is a request from a fighter to the carrier, the wumpus will focus on the coming reply from the carrier. But, if the wumpus has jumped before the time at which the reply is sent, we assume it no longer pays specific attention to the message emanating from a carrier within its hearing radius. Agents are assumed to know the wumpus' hearing radius and the probability of hearing a message, but they may not know whether or not the message sender is within the wumpus' hearing radius. The wumpus also has short sensing ability to an adjacent cell. Once a wumpus is alerted by a message and identifies itself as the target, it can sense the adjacent fighter and starts to fight with the fighter. The wumpus has a probability of winning the fight. Agents are also assumed to know this probability. One can see that, if more information has been sent, chances are greater that the fighters may be killed, and consequently fewer wumpuses will be killed in the limited length of time. Moreover, the game may be forced to end before the time limit if there are no fighters left.

We assume that the wumpuses are distinguishable. This means that the carrier can determine whether it is sensing a wumpus for the second time (after a series of moves by the carrier) or is sensing a second wumpus. While the problem where the wumpuses

are indistinguishable is very interesting, it focuses more on the planning and reasoning required and obscures the principal issues of communication being addressed here. So we focus on the case in which the wumpuses are distinguishable in this implementation.

7.2.2. Problem Analysis

Information to be communicated in the team consists of conjuncts which are part of the precondition of a plan or an action and their value are unknown. For this example, the information is "wumpus location".

A list of needers and a list of providers for a given piece of information are generated by analyzing the preconditions and effects of plans and actions [135]. For this example, the provider is the carrier, and the needers are the fighters. We removed the domain-specific constraint which says the closest fighter will be assigned a found wumpus' location (used by Yin [137]). Instead, the assignment is decided by utility function in determining communication policies.

Challenges come from the problem of *when* the information will be provided or asked for. It is unnecessary that all needers be told about "wumpus location" when the location is discovered or that the provider always be asked about "wumpus location" when the location is needed. The decisions about whether or not to communicate depend on two kinds of knowledge: agents' observabilities and time points at which the information is produced or needed. Observabilities are useful in the case that one agent can deduce others' beliefs from what it has sensed. For example, if the carrier believes the fighter also can sense the wumpus' location that was discovered by the carrier, the

carrier does not need to tell a fighter about this location, or if a fighter can find wumpuses by itself, the fighter does not need to ask the carrier for this information. However, knowing *when* the information is needed or produced is very important in the system where agents' observabilities are limited. For example, in this domain, because of the fighters' limited observabilities, it is very likely that the fighters have to obtain the "wumpus' location" information via communication. The communication could be either the carrier proactively telling the information to the fighters, or the fighters actively requesting it from the carrier.

However, knowing when the information is needed or produced is sometimes impossible, because the domain has uncertainties. The uncertainties come from two aspects. First, since the carrier moves randomly, and since the wumpuses appear randomly, the time duration needed for finding and killing the wumpuses would not be fixed and cannot be precisely calculated. Second, the agents' decisions are not fixed and may vary in different situations. For example, the more up-to-date the information the fighters receive, the better their chance of locating the wumpuses. Nevertheless, since the wumpuses do not move before the next jump, a piece of old information may also be useful to the fighters, in the case that the fighters do not have the most recent one.

7.2.3. Determining the Form of the Utility Function

The utility function is composed of risk, cost, timeliness, and currency functions and those functions have been defined on a higher level in Section 6.4. In this implementation, they are given specific forms.

7.2.3.1. Risk Function

Communication incurs risk in the Multi-Agent Wumpus World. The risk is defined as the potential loss of wumpuses that a fighter would kill resulting from the possibility that a fighter may be killed if a wumpus overhears the message delivered (no matter by a carrier to the fighter or by the fighter to a carrier). This is because only fighters are able to kill wumpuses and consequently if the wumpus has been alerted by the communication and kills a fighter, the loss of the fighter will seriously degrade the team's ability to kill wumpuses in the future. On the other hand, because of the carrier's large observability radius, it may sense wumpuses far enough away that there is a much lower chance of the carrier being killed; only if a wumpus is generated in the same cell as the carrier will the carrier be killed. The probability of this occurring is much lower than that of a message being overheard, and thus, we do not consider the possible loss of a carrier in the risk function.

There are two cases to consider: 1) the wumpus overhears a carrier, and 2) the wumpus overhears a fighter. In the case that the carrier is the sender, the risk is directly associated with the message about wumpus whose location was found.[20] However, in the case that a fighter is the sender, the message sent will not cause a risk directly, because the message is just a request and wumpuses are unable to tell who will be the target by hearing it. Instead, the risk is associated with the reply sent by the carrier to the request. Therefore, for both cases, the risk is associated with the messages sent by

[20] Though multiple wumpuses may be discovered simultaneously, the carrier will make decisions regarding to each of their locations. Therefore at each decision point, we consider one wumpus discovered.

the carrier. The fighter who may be killed is the one to be told about the wumpus location and the wumpus that might kill the fighter is the one whose location was transmitted. Although unobserved wumpuses may hear the message as well, we assume they will not cause the risk since they are able to identify that they are not the target so they will not fight with the fighter.

The risk function has been defined as $R(e, S_t, \delta)$ where e is the agent's internal state, S_t is the agent's situation at the time t of decision-making, δ is the policy under consideration (see Section 6.4.1). In this domain, we assume the risk of communication is associated with the number of steps left in a game, and two probabilities Pr_h and Pr_f:

$$R = k \times (5000 - t) \times Pr_h \times Pr_f \qquad (7\text{-}1)$$

where k is the number of wumpuses a fighter can kill per unit time, t is the number of steps passed, Pr_h is the probability that the wumpus hears the message sent by the carrier, and Pr_f is the probability that the wumpus can win against the fighter.

To give a number to k for initial tests, we ran the system in a trial mode and k was estimated by the data collected from previous test runs[21]. We learned that the average number of wumpuses a fighter can kill per step is 0.01. We then set

$$k=0.01.$$

Regarding to Pr_f, we learned that if Pr_f is set to a number like 0.2, it is usually the case that all fighters died before the game ends at 5000 steps. We also learned that setting Pr_f to a low number such as 0.05 will limit the effect of risk factor. We thus set

[21] It would be possible to obtain estimates for k dynamically as a game progresses, using EDF in a manner similar to that used for other aspects of the process.

$$Pr_f = 0.1.$$

The last value to compute is Pr_h. Computing Pr_h is more complex and dynamic than computing the other variables. Suppose the area of the world is O, and r_c and r_f represent observable rhombus vertex distance from the carrier and the fighter, respectively. Further, we assume that the wumpus is able to hear messages sent by an agent within a rhombus with vertex distance of r_w from the wumpus. However, the wumpus may or may not be paying attention to messages (e.g., it might be asleep). We are interested in the probability that the wumpus does hear and take action on a message it intercepts. We thus denote the probability that the wumpus "pays attention" to a message, given that it can hear the message, by Pr_a. We use Pr_r to denote the probability that an agent is within the hearing range r_w. Then Pr_h, the probability of hearing a message, is the product of Pr_r, the probability that an agent is within the wumpus' hearing range r_w, and Pr_a, the probability that the wumpus pays attention to the message:

$$Pr_h = Pr_r \times Pr_a.$$

There are two cases to consider for Pr_a: 1) the wumpus is "unalerted" by the request from the fighter, and consequently will not pay attention to the coming reply; and 2) the wumpus is "alerted" so will pay attention to the coming reply. For the "unalerted" status, since the wumpus may or may not focus on hearing at any time, we assume Pr_a is 0.1:

$$Pr_a = 0.1.$$

One can expect that the consideration of Pr_a for the "alerted" status is more complex. Therefore in the form $Pr_h=Pr_r \times Pr_a$, we focus on Pr_r and Pr_a for the "alerted" status. We consider $r_c < r_w$, which is our current setting[22]. We analyze Pr_h separately for the case that the carrier is proactively telling a message or that a message is a reply sent by the carrier regarding to a request from a fighter. Then the risk will take place for three of carrier's policies: *ProactiveTell*, *Reply* and *WaitUntilNext*.

Table 7.2 shows Pr_h, the probability that the wumpus hears the message sent by the carrier, for these policies (calculation details is presented in Appendix C). Once Pr_h is computed, risk can be easily computed by Eq. (7-1). The difficulty is that there are two unknown parameters in this table, H and D_n. H is the length of time between the last time the carrier was the wumpus and the time the wumpus will jump. D_n is the time between the time the carrier last saw the wumpus and the current time. These parameters must be estimated. The methods of estimating these parameters are also given in Appendix C.

[22] The calculation of Pr_h in $r_c \geq r_w$ would be simpler than the current setting, in that the carrier can explicitly calculate whether or not the wumpus is able to hear the message. So we only consider $r_c < r_w$ in experiments.

Table 7.2. Pr_h in Risk Function for Different Policies.

Policy \ Probability			Pr_h
ProactiveTell			0.1
Reply	Carrier still see wumpus	Fighter was within r_w	0.19
		Fighter was not within r_w	0.1
	Carrier not see wumpus	Fighter was within r_w	$0.19\times E\{Pr(D_n<J)\}+\frac{(2r_w^2-2r_w-1)-(2r_c^2-2r_c-1)}{O-(2r_c^2-2r_c-1)}\times 0.1\times(1-E\{Pr(D_n<J)\})$
		Fighter was not within r_w	$0.1\times E\{Pr(D_n<J)\}+\frac{(2r_w^2-2r_w-1)-(2r_c^2-2r_c-1)}{O-(2r_c^2-2r_c-1)}\times 0.1\times(1-E\{Pr(D_n<J)\})$
WaitUntilNext			$0.1+0.09\times\frac{(2r_w^2-2r_w-1)-(2r_f^2-2r_f-1)}{O-(2r_f^2-2r_f-1)}$

7.2.3.2. Cost Function

We have defined the cost as the following form in Section 6.4:

$$C(\{m\})=\begin{cases}0 & \text{if } \{m\}=\varphi\\ k_0+k_1\times len(\{m\}) & \text{otherwise}\end{cases}$$

where k_0 and k_1 are coefficients and $\{m\}$ is a set of messages used by a policy. Empirical data shows that k_0 is a base cost of sending a message and k_1 is a parameter which adjusts the effect of the message size to the cost [134]. Typically, k_0 is much larger than k_1. Therefore, we assume that $k_1=0$. Communication is generally considered to be cheap in the present time. Hence we also set $k_0=0$. According to this,

$$C(\{m\})=0.$$

7.2.3.3. Timeliness Function

At the abstract level, the timeliness is represented by two functions: f_s, the value of timeliness under the condition that the most recently produced information is sent, and f_f, the value of using old information (see Section 6.6.1 for detail).

f_s has been defined as a non-increasing function of a time difference d between t_n, the time at which an information item is needed, and t_p, the time at which the information provided is produced:

$$f_s(d(t_n, t_p))$$

where $d=\max(0, t_p-t_n)$. If the fighters are able to receive wumpuses' locations as quickly as possible, they may catch more wumpuses before the wumpuses jump and kill them. Thus, the timeliness loss is the number of wumpuses a fighter can kill per unit time multiplied by the delay time. Accordingly, we assume the form of function f_s in this example as:

$$f_s=\begin{cases} 0 & \text{if } t_p < t_n \\ k(t_n - t_p) & \text{otherwise} \end{cases}.$$

The rationale for this form of timeliness is that the further in the future the value used will be produced, the more likely the more opportunity to kill wumpuses is lost while waiting.

f_f generally is domain dependent and can be determined on many different bases. In this domain, the wumpuses periodically jump to some other random location and the time duration of staying on one place is also random. Therefore, once a

wumpus jumps (meaning the value for *I* is changed), there is no gain for a fighter to chase the old wumpus' location. Hence the value for communicating the old location is zero:

$$f_f = 0.$$

7.2.3.4. Concurrency Function

On the highest level, the currency has been defined as a probability function P:

$$\Pr(\neg\exists\ \tau \in \mathrm{Int}(t_p, t_u] \ni I_P(\tau) \mid S_t \wedge \delta),$$

which means that no other value is produced between $(t_p, t_u]$, conditional on a policy δ which is chosen by agents in the situation event S_t at time t (see Section 6.6.2).

The currency function of this implementation is not exactly the same because of the unique characteristics of the domain. However, it rests on the same idea worked out in Section 6.6.2. At the highest level, P uses t_u, which is the time at which $I_P(t_p)$ is used by the needer. But we did not specify whether t_u denotes when the use of $I_P(t_p)$ begins or ends. In this domain, it is important for a fighter to arrive at a found wumpus' location before the wumpus jumps. Therefore t_u represents the end use of $I_P(t_p)$. Hence in this domain, t_u is the time at which the fighter arrives at the wumpus' location. We assume P represents the probability that the wumpus does not jump between the time interval $\mathrm{Int}(t_p, t_u]$:

$$\Pr(\neg\exists\ \tau \in \mathrm{Int}(t_p, t_u] \ni \text{wumpus jumps at } \tau \mid S_t \wedge \delta).$$

For simplicity, P is represented as:

$$\Pr(\mathrm{WNJ}(t_p, t_u)),$$

where the event "wumpus not jump between Int(t_p, t_u]" is abbreviated as WNJ(t_p, t_u).Though the probability function P is computed on a general level in Section 6.7, Pr(WNJ(t_p, t_u)) must be reconsidered since it is a specific form for this domain.

In the following sub-sections, we describe the general idea of computing Pr(WNJ(t_p, t_u)).

7.2.3.4.1. General Ideas

We need to determine the probability of a wumpus not jumping between t_p, the time at which the wumpus location provided to the needer is produced, and t_u, the time at which the fighter arrives at that location. This probability equals:

$$\Pr(\mathrm{WNJ}(t_p, t_u))=\Pr(t_u-t_p<J), \tag{7-2}$$

where J denotes the difference between the time at which the value for *I* was produced and the time at which the wumpus jumps. J must conform to this constraint:

$$J\in[1, 40-D_0],$$

where D_0 ($D_0\in[1, 40]$) denotes the length of time that the wumpus was in its current location before being sensed by the carrier.

Eq. (7-2) depends upon a number of parameters, the obvious ones being D_0, J, t_u and t_p. However, it also depends upon $L_w(t_p)$, and $L_f(t_n)$, where L_w and L_f refer to the locations of the wumpus and the fighter respectively because these locations determine the distance between the fighter and the wumpus, and hence the time required for the fighter to move to the wumpus. In addition, for some possible values for L_w, there is a dependence upon the location of the carrier, L_c. Let t be the current time at which the evaluation is being performed. One important notational issue is that, rather than

referring to locations at specific times, L_w and L_c refer to locations at t_p while L_f refers to the location at t_n.

Depending upon the communication policy for which $\Pr(t_u - t_p < J)$ is being calculated, several of the parameters are known. Nevertheless, in each case some parameters remain unknown. In some of these cases, the probability distributions of the values of unknown parameters can be obtained through the EDF methodology described in Section 5.3; in others, the probability distributions can be determined easily from the environment description. In a few cases, it is possible to make reasonable estimates of the distributions. Thus, we treat the unknown parameters as random variables and use the law of total probability to calculate $E\{\Pr(t_u - t_p < J)\}$ with respect to the unknown values:

$$\sum_{d}\sum_{j}\sum_{\tau_1}\sum_{\tau_2}\sum_{l_c}\sum_{l_w}\sum_{l_f} \Pr(t_u - t_p < J \mid D_0 = d, J = j, t_p = \tau_1, t_n = \tau_2, L_c = l_c, L_w = l_w, L_f = l_f) \times \Pr(D_0 = d, J = j, t_p = \tau_1, t_n = \tau_2, L_c = l_c, L_w = l_w, L_f = l_f) \quad (7\text{-}3)$$

The parameters all have discrete values. Some of the distributions are joint. Some of the random variables are independent, e.g., D_0, and their distributions can be determined separately. Though theoretically, the time parameters have infinite ranges, in practice, the EDF distributions used as approximations will have only finite ranges.

Some notes can be made regarding parameters in this expression. First of all, J has a relationship to D_0, i.e., $J \leq 40\text{-}D_0$. The carrier can sometimes decide whether $D_0=0$ or not by analyzing what it has observed. The carrier moves only one step every time. So a newly found wumpus should be on the edge of the carrier's observability

radius. If this is not true, the carrier can then decide that this wumpus just jumped into the carrier's observable area and hence D_0=0. The judgment also can be made the other way around. If a found wumpus that is not on the edge suddenly disappears on current observation, the wumpus jumped, or if a found wumpus is not in the same position as a previous observation, the wumpus jumped. Remember, we assume wumpuses are distinguishable, so the carrier is able to track each wumpus' position and often use this knowledge to deduce D_0.

Handling the case in which D_0=0 is straightforward; the first summation in expression (7-3) simply drops out and d =0 is used in the rest of the equation. As this is a simple modification of the general result, we assume $D_0 \neq 0$ in the rest of our analysis. Therefore, the probability mass function for d in expression (7-3) is simply a constant equal to 1/40, and is independent of all of the other variables. J has a relationship to D_0, i.e., $J \leq 40 - D_0$; hence the mass function for j may be treated as 1/(40−d), and it is then independent of all of the remaining variables.

We can do statistical estimations for t_n and t_p based on their EDF distributions. As for L_c, L_f and L_w, determination of exact distributions is unduly computational complex because of the limited observability of agents and the random walk of agents. Hence we need to seek a reasonable and computationally feasible approximation.

7.2.3.4.2. A Study of the General Case

To elaborate the determination of the probability, we examine the most general case and analyze the process of calculating expression (7-3). This case covers all of the

situations, policies and sub-cases that will be considered later. All other cases are either subset of this case or variations that can be easily adapted from this one.

The general case is that, the decision maker (either the carrier or the fighter) does not know L_w, L_c, and L_f. The estimation for L_w can base on L_c because the wumpus must be inside of the carrier's observation area. Since L_c is also unknown (this happens even if the carrier itself is the decision maker as the carrier makes estimation to future decisions), the decision maker could use the most recent location of the carrier it knows and calculate the range of possible motions. The statistical estimations for t_n and t_p will be used to estimate this range. To estimate L_f (the fighter needs not do that since it does not need to estimate its own future decisions), the carrier can take advantage of wumpus locations sent to the fighter because the fighter won't move before receiving a new location. Having estimations for L_w, L_c, and L_f, the distance between the wumpus and the fighter can be determined. Combining this distance with the knowledge about D_0, J, t_p and t_n, expression (7-3) can be computed.

Since either t_n or t_p may be unknown, depending upon whether it is the carrier or the fighter that is making a decision, the carrier may move between the last point at which its locations were known to the decision maker (either the carrier or the fighter) and the time t_p. In order to calculate an approximation for the desired value, we will need to consider the range of possible motions from some known points in time. In order to be able to refer to these, we use the following notation:

t_{cl} = the most recent time at which the decision maker knows the location of the carrier.

Let σ be the length of time the carrier moves from the time t_{cl} to t_p, the time at which it finds the wumpus. The reason of introducing σ is that in some cases, L_c is unknown, so we need to estimate L_c, based on distance the carrier moved from a known time point t_{cl} to t_p. So σ must be greater than 0, i.e.,

$$\sigma = \max(t_p - t_{cl}, 0).$$

The actual values to be used for σ will vary with the situation, policy and decision maker being considered. However, once these values are known, the desired value can be approximated.

Next we analyze the length of time it takes the fighter to move to the wumpus' location. We denote this length of time as D_k. D_k is equal to the distance between the fighter and the wumpus, since the fighter moves one step toward the wumpus at each time step,

$$D_k = |X_f - X_w| + |Y_f - Y_w|$$

where X_f, Y_f, X_w and Y_w denote the x and y coordinate positions for the fighter and the wumpus ($L_f = (X_f, Y_f)$, $L_w = (X_w, Y_w)$). X_f, Y_f, X_w and Y_w must conform to these constraints:

$$1 \le X_w \le 20, \quad 1 \le X_f \le 20,$$

$$1 \le Y_w \le 20, \quad 1 \le Y_f \le 20.$$

Given a number of steps, j, after which the wumpus will jump, the fighter must also be able to reach the wumpus within j steps. As a first step in calculating $Pr(t_u - t_p < J)$, we calculate the probability that given j and a wumpus location, $L_w = (X_w, Y_w)$, the fighter can reach the wumpus before it jumps (recall that the fighter does not move

without knowing a wumpus location). We define a function f_m which indicates whether or not a fighter's location L_f is within j steps from the L_w:

$$f_m(X_w, Y_w, X_f, Y_f, j) = \begin{cases} 1 & \text{if } |X_w - X_f| + |Y_w - Y_f| \le j \\ 0 & \text{otherwise} \end{cases}.$$

To determine D_k, we need to know the wumpus' location L_w which in some cases depends on the carrier's location L_c, since L_w must be inside the carrier's observability area. This region is an equilateral rhombus whose vertices are observability radius r_c in the X and Y directions from L_c; the area of this rhombus is $2r_c^2+2r_c+1$. We assume the wumpus will be randomly located in the observable area of the carrier[23]. Then, given the carrier's location (X_c, Y_c) at time t_p, the probability that the wumpus is inside of the carrier's observability range, and meanwhile can be reached by the fighter within j steps, is:

$$P_w(X_c, Y_c, j)$$

$$= \sum_{X_w=\max(1, X_c-r_c)}^{\min(20, X_c+r_c)} \sum_{Y_w=\max(1, Y_c-(r_c-|X_w-X_c|))}^{\min(20, Y_c+(r_c-|X_w-X_c|))} \frac{1}{2r_c^2+2r_c+1} \times f_m(X_w, Y_w, X_f, Y_f, j).$$

Next, we consider the effect of possible carrier's motion. Since we know that at time t_p, the carrier must be able to sense the wumpus, the wumpus must be located within the region to which the carrier could have moved from its last known location, denoted L_{cl}, extended by the observability region. L_c must be inside the region which is reachable from L_{cl} within σ steps. This region is also an equilateral rhombus whose radius is σ and center is L_{cl}, and its area is $2\sigma^2+2\sigma+1$.

[23] While this is not precisely correct, it can be shown to be a good approximation.

We represent the probability that the carrier is at a location L_σ after σ steps from L_{cl} by $\Pr(L_\sigma)$. Then the probability that the carrier is within σ steps from (X_{cl}, Y_{cl}), and the wumpus is inside of the carrier's observability range and can be reached by the fighter within j steps is:

$$P_c(X_{cl}, Y_{cl}, j)$$

$$= \sum_{X_c=\max(1, X_{cl}-\sigma)}^{\min(20, X_{cl}+\sigma)} \sum_{Y_c=\max(1, Y_{cl}-(\sigma-|X_c-X_{cl}|))}^{\min(20, Y_{cl}+(\sigma-|X_c-X_{cl}|))} \Pr(L_\sigma = (X_c, Y_c)) \times P_w.$$

Finally, combining all of these, with J, D_0 as random variables,

$$E\{\Pr(t_u - t_p < J)\}$$

$$= \sum_{d=1}^{40} \frac{1}{40} \times \sum_{j=1}^{40-d} \frac{1}{40-d} \times$$

$$\sum_{X_c=\max(1, X_{cl}-\sigma)}^{\min(20, X_{cl}+\sigma)} \sum_{Y_c=\max(1, Y_{cl}-(\sigma-|X_c-X_{cl}|))}^{\min(20, Y_{cl}+(\sigma-|X_c-X_{cl}|))} \Pr(L_\sigma = (X_c, Y_c)) \times$$

$$\sum_{X_w=\max(1, X_c-r_c)}^{\min(20, X_c+r_c)} \sum_{Y_w=\max(1, Y_c-(r_c-|X_w-X_c|))}^{\min(20, Y_c+(r_c-|X_w-X_c|))} \frac{1}{2r_c^2+2r_c+1} \times f_m(X_w, Y_w, X_f, Y_f, j) \quad (7\text{-}4)$$

In Eq. (7-4), though we use several layers of summation, the computational complexity is not as high as it appears because ranges of the variables are limited. In addition, in many of the sub-cases that must be evaluated, $t_n > t_p$, which means that information *I* was produced a while before the fighter needs it, will further increase the lower bound on the values of j that are possible, and this in turn, will decrease the upper bound on the range of possible values of d. Hence the range of the summations over j and d is further reduced. Moreover, in many of the sub-cases, one or more of the variables are known, and the expression can be reduced (we will consider these sub-

cases separately in Section 7.2.4). However, Eq. (7-4) depends upon one unknown probability mass functions, $Pr(L_\sigma = (X_c, Y_c))$. In order to use it, one must either determine this mass functions or approximate it in some way.

Determining the exact value of these mass functions is difficult. In the absence of a theoretical solution, which we have not been able to find, there are several different ways of approaching the problem of determining values to use for these mass functions. For example, one could run a large number of Monte Carlo simulations to determine them. Alternatively, one can make some simplifying approximations. These will be considered in the next section.

7.2.3.4.3. Approximations for $Pr(L_\sigma)$

One probability mass function appears in Eq. (7-4), that is $Pr(L_\sigma=(X_c, Y_c))$, the probability that the location to which the carrier might have moved is within σ steps from a known location. In the following, we propose two approximations for $Pr(L_\sigma=(X_c, Y_c))$.

7.2.3.4.3.1. General Approximation

A trivially simple approximation would be that, having no information about the carrier's location, we assume that the carrier is randomly placed in the area which is within σ steps to its location at t_{cl}. In other words, $Pr(L_\sigma=(X_c, Y_c))=1/(2\sigma^2+2\sigma+1)$, which means that all points inside of the area are reachable with equal probability. However, clearly the points further away are less likely to be reached than those closer to the carrier's location at $t_{cl,}$ because the carrier's motion conforms to random walk; we can take advantage of the nature of the random walk.

7.2.3.4.3.2. Simplified Approximation

This assumption uses some results of random walk to generate a simple approximation to the probability $\Pr(L_\sigma = (X_c, Y_c))$. [126] shows that without our restriction of not revisiting a place, the average distance the carrier has gone within σ steps would be $\sqrt{\sigma}$. Note that $\sqrt{\sigma}$ will not be an integer. Then, one might use an area with a radius equal to the expected value of the distance of movement. Therefore, we will use $\sqrt{\sigma}$ as the expected distance the wumpus will move in σ steps, and approximate the distribution by a constant within an equilateral rhombus whose vertices are $\sqrt{\sigma}$ in the X and Y directions from the last wumpus location. That is, we could take $\Pr(L_\sigma = (X_c, Y_c))=1/(2\sigma+2\sqrt{\sigma}+1)$.

7.2.3.4.3.3. Finalizing Pr(P_σ)

While the *Simplified Approximation* is still a coarse approximation, it is likely to be better than the *General Approximation* because it does capture, in some sense, the feature of random movement of carrier. Therefore we use *Simplified Approximation*.

Using the *Simplified Approximation*, Eq. (7-4) is modified to:

$$E\{\Pr(t_u - t_p < J)\}$$

$$= \sum_{d=1}^{40} \frac{1}{40} \times \sum_{j=1}^{40-d} \frac{1}{40-d} \times$$

$$\sum_{X_c=\max(1, X_{cl}-\sqrt{\sigma})}^{\min(20, X_{cl}+\sqrt{\sigma})} \sum_{Y_c=\max(1, Y_{cl}-(\sqrt{\sigma}-|X_c-X_{cl}|))}^{\min(20, Y_{cl}+(\sqrt{\sigma}-|X_c-X_{cl}|))} \frac{1}{2\sigma+2\sqrt{\sigma}+1} \times$$

$$\sum_{X_w=\max(1, X_c-r_c)}^{\min(20, X_c+r_c)} \sum_{Y_w=\max(1, Y_c-(r_c-|X_w-X_c|))}^{\min(20, Y_c+(r_c-|X_w-X_c|))} \frac{1}{2r_c^2+2r_c+1} \times f_m(X_w, Y_w, X_f, Y_f, j) \quad (7\text{-}5)$$

which needs three inputs: σ, L_{cl} and L_f.

The general case will be referred frequently by later cases, so we abbreviate it as *GenrealCase*. Based on it, we can compute $Pr(t_u - t_p < J)$ for each situation/policy combinations. In Appendix D, we study $Pr(t_u - t_p < J)$ for each situation/policy combination needed to apply our decision theoretic model. For specific combinations, several variables have known or estimatable values, reducing the number of summations of Eq. (7-5), and the probability mass function usually take on a simple form. Moreover, the ranges of the summations are related in many cases, further reducing the complexity of the computation.

7.2.4. EDF Implementation Issues

In Section 5.5, we discussed implementation issues related to our EDF approach and proposed various possible solutions for each issue. In this section, we specify the solutions that are feasible in our domain.

7.2.4.1. System Initialization

We assume that needers have needs at the beginning of a trial, i.e., they are ready to fight wumpuses, but have no knowledge of wumpus locations (except in the low probability event that a wumpus is initially adjacent to a fighter). We ran the system in a trial mode and collected data of information production or need time intervals. We used these data to create initial values for the need and production intervals; these are replaced with EDF generated values when the agents begin to collect current data. Therefore agents are able to predict time points of productions or needs as soon as the system starts.

7.2.4.2. Preventing Having History Starvation and Communication Deadlock

A fighter could wait a long time if the carrier's estimate of the fighter's need time is far off due to limited data. Communication also could be in deadlock if both needer and provider each wait for a message from the other. In both cases, agents need secondary decisions. Our approach is to set a time cutoff T_o=80 steps. If a fighter has being waiting beyond this time, it will send out a request to the carrier and attach its historical data. It is unnecessary for the carrier to initiate the contact or to make contact whenever the delay expires, because it is important that the carrier have the historical data about the fighter's need, adjust its own prediction, and help the fighter proactively.

7.2.5. Experiments

We report two experiments. The first validates the systems development and the second evaluates the effectiveness of Proactive Communication.

7.2.5.1. Comparison Conditions

We will compare our approach, Proactive Communication, with other two approaches: Always Tell and Always Ask. All other settings are the same for each of the three test conditions except communication policies being used, which is described below.

7.2.5.1.1. Always Tell Condition

The different location for each distinguishable wumpus will be told when the carrier observes it. The carrier's decision-making is based on each fighter's last location. Initially, this is each fighter's initial location. Later, it is the location of the last wumpus to which the fighter was directed. It will assign the wumpus to the nearest

fighter on that basis. The fighters do not issue communication throughout the game and will stay at the wumpuses' locations if the wumpuses have jumped before the fighter's arrival. If the fighters have more than one wumpus location, they will use the most recent one.

7.2.5.1.2. Always Ask Condition

Since we assume needers have needs at the beginning, all fighters will ask the carrier for wumpuses' locations at the first step. After that, each fighter will send out a request once it finishes killing a wumpus about which the carrier told it. When the fighter arrives at the location told but does not sense the wumpus, it will ask the carrier again and stay at this position until it receives other information.

Rules of the carrier's reply are: 1) if the carrier has one request and one information item available, it will assign the information to that fighter who sent the request; 2) if the carrier has multiple requests and multiple information items, it will assign each location to the nearest requesting fighter; initially, each fighter's location is its initial position, and later, it will be the location of the last wumpus it killed; 3) if the carrier has multiple requests and one information item, it will send that information item to the fighter that made the earliest request; 4) if the carrier has no information item available when it receives a request, the reply will be deferred to the time at which an item is produced; 5) if the carrier receives no request at the time of production, it will save this information and will provide it to the next request. If multiple information items are saved, the providing is based on the order from the most recent item to the old one.

7.2.5.1.3. Proactive Communication Condition

For both carrier and fighters, decisions about communication policies depend on the utility of the policies. The carrier makes decisions every time it finds a different location for each distinguishable wumpus, whether or not this wumpus' previous locations were sent. The fighters make decisions every time they finish killing the wumpuses about which the carrier told them. In the case that the fighters do not see the wumpuses when arriving at the locations indicated, a new need will raise and the fighters need to make new decisions at this time.

7.2.5.2. Experiment Data

The following data were collected from experiments:

WK: the number of wumpuses killed;

WF: the number of wumpuses found;

WT: the total number of wumpuses generated;

AL: the number of agents left alive;

FK: the number of fighters killed;

MT: the total number of messages exchanged;

ST: the total number of steps a game runs before the end (the game may be forced to end before 5000 steps if all fighters died).

7.2.5.3. Measurements

We measure the effectiveness of Proactive Communication, Always Tell and Always Ask based on the elements listed in Table 7.3.

Table 7.3. Experiment Measurements.

Measurement	Formula	Criterion
Metric1	$\frac{FK}{WK} \times 1000$	The lower the better
Metric2	Metric1 × MT	The lower the better
Metric3	AL × WK	The higher the better
Metric4	$\frac{Metric3}{MT}$	The higher the better

We measured team performance from two aspects: loss and gain. Metric1 and 2 regard the former and Metric3 and 4 regard the latter.

Metric1 presents the loss ratio of FK (fighter killed) vs. WK (wumpus killed). Since FK may be much less than WK, we amplify the ratio 1000 times. We expect fewer fighters dead but more wumpus killed, so the lower Metric1, the better team performance.

In Metric2, MT (message total) is added to Metric1 as a factor. Metric2 combines effects of loss ratio and communication load. We expect the low loss ratio and the low communication load. So a low Metric2 is desirable for an effective team.

Metric3 measures team performance from a gain aspect. It evaluates AL (agent alive) and WK (wumpus killed). Obviously high Metric3 means effectiveness.

Metric4 adds MT (communication amount) to Metric3 as a numerator, meaning that we expect more alive agents and killed wumpuses, but less communications.

7.2.5.4. Experiment Basics

We used three teams, Team PC (using Proactive Communication), Team AT (using Always Tell) and Team AA (using Always Ask). Except for the communication conditions, settings of all teams were exactly the same.

We ran 30 randomly generated worlds under each condition. We use statistical t-test to test means of results for three teams. The t-test is often used to assess the equality of a pair of means by using the formula as follow [42]:

$$P = \frac{\text{mean1} - \text{mean2}}{s}$$

where *s* is a measure of variation, which has specific form for different types of tests [51]. We use unpaired t-test, where *s* is combination of standard deviations of two samples (detail about the combination can be found in [42]).

7.2.5.5. System Developments Validation and Analysis

Obviously WK, the number of wumpuses killed, is a key measurement. WK largely depends on WF, the number of wumpuses found by the carrier. WF in turn depends on the carrier's observability, the fighter's observability and WT, the total number of wumpuses which are generated during a game. Since the fighter has very limited observation radius (only 1), the large amount of WF is produced by the carrier's observability. Since the carrier's observability radius is the same for all three test conditions, WT becomes the prime element on deciding WF. In fact there are two way relations between WT and WF. On the one hand, the higher WT may lead to more WF. On the other hand, the higher WF is, the more wumpus locations will be told by

the carrier to the fighters, resulting in more wumpuses being killed by the fighters, and consequently more new wumpuses being generated, resulting in more WT. To present relations between WF and WT, and more important, to provide a fair test base for three teams, we validate WF and WT for each team and show they produce the same quantity of data. We use the ratio WF/WT as the base for validation. The data is shown in Table 7.4.

Table 7.4. Experiment Base Validations in Sample Runs.

Data / Team	ST	WF	WT	WF/WT(%)
Team PC	5000	358	1146	31.29
Team AT	4761	369	1167	31.59
Team AA	4300	304	992	31.02

Table 7.4 first gives us basic ideas about general performance of each team. Basically about one thousand wumpuses are generated for a 5000 steps game. Team PC is able to perform all games throughout 5000 steps; while the other two teams end the game earlier (especially for Team AA), meaning all fighters were dead before 5000 steps. The cause for fighters' early death is the communication risk. We will elaborate this point in next experiment.

Another observation to Table 7.4 is that the value of WF/WT is about the same for three teams. Table 7.5 shows P value for two pair teams, PC vs. AT and PC vs. AA, with respect to WF/WT. By conventional criteria, their differences are considered to be not statistically significant. This validates system developments such as agents' motion

rules, agents' observability rules, rules about wumpuses' random jumping, and rules about new wumpuses' generation. Based on these, we can perform the further evaluation.

Table 7.5. P Value with Respect to WF/WT.

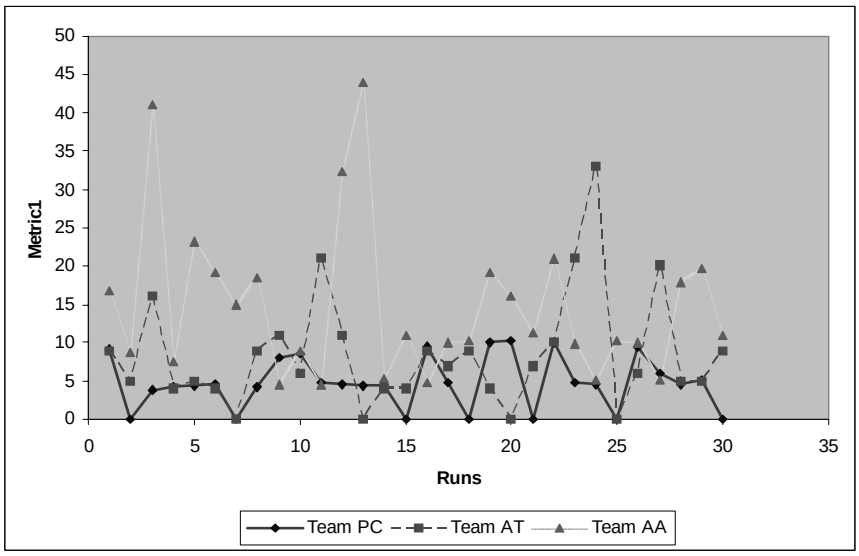
Team	P	Explanation
PC vs. AT	0.3454	Not statistic different
PC vs. AA	0.5700	Not statistic different

7.2.5.6. Effectiveness Evaluation and Analysis

This experiment explores how Proactive Communication reduces communication load and improves team performance in multi-agent teamwork. Fig. 7.4 shows the effectiveness evaluation and the P value of the two pair teams, with respect to Metrics1, 2, 3 and 4. We expect Team PC to have lower values for Metric1 and 2, and higher values for Metric3 and 4.

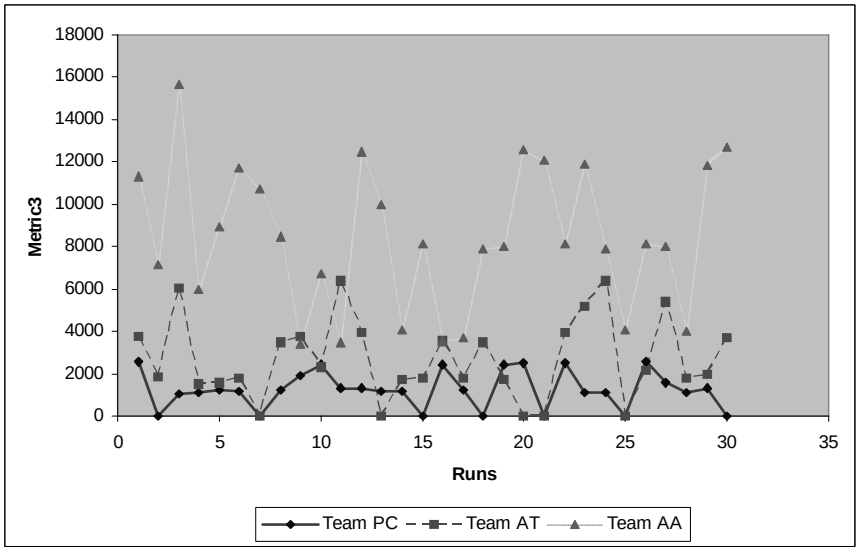
By studying Fig. 7.4[24], we find that, as our expectation, Team PC performs best , regarding Metric1, Metric2, and Metric4. The P values also show that the differences between the two pair teams are statistically significant.

[24] The lines between the sample points should not be there, as the points are not interpolating values. The lines are only used to distinguish the data for each team.



	Average		T-test		
Metric1	PC	4.88	P		Explanation
	AT	8.62	PC vs. AT	0.0298	Statistic different
	AA	15.16	PC vs. AA	<0.0001	Extremely statistic different

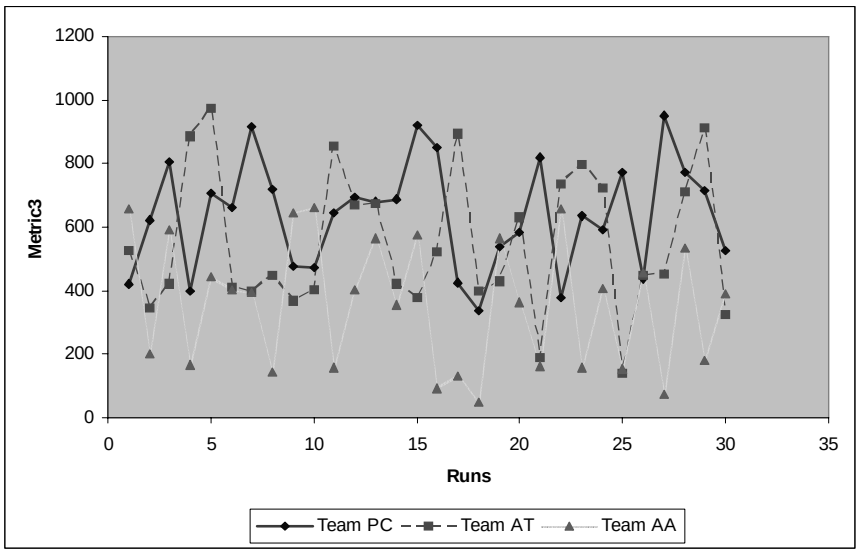
(a) Metric1.



	Average		T-test		
Metric2	PC	1252.09	P		Explanation
	AT	2691.09	PC vs. AT	0.0004	Extremely statistic different
	AA	8422.00	PC vs. AA	<0.0001	Extremely statistic different

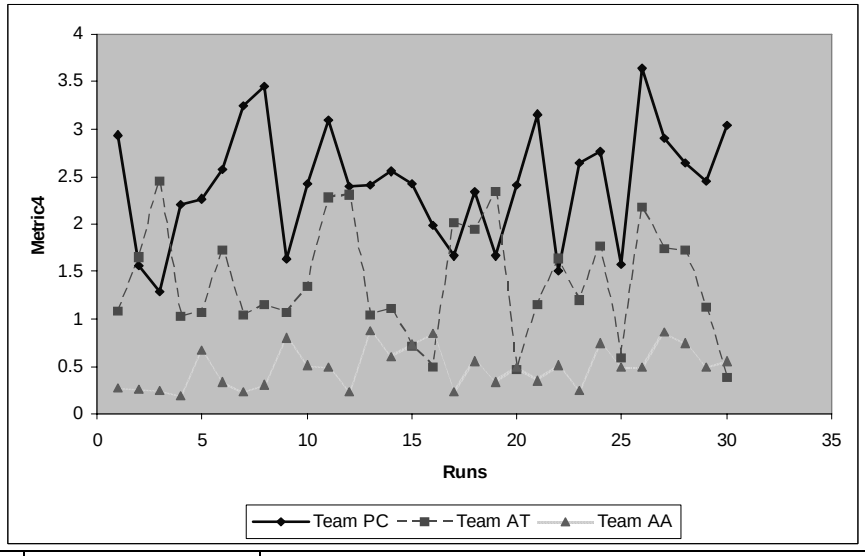
(b) Metric2.

Fig. 7.4. Effectiveness Evaluation with Respect to Metric1-Metric4.



	Average		T-test		
Metric3	PC	647.53	P		Explanation
	AT	545.97	PC vs. AT	0.0710	Not quite statistic different
	AA	344.73	PC vs. AA	<0.0001	Extremely statistic different

(c) Metric3.



	Average		T-test		
Metric4	PC	2.43	P		Explanation
	AT	1.43	PC vs. AT	<0.0001	Extremely statistic different
	AA	0.48	PC vs. AA	<0.0001	Extremely statistic different

(d) Metric4.

Fig. 7.4. Continued.

The results for Metric3 were not quite as good in that while Team PC has a very significantly better performance than Team AA, and its average metric is better than that of Team AT, the difference from AT is not quite statistically significant. The reason is that, by being always told about wumpuses locations, fighters of Team AT, are able to receive the most timely and the most recent items, allowing them to kill as many wumpuses as they can in a limited length of time, though they are exposed to greater risk and suffer some loss from this. To help understand these results, it is interested to take a close look to communication policies used by each team.

By Always Asking for information at the time when needs occur, fighters are able to receive the most timely and the most recent items, allowing them to kill as many wumpuses as they can in a limited length of time. Moreover, carriers can track the exact locations of fighters, which are the locations of the wumpuses' the fighters were last told. This ensures their choosing the fighter closest to the wumpus found every time. The disadvantage of Team AA is the possible high communication risk. Under the Always Ask condition, obtaining an information item by a fighter costs at least two messages (one ask and one reply). So Team AA would exchange the largest numbers of messages. When the fighter asks for a wumpus location, the chance of alerting the wumpus will be increased. This again increases the chance that the fighter is killed, and consequently, fewer wumpuses killed, and often ends the game ahead of time with all of the fighters being killed. Hence, the Proactive Communication approach is better because of the better management of risk.

The Always Tell condition can almost guarantee a high degree of effectiveness in conveying timely and the most recent information as Always Ask does. The carrier is able to (without time delay) provide the latest wumpus location to fighters and then the fighters can use the most recent location to chase wumpuses. However, this approach also has a high communication risk, resulting in similar disadvantages to those of Always Ask. Therefore, the more wumpus locations provided; the more wumpuses will be alerted. Consequently, the chance that the fighter will be killed is increased over that of proactive communication, though not as much so as with active ask.

Proactive Communication may not be able to deliver as timely as the other two conditions. Sometimes carriers must keep silence or fighters have to wait or use the old information, if the risk of communication exceeds its value. This keeps the team safe but brings two side effects: 1) fewer wumpus locations are told, compared to the number of wumpuses found; and 2) information exchange is delayed. Hence Team PC may not be able to kill as many wumpuses as the other two teams do. However, this could be compensated by minimizing the number of messages sent and the risk of fighters' death. In fact, the death of a fighter is a heavy loss to the team. It will lead to the fewer wumpuses been killed, or even the forcefully end of the game. Proactive Communication wins on communication amount mainly because it sends the information only when it is needed, in spite of changes of the information. So Proactive Communication results in the fewest messages exchanged.

Based on above analysis, it makes more sense to compare the average number of wumpus killed per message. In this term, which is Metric4, the performance of Team PC is statistically better than those of Team AT and Team AA. Hence, our algorithms for managing Proactive Communication have been effective.

7.3. Summary

In this section, we have first conducted in-depth empirical evaluations in the Multi-Agent Wumpus World, comparing the relative numbers of *O-Tell* and *O-Ask* for agent teams with and without observability. We have also given specific forms of risk, cost, timeliness and currency functions in the Multi-Agent Wumpus World. We presented two experiments that validate the system developments and explore the effectiveness of operating teamwork under Proactive Communication. The results of these experiments show that our approaches have improved the team performance.

CHAPTER VIII

CONCLUSIONS AND FUTURE WORK

8.1. Conclusions

My long-term research goal is to understand intelligence and to build human knowledge into software agents to support decision-making, and to improve the productivity and adaptability of autonomous systems in complex and dynamic environments. Toward this goal, I have researched Proactive Communication in agent teamwork

In Observation-Based Proactive Communication, we employed agents' observabilities as major means for decreasing the volume of communication in a dynamic and partially observable environment [141]. We formally defined what is observable and under which conditions. The exploration of observability also carefully clarifies the relationships among what an agent can see, what it actually sees, and what it believes from its seeing. This, however, is not enough to allow inference of belief about other agents and use of this belief to track their mental states. We then defined agents' beliefs about the observabilities of other agents. The amount of communication is reduced by agents' using observation of the environment and beliefs of teammates' observabilities to estimate the teammates' beliefs without generating unnecessary messages.

Decision-Theoretic Proactive Communication uses decision theory to enable agents to decide whether or not to engage in a communication act when less is known

about the domain and the results of their interaction with it, and communication may incur cost and risk. It allows agents to tell others proactively about a piece of information when producing it, or to ask actively for a piece of information when needing it [14, 141]. It formally includes the notion that the times at which an agent needs or produces a value for an item of information is random according to some (unknown) distributions. The idea of Dynamic Information Prediction is to develop techniques for estimating the distributions of information production or need and use these data to model utilities of each communication strategy available for agents on each situation of decision-making. The estimation serves proactive communication in two ways: first, agents can proactively tell up-to-date information to agents who need it; second, it helps on providing a more accurate and efficient way of communicating information than randomly selecting a receiver, or making the selection in a specific order, in that agents can dynamically issue communication at the right time to the right receivers without having to know all about the receivers.

The decision-theoretic approach provides agents with an optimal way to fulfill their information needs under uncertainties, caused by incomplete information about the teamwork, the environment, and the potential value, cost and risk of information delivery. We developed a set of communication policies for agents in different situations. Since the various policies involve using the information produced at different times or satisfying needs at different times, we carefully studied different points in time, which describes the range of possibilities encompassed by the different strategies adequately. We analyzed parameters that should be included in the utility

function and recognized effects in determining the form of the value, the cost and the risk which compose the utility function. The distinguishing feature of the decision-theoretic approach is that we focus on analyzing the information production and need of team members and use these data to capture the complex decision processes of information needer and provider. Moreover, this approach emphasizes decision interactions between the needer and provider, i.e. their decisions are interdependent, so they must consider the impact of their counterpart's decisions upon their own.

8.2. Future Work

There are two aspects work we plan to do in the future. First we will enhance the current model by extending its functionalities. Then this model will be applied into several real applications.

8.2.1. Extensions to This Research

8.2.1.1. Extending the Current Model to Multiple Needers and Providers Model

The current *selectPolicy* algorithm (see section 6.9) can well handle the one-to-one model, i.e., there is one provider and one needer for an information item I[25]. In this case, agents consider interactions with their counterpart agents and make decisions. To deal with many-to-many model, which includes multiple needers and providers of an information item I, we did a straightforward extension to the one-to-one model. We assume agents still focus on interactions with their counterparts and consider only the number of counterparts is extended.

[25] The system may still contain multiple agents. But for a piece of information, there is only one needer and one provider.

Extending the one-to-one model to the many-to-many model in a more complete way would require including complicated interactions among needers and providers. For a needer, it must monitor not only each provider, but also other needers. Thus, when making decision, the needer must estimate every provider's policy to the needer itself, every provider's policy to every other needer, and every other needer's policy to every provider. Similar processes can occur to the provider. This extension of our work could take advantage of policies and utilities devised in current model and focus on developing feasible decision rules to coordinate time orders of multiple productions and needs.

We would start with analyzing the case of multiple needers. When several needers want *I*, their needs for *I* are dynamically changed during teamwork. In many situations, every new value for *I* must be used and once it is used, it is unnecessary to use it again, such as location of an enemy target. Hence, except that the provider should pay attention to provide the unused *I* to needers who have needs, every needer also needs to watch when other needers' needs raised and whether they have actively asked for *I* or have been told proactively. Therefore, when making a decision, a needer needs to consider the provider's decision in relation to the needer, the provider's decision relative to every other needer, and every other needer's decision relative to the provider.

In the case of multiple providers, a provider has the similar concerns to those of the needers above. In the situation where a provider produces *I*, it is possible that another provider also produced *I* recently and has sent *I* to the needer, by either

proactive tell or reply to the needer's request. In this case, the provider may not provide *I* because the needer may not need *I* soon. Just as the needer can ask an arbitrary provider for *I*, an arbitrary provider can help the needer proactively. Thus, a provider needs to consider the needer's decision in relation to the provider, the needer's decision relative to every other provider, and every other provider's decision relative to the needer.

In the future, we still want to focus on decision interactions between the decision maker and its counterparts. In the many-to-many model, we can adjust the decision rule we will develop for the one-to-one model. We will add the counterpart agent as one parameter in the utility function. In the case of multiple providers, the needer will evaluate the utility function with respect to each provider, and then follow the strategy of the provider that yields the highest utility. However, when end states in the needers state diagram that call for additional reasoning are reached, the needer will re-evaluate the utility functions and possibly make a new decision, which may mean asking a different provider. However, the decision process will have to be extended because additional situations can occur, e.g., a needer might have received multiple proactive tells before its need arose. In the case of multiple needers, the provider will evaluate the utility function with respect to each needer, and then follow the strategy that has the maximum utility.

8.2.1.2. Using Plan Recognition to Generate Information Flow

Our present proactive communication algorithm analyzes preconditions and effects of actions and plans for which each agent is responsible in the teamwork. The

purpose of doing so is to determine potentially useful information flow among agents. Currently the information flow is generated offline. The problem of this approach is that the predicates of potential information needers or providers extracted by the offline algorithm may contain variables, and agents will bind actual values to these variables dynamically during the teamwork processes. For example, in the Multi-Agent Wumpus World, the offline algorithm only extracts that a fighter needs wumpus' location, but cannot identify which fighter among three fighters. Our solution to this problem is to use the decision-theoretic proactive communication to estimate agents that would be most likely bound to a plan at the time of information production or need.

Alternatively, we could make the recognition of information needers or providers more dynamic by doing plan recognition [69]. Agents can recognize the plans of other agents by observing actions of the other agents, and tracking the sequence of sub-goals on which they are working dynamically. Using this information together with the action an agent has most recently performed, the most likely information need or production of other agents could be dynamically estimated over a finite time horizon. Then we could send or ask other agents information only when they had just needed/produced the information, or if they are expected to need/produce the information in the near future.

8.2.2. Future Directions

8.2.2.1. Multi-Agent Learning

Multi-agent learning supports learning from interaction with open-ended, dynamic environments that include multiple, autonomous data and knowledge sources.

Examples of such domains include data-driven collaborative knowledge discovery; distributed information networks for selective information retrieval; distribution of software and real-time audio/video streams; and distributed parallel processing. Multi-agent learning methods would include design of algorithms for learning from heterogeneous data sources, distributed in time and space. As a consequence of the developments in technology that make it possible to accumulate large amount of data incrementally every day, in physically distributed, autonomous data repositories (e.g. bioinformatics), such algorithms seem to be the need of the hour. Analogous to the work done here, decision theory and empirical distribution function analysis might be usable for analyzing data pattern of various applications and reducing communication among data sources and sinks.

8.2.2.2. Distributed Information Networks for Selective Information Gathering

There are many types of applications for which time-constrained and predictable response is required, which is closely related to my research; the most familiar are electronic trading systems, games, defense systems, and multimedia applications. There, time-critical applications depend on careful system design and timely resource allocation to deliver the required performance. For example, in an online E-Commerce system, stores that sell the same types of products consist of a multi-agent system over the Internet. Each agent (store) in such a system wants to charge a price that beats the other stores, but at the same time maintain maximum profits for the store over a period of time. The interaction among the stores can be modeled as a stochastic game. The agent's price has to take into account the future

prices charged by other stores and potential loss resulting from that. Agents then need to monitor the prices charged by other stores continuously, learn about their price-setting pattern, and modify their stores' prices accordingly. Proactive Communication fits these requirements well and could support interaction analysis between two stores.

8.2.2.3. Virtual Humans for Training

Agents can be used to develop a foundation for efficacious training of complex performance. In Intelligent Team Training Systems, human team members are trained by putting them into a simulation, which allows them to perform and refine their team skills. The two types of agents which can be developed to assist team training are partner agents [113] and coaching agents [87, 132]. Coaching agents provide coaching feedback to trainees and their team based on the performance and the process of the team. Partner agents assist individual trainees by taking over the execution of some of the component tasks, allowing the trainee to concentrate on learning specific components, and assist team training by fulfilling the roles of some team members. Both types of agents require communication to achieve the desired team interactions. The agents should track the activities of the human trainee, reason about possible conflicts or constraints, establish certain parameters for performing joint actions, and provide or request any information needed by the human trainees to perform their tasks. However, this complex team cooperation behavior may involve much unnecessary message exchange because of introducing the human team members. Proactive Communication could provide desired interactions for humans and agents.

REFERENCES

[1] J. Anderson, C. Boyle, A. Corbett, M. Lewis, Cognitive modeling and intelligent tutoring, Artificial Intelligence 42 (1990) 7-49.

[2] T. Balch, R. Arkin, Communication in reactive multi-agent robotic systems, Autonomous Robots 1 (1994) 27-53.

[3] R. Bartels, J. Beatty, B. Barsky, Hermite and Cubic Spline Interpolation, Morgan Kaufmann, New York, 1998.

[4] H. Becker, Notes on the concept of commitment, American Journal of Sociology 66 (1960) 32-40.

[5] J. Bell, S. Huang, Seeing is believing, in: Proceedings of the Conference of Common Sense (CS-98), London, United Kingdom, 1998, pp. 321-327.

[6] K. Biggers, Automatic generation of communication and teamwork within multi-agent teams. M.S. Thesis, Department of Computer Science, Texas A&M University, College Station, Texas, 2001.

[7] A. Binas, T. Ioerger, Multi-agent belief reasoning in a first-order logic back-chainer, Technical Report TSSTI-TR-10-04, Training System Science and Technology Initiative, Texas A&M University, College Station, Texas, 2004.

[8] A. Bond, Commitment: some DAI insights from symbolic interactionist society, in: Proceedings of the 9th Workshop on DAI (DAI-89), Bellevue, Washington, 1989, pp. 239-261.

[9] M. Bratman, Intention, Plans, and Practical Reasons, Harvard University Press, Cambridge, Massachusetts, 1987.

[10] M. Bratman, D. Israel, M. Pollack, Plans and resources bounded practical reasoning, Computational Intelligence 4 (1988) 349-355.

[11] H. Bui, D. Kieronska, S. Venkatesh, Optimal communication among team members, in: Lecture Notes in Artificial Intelligence, vol. 1342, 1997, pp. 116-126.

[12] J. Cannon-Bowers, E. Salas, S. Converse, Shared mental models in expert team decision-making, in: J. Castellan (Ed.), In Individual and Group Decision-Making: Current Issues, Lawrence Erlbaum Associates, Mahwah, New Jersey, 1993, pp. 221-246.

[13] J. Cannon-Bowers, E. Salas, A framework for developing team performance measures in training, in: M. Brannick, E. Salas, C. Prince (Eds.), Team Performance Assessment and Measurement: Theory, Research and Applications, Lawrence Erlbaum Associates, Mahwah, New Jersey, 1997, pp. 45-62.

[14] S. Cao, R. Volz, T. Ioerger, Y. Zhang, J. Yen, Role-based and agent-oriented teamwork modeling, in: Proceedings of the International Conference on Artificial Intelligence (ICAI-02), Las Vegas, Nevada, 2002, pp. 1190-1196.

[15] G. Casella, R. Berger, Statistical Inference, Duxbury Press, Belmont, California, 1990.

[16] C. Castelfranchi, Guarantees for autonomy in cognitive agent architecture, in: W. Jennings (Ed.), Intelligence Agents, Springer-Verlag, New York, 1996, pp. 56-70.

[17] E. Charniak, Bayesian networks without tears, AI Magazine 12 (1991) 50-63.

[18] M. Cohen, J. Freeman, B. Thompson, Critical thinking skills in tactical decision making: a model and a training strategy, in: J. Cannon-Bowers, E. Salas (Eds.), Decision Making Under Stress: Implications for Training and Simulation, American Psychological Association, Washington, D.C., 1997, pp. 155-189.

[19] P. Cohen, H. Levesque, Intention is choice with commitment, Artificial Intelligence 42 (1990) 213-261.

[20] P. Cohen, H. Levesque, Teamwork, Nous (Special Issue on Cognitive Science and Artificial Intelligence) 25 (1991) 487-512.

[21] J. Cremer, J. Kearney, Y. Papelis, R. Romano, The software architecture for scenario control in the Iowa driving simulator, in: Proceedings of the 4th Computer Generated Forces and Behavioral Representation Conference (CGFBRC-94), Orlando, Florida, 1994, pp. 213-218.

[22] K. Crowston, E. Kammerer, Coordination and collective mind in software requirements development, Journal of IBM Systems 37 (1998) 227-245.

[23] B. Curtis, H. Krasner, N. Iscoe, Field study of the software design process for large systems, Comm. ACM 31 (11) (1988) 1268-1286.

[24] R. D'Agostino, M. Stephens, Goodness-of-Fit Techniques, Marcel Dekker, New York, 1986.

[25] E. Davis, Solutions to a paradox of perception with limited acuity, in: Proceedings of the 1st International Conference on Knowledge Representation and Reasoning (KRR-89), Toronto, Canada, 1989, pp. 79-82.

[26] J. de Kleer, An assumption-based truth maintenance system, Artificial Intelligence 28 (1986) 127-162.

[27] D. Dennett, The Intentional Stance, MIT Press, Cambridge, Massachusetts, 1987.

[28] J. Doyle, A truth maintenance system, Artificial Intelligence 12 (1979) 231-272.

[29] D. Ellsberg, Risk, ambiguity and the savage axioms. Quarterly Journal of Economics 75 (1961) 643-669.

[30] J. Espinosa, K. Carley, R. Kraut, F. Lerch, S. Fussell (Eds.), The Effect of Shared Mental Models and Knowledge Distribution on Asynchronous Team Coordination and Performance: Empirical Evidence from Management Teams, Carnegie Mellon University Press, Pittsburgh, Pennsylvania, 2001.

[31] R. Fagin, J. Halpern, Reasoning about knowledge and probability, Journal of the ACM 39(1994) 328-396.

[32] X. Fan, J. Yen, M. Miller, R. Volz, The semantics of MALLET - an agent teamwork encoding language, in: Proceedings of the 4th International Joint Conference on Autonomous Agents and Multi Agent Systems Workshop on Declarative Agent Languages and Technologies (DALT-04), New York, 2004, pp. 69-91.

[33] X. Fan, S. Sun, J. Yen, M. McNeese, Extending recognition-primed decision model for human-agent collaboration, in: Proceedings of the 4th International

Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-05), Utrecht, The Netherlands, 2005, pp. 334-341.

[34] M. Fenster, S. Kraus, J. Rosenschein, Coordination without communication: experimental validation of focal point techniques, in: Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, California, 1995, pp. 102-108.

[35] R. Fikes, N. Nilsson, STRIPS: a new approach to the application of theorem proving to problem solving, Artificial Intelligence 2 (1971) 189-208.

[36] R. Fikes, A commitment-based framework for describing informal cooperative work, Cognitive Science 6 (1982) 331-347.

[37] T. Finin, Y. Labrou, J. Mayfield, KQML as a communication language, in: J. Bradshaw (Ed.), Software Agents, AAAI, Menlo Park, California, 1997, pp. 291-316.

[38] M. Genesereth, S. Ketchpel, Software agents, Comm. ACM, 37 (7) (1984) 48-53.

[39] P. Gmytrasiewicz, E. Durfee, D. Wehe, A decision-theoretic approach to coordinating multi-agent interactions, in: Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91), Sydney, Australia, 1991, pp. 62-68.

[40] P. Gmytrasiewicz, E. Surfee, Rational coordination in multi-agent environments, Autonomous Agents and Multi-Agent Systems 3 (2000) 319-350.

[41] C. Goldman, S. Ziberstein, Optimizing information exchange in cooperative multi-agent systems, in: Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-03), Melbourne, Australia, 2003, pp. 137-144.

[42] W. Gosset, The probable error of a mean, Biometrika 6 (1908) 1-25.

[43] B. Grosz, C. Sidner, Plans for discourse, in: P. Cohen, J. Morgan, M. Pollack (Eds.), Intentions in Communication, MIT Press, Cambridge, Massachusetts, 1990, pp. 417-443.

[44] B. Grosz, S. Kraus, Collaborative plans for complex group actions, Artificial Intelligence 86 (1996) 269-357.

[45] B. Grosz, Collaborative systems, AI Magazine 17 (1996) 67-85.

[46] B. Grosz, S. Kraus, Planning and acting together, AI Magazine 20 (1999) 23-34.

[47] J. Halpern, Y. Moses, A guide to completeness and complexity for modal logics of knowledge and belief, Artificial Intelligence 54 (1992) 319-379.

[48] L. He, T. Ioerger, Combining bundle search with buyer coalition formation in electronic markets: a distributed approach through explicit negotiation, in: Proceedings of the 6th International Conference on Electronic Commerce (ICEC-04), Delft, The Netherlands, 2004, pp. 95-104.

[49] J. Hintikka, Knowledge and Belief, Cornell University Press, New York, 1962.

[50] C. Hoare, An axiomatic basis for computer programming, Comm. ACM 12 (10) (1969) 576-580.

[51] P. Hoel, Introduction to Mathematical Statistics, John Wiley & Sons, New York, 1984

[52] J. Hu, M. Wellman, Online learning about other agents in a dynamic multi-agent system, in: Proceedings of the 2nd International Conference on Autonomous Agents (Agents-98), Minneapolis, Minnesota, 1998, pp. 239-246.

[53] M. Huber, E. Durfee, Deciding when to commit to action during observation-based coordination, in: Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, California, 1995, pp. 163-170.

[54] M. Huber, E. Durfee, On acting together: without communication, in: Proceedings of the AAAI Spring Symposium on Representing Mental States and Mechanisms, Stanford, California, 1995, pp. 60-71.

[55] M. Huhns, D. Bridgeland, Multi-agent truth maintenance, IEEE Transactions on Systems, Man and Cybernetics 21 (1991) 1437-1445.

[56] U. Hustadt, Do we need the closed world assumption in knowledge representation, in: Proceedings of the 1st Workshop of Knowledge Representation Meets Databases (KRDB-94), Saarbrucken, Germany, 1994, pp. 24-26.

[57] T. Ioerger, R. Volz, J. Yen, Modeling cooperative, reactive behaviors on the battlefield using intelligent agents, in: Proceedings of the 9th Conference on Computer Generated Forces (CGF-00), Orlando, Florida, 2000, pp. 13-23.

[58] T. Ioerger, JARE Menu, Available at http://jare.sourceforge.net, 2001.

[59] T. Ioerger, Literature review: modeling teamwork as part of human behavior representation, Technical Report TSSTI-TR-10-03, Training System Science and Technology Initiative, Texas A&M University, College Station, Texas, 2003.

[60] T. Ioerger, Reasoning about beliefs, observability, and information exchange in teamwork, in: Proceedings of the 17th International Conference of the Florida Artificial Intelligence Research Society (FLAIRS-04), Miami Beach, Florida, 2004, pp. 23-31.

[61] H. Isozaki, H. Katsuno, A semantic characterization of an algorithm for estimating others' beliefs from observation, in: Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96), Menlo Park, California, 1996, pp. 543-549.

[62] H. Isozaki, H. Katsuno, Observability-based nested belief computation for multi-agent systems and its formalization, in: Lecture Notes in Intelligent Agent, vol. 1757, 2000, pp. 27-41.

[63] N. Jennings, Commitments and conventions: the foundation of coordination in multi-agent systems, Knowledge Engineering Review 8 (1993) 223-250.

[64] N. Jennings, L. Varga, R. Aarnts, J. Fuchs, P. Skarek, Transforming standalone expert systems into a community of cooperating agents, Engineering Applications of AI 6 (1993) 317-331.

[65] N. Jennings, Controlling cooperative problem-solving in industrial multi-agent systems using joint intentions, Artificial Intelligence 75 (1995) 195-240.

[66] D. Jensen, M. Atighetchi, V. Lesser, Learning quantitative knowledge for multi-agent coordination, in: Proceedings of the National Conference on Artificial Intelligence (AAAI-99), Orlando, Florida, 1999, pp. 24-31.

[67] G. Kaminka, M. Tambe, Robust agent teams via socially-attentive monitoring, Journal of Artificial Intelligence Research 12 (2000) 105-147.

[68] G. Kaminka, D. Pynadath, M. Tambe, Monitoring deployed agent teams, in: Proceedings of the International Conference on Autonomous Agents (Agents-01), Quebec, Canada, 2001, pp. 308-315.

[69] G. Kaminka, M. Tambe, Monitoring teams by overhearing: a multi-agent plan-recognition approach, Journal of Artificial Intelligence Research 17(2002) 83-135.

[70] D. Kinny, M. Ljungberg, A. Rao, G. Tidhar, E. Werner, E. Sonenberg, Planned team activity, in: Proceedings of the 4th European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW-92), Viterbo, Italy, 1992, pp. 226-256.

[71] G. Klein, Recognition-primed decisions, Advances in Man-Machine Systems Research 5 (1989) 47–92.

[72] G. Klein, J. Orasanu, R. Calderwood, C. Zsambok, Decision Making in Action: Models and Methods, Ablex Publishing Corporation, Norwood, New Jersey, 1993.

[73] R. Klimoski, S. Mohamed, Team mental model: construct or methaphor, Journal of Management 20 (1994) 403-437.

[74] K. Konologe, A Deduction Model of Belief, Morgan Kaufmann, New York, 1986.

[75] S. Kraus, J. Rosenschein, The role of representation in interaction: discovering focal points among alternative solutions, ACM SIGOIS Bulletin 13 (1992) 12-25.

[76] S. Kraus, Negotiation and cooperation in multi-agent environments, Artificial Intelligence 94 (1997) 79-98.

[77] Y. Kuniyoshi, S. Rougeaux, M. Ishii, Cooperation by observation: the framework and basic task patterns, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-94), San Diego, California, 1994, pp. 767-774.

[78] Y. Lashkari, M. Metral, P. Maes, Collaborative interface in agents, in: Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94), Seattle, Washington, 1994, pp. 444-449.

[79] N. Lesh, C. Rich, C. Sidner, Using plan recognition in human-computer collaboration, in: Proceedings of the 7th International Conference on User Modeling (UM-99), Banff, Canada, 1999, pp. 23-32.

[80] H. Levesque, A logic of implicit and explicit belief, in: Proceedings of the National Conference on Artificial Intelligence (AAAI-84), Austin, Texas, 1984, pp. 198-202.

[81] H. Levesque, P. Cohen, J. Nunes, On Acting Together, in: Proceedings of the National Conference on Artificial Intelligence (AAAI-90), Boston, Massachusetts, 1990, pp. 94-99.

[82] B. Linder, W. Hoek, J. Meyer, Seeing is believing - and so hearing and jumping, in: Lecture Notes in Artificial Intelligence, vol. 992, 1995, pp. 402-413.

[83] J. Mathieu, G. Goodwin, T. Heffner, E. Salas, J. Cannon-Bowers, The influence of shared mental models on team process and performance, Journal of Applied Psychology 85 (2000) 273-283.

[84] M. Mazer, Reasoning about knowledge to understand distributed AI systems, IEEE Transactions on Systems, Man and Cybernetics 21 (1991) 1333-1346.

[85] D. McAllester, A three valued truth maintenance system, AI Memo 473, MIT Artificial Intelligence Laboratory, MIT, Cambridge, Massachusetts, 1978.

[86] B. Mellers, A. Schwartz, A. Cooke, Judgment and decision making, Annual Review of Psychology 49 (1998) 447–478.

[87] M. Miller, J. Yin, T. Ioerger, J. Yen, R. Volz, Training teams with collaborative agents, in: Proceedings of the 5th International Conference on Intelligent Tutoring Systems (ITS-00), Quebec, Canada, 2000, pp. 63-72.

[88] G. Monahan, A survey of partially observable Markov decision processes: theory, models, and algorithms, Management Science 28 (1982) 1-16.

[89] R. Moore, Reasoning about Knowledge and Action, MIT Press, Cambridge, Massachusetts, 1980.

[90] B. Morgan, Measurement of team behaviors in a Navy environment, Naval Training Systems Center, Orlando, Florida, 1986.

[91] D. Musto, K. Konolige, Reasoning about perception, in: Proceedings of the AAAI Spring Symposium on Reasoning about Mental States (RMS-93), Stanford, California, 1993, pp. 90-95.

[92] S. Parsons, M. Wooldridge, Game theory and decision theory in multi-agent systems, in: Proceedings of the 1st Autonomous Agents and Multi-Agent Systems (AAMAS-02), Bologna, Italy, 2002, pp. 243-254.

[93] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, New York, 1988.

[94] J. Pollack, Knowledge and Justification, Princeton University Press, Princeton, New Jersey, 1974.

[95] D. Pynadath, M. Tambe, Multi-agent teamwork: analyzing the optimality and complexity of key theories and models, in: Proceedings of the 1st Autonomous Agents and Multi-Agent Systems Conference (AAMAS-02), Bologna, Italy, 2002, pp. 873-880.

[96] A. Rao, M. Georgeff, BDI agents: from theory to practice, in: Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, California, 1995, pp. 312-319.

[97] C. Rich, C. Sidner, COLLAGEN: when agents collaborate with people, in: Proceedings of the 1st International Conference on Autonomous Agents (Agents-97), Marina del Rey, California, 1997, pp. 284-291.

[98] S. Ross, Stochastic Processes, Wiley, Alameda, California, 1996.

[99] W. Rouse, N. Morris, On looking into the black box: prospects and limits in the search for mental models, Psychological Bulletin 100 (1986) 349-363.

[100] W. Rouse, J. Cannon-Bowers, E. Salas, The role of mental model in team performance in complex systems, IEEE Transactions on Systems, Man, and Cybernetics 22 (1992) 1296-1308.

[101] R. Rozich, A practical method for proactive information exchange within multi-agent teams, M.S. Thesis, Department of Computer Science, Texas A&M University, College Station, Texas, 2003.

[102] S. Russell, P. Norvig, Artificial Intelligence A Modern Approach, Prentice Hall, Upper Saddle River, New Jersey, 1995.

[103] C. Sakama, K. Inoue, Prioritized logic programming and application to commonsense reasoning, Artificial Intelligence 123 (2000) 185–222.

[104] T. Sandholm, Distributed rational decision making, in: G. Weiss (Ed.), Multi-Agent Systems, MIT Press, Cambridge, Massachusetts, 1999, pp. 201-258.

[105] T. Schelling, The Strategy of Conflict, Oxford University Press, Oxford, United Kingdom, 1963.

[106] J. Schmidhuber, J. Zhao, Multi-agent learning with the success-story algorithm, in: G. Weiss and J. Gerhard (Eds.), Distributed Artificial Intelligence Meets Machine Learning: Learning in Multi-Agent Environment, Springer, New York, 1996, pp. 82-93.

[107] J. Searle, Internationality: An Essay in the Philosophy of Mind, Cambridge University Press, Cambridge, United Kingdom, 1983.

[108] J. Searle, Collective intentions and actions, in: P. Cohen, J. Morgan, M. Pollack (Eds.), Intentions in Communication, MIT Press, Cambridge, Massachusetts, 1990, pp. 401-415.

[109] S. Sen, E. Durfee, The role of commitment in cooperative negotiation, International Journal on Intelligent and Cooperative Information Systems 3 (1994) 67-81.

[110] D. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, Chapman & Hall / CRC, Oxford, United Kingdom, 2004.

[111] Y. Shoham, M. Tennenholtz, On the synthesis of useful social laws for artificial agents societies (preliminary report), in: Proceedings of the National Conference on Artificial Intelligence (AAAI-92), San Jose, California, 1992, 276-281.

[112] Y. Shoham, Agent-oriented programming, Artificial Intelligence 60 (1993) 51-92.

[113] J. Sims, Use of partner agents in training systems for complex tasks, M.S. Thesis, Department of Computer Science, Texas A&M University, College Station, Texas, 2002.

[114] I. Smith, P. Cohen, Toward a semantics for an agent communications language based on speech-acts, in: Proceedings of the National Conference of Artificial Intelligence (AAAI-96), Menlo Park, California, 1996, pp. 24-31.

[115] R. Smith, The contract net protocol: high-level communication and control in a distributed problem solver, IEEE Transactions on Computers 29 (1980) 1104-1113.

[116] P. Stone, M. Veloso, Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork, Artificial Intelligence 110 (1999) 241-273.

[117] T. Sugawara, V. Lesser, Learning to improve coordinated actions in cooperative distributed problem-solving environments, Machine Learning 33 (1988) 129-153.

[118] K. Sycara, C. Lewis, Forming shared mental models, in: Proceedings of the 13th Annual Meeting of the Cognitive Science Society (CSS-91), Chicago, Illinois, 1991, pp. 400-405.

[119] K. Sycara, M. Paolucci, M. Velsen, J. Giampapa, The RETSINA MAS infrastructure, Autonomous Agents and Multi-Agent Systems 7 (2003) 29-48.

[120] M. Tambe, P. Rosenbloom, RESC: an approach for real-time dynamic agent tracking, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95), Quebec, Canada, 1995, pp. 103-111.

[121] M. Tambe, Towards flexible teamwork, Journal of Artificial Intelligence Research 7 (1997) 83-124.

[122] A. Val, Y. Shoham, Qualitative reasoning about perception and belief, in: Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97), Nagoya, Japan, 1997, pp. 508-513.

[123] M. Viroli, A. Omicini, An observation approach to the semantics of agent communication languages, Applied Artificial Intelligence 16 (2002) 775-793.

[124] D. Walz, J. Elam, B. Curtis, Inside a software design team: knowledge acquisition, sharing, and integration, Comm. ACM 36 (10) (1993) 63-77.

[125] K. Weick, The collapse of sensemaking in organizations: the Mann Gulch disaster, Administrative Science Quarterly 38 (1993) 628-652.

[126] E. Weisstein, Random Walk: 2-dimensional, Mathworld - A Wolfram Web Resource, Available at http://mathworld.wolfram.com, 2005.

[127] M. Wooldridge, N. Jennings, Intelligent agents: theory and practice, Knowledge Engineering Review 10 (1995) 115-152.

[128] M. Wooldridge, A. Lomuscio, Reasoning about visibility, perception and knowledge, in: Lecture Notes in Artificial Intelligence, vol. 1757, 2000, pp. 1-12.

[129] M. Wooldridge, A. Lomuscio, A logic of visibility, perception and knowledge: completeness and correspondence results, in: Proceedings of the 3rd International Conference on Pure and Applied Practical Reasoning (PAPR-00), London, United Kingdom, 2000, pp. 23-31.

[130] M. Wooldridge, A. Lomuscio, Multi-agent VSK logic, in: Proceedings of the 17th European Workshop on Logics in AI (ELAI-00), London, United Kingdom, 2000, pp. 300-312.

[131] M. Wooldridge, A. Lomuscio, A computationally grounded logic of visibility, perception and knowledge, Logic Journal of the ZGPL 9 (2001) 257-272.

[132] D. Xu, M. Miller, R. Volz, T. Ioerger, Collaborative agents for C2 teamwork simulation, in: Proceedings of the International Conference on Artificial Intelligence (ICAI-03), Las Vegas, Nevada, 2003, pp. 723-729.

[133] P. Xuan, V. Lesser, S. Zilberstein, Communication decisions in multi-agent cooperation: model and experiments, in: Proceedings of the 5th International Conference on Autonomous Agents (Agents-01), Quebec, Canada, 2001, pp. 616-623.

[134] B. Yang, Performance Measurement in ADEPT/JxA, M.S. Thesis, Department of Computer Science, Texas A&M University, College Station, Texas, 1998.

[135] J. Yen, J. Yin, T. Ioerger, M. Miller, D. Xu, R. Volz, CAST: collaborative agents for simulating teamwork, in: Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), Seattle, Washington, 2001, pp. 1135-1144.

[136] J. Yin, M. Miller, T. Ioerger, J. Yen, R. Volz, A knowledge-based approach for designing intelligent team training systems, in: Proceedings of the 4th International Conference on Autonomous Agents (Agents-00), Barcelona, Spain, 2000, pp. 427-434.

[137] J. Yin, A multi-agent framework for simulating proactive teamwork, Ph.D. Dissertation, Department of Computer Science, Texas A&M University, College Station, Texas, 2001.

[138] W. Zachary, J. Ryder, J. Hicinbothom, Cognitive task analysis and modeling of decision making in complex environments, in: J. Cannon-Bowers, E. Salas

(Eds.), Decision Making Under Stress: Implications for Training and Simulation, American Psychological Association, Washington, D.C., 1997, pp. 315-344.

[139] Y. Zhang, K. Biggers, L. He, S. Reddy, D. Sepulvado, J. Yen, T. Ioerger, A distributed intelligent agent architecture for simulating aggregate-level behavior and interactions on the battlefield, in: Proceedings of the 5th World Multi-Conference on Systemics, Cybernetics, and Informatics (SCI-01), Orlando, Florida, 2001, pp. 58-63.

[140] Y. Zhang, L. He, K. Biggers, J. Yen, T. Ioerger, Simulating teamwork and information-flow in tactical operations centers using multi-agent systems, in: Proceedings of the 10th Conference on Computer Generated Forces (CGF-01), Norfolk, Virginia, 2001, pp. 529-539.

[141] Y. Zhang, R. Volz, T. Ioerger, S. Cao, J. Yen, Proactive information exchange during team cooperation, in: Proceedings of the International Conference on Artificial Intelligence (ICAI-02), Las Vegas, Nevada, 2002, pp. 341-346.

[142] C. Zsambok, L. Beach, G. Klein, A literature review of analytical and naturalistic decision making, Technical Report, Naval Command, Control and Ocean Surveillance Center, San Diego, California, 1992.

[143] C. Zsambok, G. Klein, Naturalistic Decision Making, Lawrence Erlbaum Associates, Mahwah, New Jersey, 1997.

[144] C. Zsambok, Naturalistic decision making: where are we now, in: C. Zsambok, G. Klein (Eds.), Naturalistic Decision Making, Lawrence Erlbaum Associates, Mahwah, New Jersey, 1997, pp. 3-16.

APPENDIX A

CALCULATING PROBABILITIES

In this appendix we calculate the probabilities, Pr($T_{b,N} \leq T^0_{a,P}$), Pr($T^0_{a,P} < T_{b,N}$), and Pr($T^0_{a,P} < T_{b,N} < T^1_{a,P}$), used by the provider in Chapter VI for calculating the currency.

- Pr($T_{b,N} \leq T^0_{a,P}$)

First we need to decide the base from which the provider can estimate $T_{b,N}$. There are two bases, depending on the needer's decision for the need immediately before $T_{b,N}$, i.e. the need raised at $T^{-1}_{b,N}$. If the needer chose *Silence* at $T^{-1}_{b,N}$, the base for estimating $T_{b,N}$ is $T^{-1}_{b,N}$; if the needer *ActiveAsk* or *Wait* at $T^{-1}_{b,N}$, this need can be satisfied by $I_P(T^{ls}_{a,P})$, so the base is $T^{ls}_{a,P}$. In this latter case, the calculation is straightforward, as the provider knows both values.

$$\Pr(T_{b,N} \leq T^0_{a,P})$$

$$= \Pr(T_{b,N} - (T^{ls}_{a,P}) \leq T^0_{a,P} - (T^{ls}_{a,P}))$$

$$= \mathrm{CDF}_{b,N}(T^0_{a,P} - T^{ls}_{a,P}).$$

In the former case, the provider knows $T^{ls}_{a,P}$ but not $T^{-1}_{b,N}$. It is easy to conclude that $T^{-1}_{b,N} < T^{ls}_{a,P} < T_{b,N}$. To seek a reasonable and computationally feasible solution for deciding the base for $T_{b,N}$, we used the expected value of where $T^{ls}_{a,P}$ would lie in the interval ($T^{-1}_{b,N}$, $T_{b,N}$), which under reasonable assumptions would be half way in

between them. Then, we use $T_{a,P}^{ls}-\tau_n/2$ as an estimate for the base for $T_{b,N}$ (τ_n denotes the average length of time between needs of *I* and is defined in Section 6.7.1.1.1).

$$\Pr(T_{b,N} \le T_{a,P}^{0})$$

$$=\Pr(T_{b,N}-(T_{a,P}^{ls}-\tau_n/2) \le T_{a,P}^{0}-(T_{a,P}^{ls}-\tau_n/2))$$

$$=\mathrm{CDF}_{b,N}(T_{a,P}^{0}-T_{a,P}^{ls}+\tau_n/2).$$

- $\Pr(T_{a,P}^{1} \le T_{b,N})$

$$\Pr(T_{a,P}^{1} \le T_{b,N})$$

$$=\sum_{\tau=T_{a,P}^{1}}^{\infty} \Pr(T_{a,P}^{1} \le T_{b,N} \mid T_{b,N}=\tau)\times\Pr(T_{b,N}=\tau)$$

$$=\sum_{\tau=T_{a,P}^{1}}^{\infty} 1\times\Pr(T_{b,N}=\tau)$$

$$=\sum_{\tau=T_{a,P}^{1}}^{\infty} \mathrm{PMF}_{b,N}(\tau-T_{a,P}^{ls}+\tau_n/2)$$

$$=1-\mathrm{CDF}_{b,N}(T_{a,P}^{1}-T_{a,P}^{ls}+\tau_n/2)$$

$$=\sum_{\tau 1=T_{a,P}^{0}+1}^{\infty} \Pr(T_{a,P}^{1}=\tau 1)\times(1-\mathrm{CDF}_{b,N}(\tau 1-T_{a,P}^{ls}+\tau_n/2))$$

$$=\sum_{\tau 1=T_{a,P}^{0}+1}^{\infty} \mathrm{PMF}_{a,P}(\tau 1-T_{a,P}^{0})\times(1-\mathrm{CDF}_{b,N}(\tau 1-T_{a,P}^{ls}+\tau_n/2)).$$

Though above equation involves infinite ∞, since the distributions are based on a finite number of measured values, therefore only a finite number of terms needed to be added.

- $\Pr(T_{a,P}^{0}<T_{b,N}<T_{a,P}^{1})$

$$\Pr(T_{a,P}^{0}<T_{b,N}<T_{a,P}^{1})$$

$$=1-\Pr(T_{b,N} \le T^{0}_{a,P})-\Pr(T^{1}_{a,P} \le T_{b,N})$$

$$=1-\mathrm{CDF}_{b,\mathrm{N}}(T^{0}_{a,P}-T^{ls}_{a,P}+\tau_{\mathrm{n}}/2)-$$

$$\sum_{\tau 1=T^{0}_{a,P}+1}^{\infty} \mathrm{PMF}_{a,\mathrm{P}}(\tau 1-T^{0}_{a,P})\times(1-\mathrm{CDF}_{b,\mathrm{N}}(\tau 1-T^{ls}_{a,P}+\tau_{\mathrm{n}}/2)).$$

- $\Pr(T^{0}_{a,P}<T_{b,N})$

$$\Pr(T^{0}_{a,P}<T_{b,N}) = 1-\Pr(T_{b,N} \le T^{0}_{a,P})$$

$$=1-\mathrm{CDF}_{b,\mathrm{N}}(T^{0}_{a,P}-T^{ls}_{a,P}+\tau_{\mathrm{n}}/2).$$

APPENDIX B

MULTI-AGENT WUMPUS WORLD MALLET PLAN

```
(plan wumpusgame(?ca ?f1 ?f2 ?f3)
  (process
    (par
      (while (cond (goal killwumpus))
        (seq
          (do ?ca (findWumpus ?ca))
          (do ?ca (retract (newKnow ?wumpusId ?x ?y)))
        ) ;end seq
      )   ;end while
      (while (cond (goal killwumpus))
        (do ?f1 (killWumpus ?f1))
      ) ;end while
      (while (cond (goal killwumpus))
        (do ?f2 (killWumpus ?f2))
      ) ;end while
      (while (cond (goal killwumpus))
        (do ?f3 (killWumpus ?f3))
      ) ;end while
    ) ;end par
  )   ;end process
```

```
)     ;end plan
(plan findWumpus(?ca)
  (effects (newKnow ?wumpusId ?x ?y))
  (process
    (while (cond (not (newKnow ?wumpusId ?x ?y)))
      (do ?ca (moveAndFindStep ?ca))
    )   ;end while
  )     ;end process
)       ;end plan
(plan moveAndFindStep(?who)
  (process
     (seq
       (do ?who (observe ?who))
       (do ?who (move))
       (do ?who (nextstep ?who))
     )  ;end seq
  )     ;end process
)       ;end plan
(plan observe(?who)
  (process
    (seq
      (do ?who (see ?who))  ;see is an operator; when see is executed,
```

```
                                ;all bound (canSeeNow ?who ?item ?x ?y)
                                ;will be asserted to KB of ?who
      (do ?who (generateNewKnow ?who))
      (do ?who (updateMostRecentSeen ?who))
      (do ?who (retract (canSeeNow ?who ?anyitem ?anyx ?anyy)))
    ) ;end seq
  )   ;end process
)     ;end plan
(plan generateNewKnow(?who)
  (process
    (foreach (cond (canSeeNow ?who ?item ?x ?y))
      (if (cond (not (mostRecentSeen ?item ?x ?y)))
        (seq
          (do ?who (assert (mostRecentSeen ?item ?x ?y)))
          (if (cond (wumpus ?item))
            (seq
              (do ?who (assert (newKnow ?item ?x ?y)))
              (do ?who (assert (unsafe ?x ?y)))
            ) ;end seq
          )   ;end inner if
        )     ;end seq
      ) ;end if
```

```
    )   ;end foreach
  );end process
)  ;end plan
(plan updateMostRecentSeen(?who)
  (process
    (foreach (cond (mostRecentSeen ?anyLastItem ?anyLastx ?anyLasty))
      (if (cond (not (canSeeNow ?who ?anyLastItem ?anyLastx ?anyLasty)))
        (seq
          (do ?who (retract (mostRecentSeen ?anyLastItem ?anyLastx ?anyLasty)))
          (do ?who (retract (unsafe ?anyLastx ?anyLasty)))
        );end seq
      )  ;end if
    )    ;end foreach
  );end process
)  ;end plan
(plan killWumpus(?fi)
  (process
    (par
      (while (cond (not (newKnow ?wumpusId ?x ?y)))
        (seq
          (do ?fi (noop))
          (do ?fi (nextstep ?fi))
```

```
        );end seq
      ) ;end while
      (do ?fi (startKill ?fi))
    ) ;end par
  )   ;end process
)     ;end plan
(plan startKill(?fi)
  (pre-cond (newKnow ?wumpusId ?x ?y))
  (process
    (seq
      (do ?fi (moveToWumpus ?fi ?wumpusId ?x ?y))
      (do ?fi (shootwumpus ?wumpusId ?x ?y))
      (do ?fi (retract (newKnow ?wumpusId ?x ?y)))
      (do ?fi (nextstep ?fi))
    ) ;end seq
  )   ;end process
)     ;end plan
(plan moveToWumpus(?fi ?wumpusId ?x ?y)
  (process
    (while (cond (notAdjacent ?fi ?wumpusId ?x ?y))
      (do ?fi (moveToStep ?fi ?x ?y))
    );end while
```

```
  )  ;end process
)    ;end plan
(plan moveToStep(?fi ?x ?y)
  (process
    (seq
      (do ?fi (observe ?fi))
      (do ?fi (moveto ?x ?y))
      (do ?fi (nextstep ?fi))
    )        ;end seq
  )          ;end process
)            ;end plan
```

APPENDIX C

CALCULATING RISK IN MULTI-AGENT WUMPUS WORLD

The risk function has been defined in Section 7.2.3.1 as

$$R = k \times (5000 - t) \times Pr_h \times Pr_f$$

where k=0.01 is the number of wumpuses a fighter can kill per unit time, t is the number of steps passed, Pr_h is the probability that the wumpus overhears the message sent by the carrier, and Pr_f=0.1 is the probability that the wumpus can win against the fighter.

The last value to compute is Pr_h,

$$Pr_h = Pr_r \times Pr_a,$$

where Pr_r denotes the probability that an agent is within the wumpus' hearing range r_w, and Pr_a denotes the probability that the wumpus pays attention to the message.

Pr_r must be calculated for all three of carrier's policies: *ProactiveTell*, *Reply* and *WaitUntilNext*.

There are two cases to consider for Pr_a.

a The wumpus is "unalerted" so may or may not pay attention to the message:

$$Pr_a = 0.1.$$

When it is important to distinguish the alerted and unalerted conditions in the same expression, the "unalerted" case will be denoted by Pr_{an}; Pr_{an} will still have the value 0.1.

b The wumpus is "alerted" by the request send by the fighter so will pay attention to the coming reply. Calculating Pr_a in this case is more complex. However, it needs only be done for two of the three carrier's policies, *Reply* and *WaitUntilNext*, as in the *ProactiveTell* case the wumpus is never alerted.

Next we calculate Pr_h for three of carrier's policies: *ProactiveTell*, *Reply* and *WaitUntilNext*.

C.1. The Case That the Carrier Proactively Tells a Message

Since $r_c<r_w$, the carrier is within the wumpus' hearing range. Therefore $Pr_r=1$ and Pr_h equals to Pr_a. Thus:

$$Pr_h=Pr_a=0.1.$$

C.2. The Case That the Carrier Replies a Request sent from a Fighter

When the carrier receives a request, it must select one of the two policies identified in the general analysis (see Section 6.2.2), i.e., *Reply* with the location of the last wumpus sensed or *WaitUntilNext* time it finds a wumpus. Our analysis follows these two policies.

C.2.1. Policy – Reply Last Location Found

As part of its decision process, the carrier needs to estimate the possible effect of a wumpus detecting that it has been found by using the risk function. There are multiple sub-cases to consider, depending upon whether or not the carrier can still sense the wumpus. The situation is complicated in that one must consider both the probability that the wumpus can hear the carrier and the probability that the wumpus

heard the request from the fighter. Fortunately, which case applies can be decided by the carrier because the fighter sends its location with the request and the carrier thus knows the locations of both the fighter and the wumpus, as well as the wumpus' hearing range.

To distinguish the situations of the wumpus hearing the carrier or the fighter, we add one subscript to Pr_h, Pr_r, and Pr_a to indicate the agent to which it refers; i.e., we use Pr_{hc} and Pr_{hf} respectively. We use Pr_{rc} and Pr_{rf} to denote the probability that the carrier or the fighter is within the hearing range of the wumpus, and use Pr_{ac} and Pr_{af} to denote the probability that the wumpus pays attention to the message sent by the carrier or the fighter. Then, we have:

$$Pr_{hc}=Pr_{rc}\times Pr_{ac}, \qquad \text{(C-1)}$$

$$Pr_{hf}=Pr_{rf}\times Pr_{af}.$$

C.2.1.1. Sub-Case 1 – Carrier Can Still Sense Wumpus

If the carrier is still able to sense the last wumpus found at the time of replying, the carrier is within the wumpus' hearing range. Therefore, $Pr_{rc}=1$, and we have

$$Pr_{hc}=Pr_{ac}.$$

Pr_{ac} is determined by Pr_{hf}, the probability that the wumpus heard the fighter's request. Pr_{ac} is give by the following relation:

$$Pr_{ac}=Pr_{an}\times(1-Pr_{hf})+1\times Pr_{hf}.$$

We have seen that $Pr_{hf}=Pr_{rf}\times Pr_{af}$. Since we are considering the fighter who initializes the communication, the wumpus was "unalerted" at that moment and Pr_{af} was assumed to be 0.1. Since the fighter attached its location to the request, the carrier

knows the distance between the fighter and the wumpus. Therefore Pr_{rf} is decidable by the carrier. There are thus two further sub-cases to consider:

a The fighter was within r_w the wumpus hearing radius. In this case, $Pr_{rf}=1$, and therefore $Pr_{hf}=Pr_{af} = 0.1$.

b The fighter was not within r_w. In this case, the wumpus could not hear the fighter and therefore $Pr_{rf}=0$. $Pr_{hf}=Pr_{rf}\times Pr_{af} =0 \times 0.1=0$.

Then the carrier will go back and use Pr_{hf} and Pr_{af} to decide Pr_{hc}. If $Pr_{hf}=0.1$ (sub-case a), then $Pr_{hc} = Pr_{af}\times(1–Pr_{hf})+1\times Pr_{hf} = 0.1\times(1–0.1)+1\times 0.1=0.19$, and if $Pr_{hf}=0$ (sub-case b), then $Pr_{hc}= 0.1\times (1–0)+1\times 0=0.1$.

C.2.1.2. Sub-Case 2 – Carrier Cannot Sense Wumpus

If the carrier cannot sense wumpus at the time of evaluation, two further sub-cases may arise:

a The wumpus has stayed at the location last sensed by the carrier.

b The wumpus has jumped since the carrier last saw it.

Pr_{hc} can then be calculated as:

$$(Pr_{hc} \mid WNJ)\times Pr(WNJ|)+(Pr_{hc} \mid \neg WNJ)\times(1–Pr(WNJ)),$$

where WNJ means “wumpus not jump” in the interval between when the carrier last saw it and the present time, conditioned upon the fact that it has not jumped between when it first appeared in the location it was observed and the time last seen. In the following, we will first consider Pr_{hc} for the two cases and then calculate Pr(WNJ).

If the wumpus has not jumped, the carrier is able to determine whether or not it is within the wumpus’ hearing range. Pr_{hc} is then as calculated in *Sub-case 1* above.

If the wumpus has jumped, we assume it no longer pays specific attention to a message emanating from a carrier within its hearing radius; it simply goes back to the "unalerted" status and hears the message with probability Pr_{ac}=0.1; from Eq. (C-1), the problem then reduces to determining Pr_{rc}. The difficulty of determining Pr_{rc} is caused by the fact that the carrier cannot decide whether or not it is within the wumpus' hearing range. Thus, we must estimate the probability that such is the case. If there is no other information available about the location, we assume that the wumpus is randomly placed in the area that is not observable at this moment. The area of the carrier's observation rhombus is $2r_c^2+2r_c+1$. So the area of possible wumpus location is $O-(2r_c^2+2r_c+1)$. Since the carrier cannot sense the wumpus, the area in which the carrier cannot sense the wumpus but can be heard by the wumpus is approximated by $(2r_w^2+2r_w+1)-(2r_c^2+2r_c+1)$ (recall that $r_w>r_c$). Therefore the probability that the carrier is within the hearing range of this wumpus is:

$$Pr_{rc}=\frac{(2r_w^2-2r_w-1)-(2r_c^2-2r_c-1)}{O-(2r_c^2-2r_c-1)}.$$

Consequently we have:

$$Pr_{hc}=Pr_{rc}\times Pr_{ac}=\frac{(2r_w^2-2r_w-1)-(2r_c^2-2r_c-1)}{O-(2r_c^2-2r_c-1)}\times 0.1.$$

What is left, then, is to calculate Pr(WNJ). Let D_0 be the time duration that the wumpus has stayed at this location before it was first sensed by the carrier, D_s be the length of time between when the carrier first saw this wumpus and when it last saw this wumpus, and D_n be the length of time between when the carrier last saw the wumpus

and the current time. The carrier knows D_s and D_n but not D_0. Further, let H denote the time difference between when the carrier last saw the wumpus and the time at which the wumpus jumps. When a wumpus is created, a random variable, X, is given a value between 1 and 40 under a uniform distribution. Then, recalling that our notation of WNJ implied a conditional probability, the probability that the wumpus did not jump during D_n is equal to:

$$\Pr(\text{WNJ}) = \Pr(D_n<H|\ D_0+D_s < X).$$

H must conform to the following constraints:

$$H\in[1,\ 40-D_0-D_s],$$

$$D_0\in[1,\ 40-D_s],$$

$$D_0+D_s+D_n<40.$$

We calculate the probability Pr(WNJ), with D_0=d and H=h as the random variables:

$$\Pr(D_n<H \mid D_0+D_s<X)$$

$$=\sum_{d=1}^{40-D_s-D_n-1} \Pr(D_0=d) \times \Pr(D_n<H \mid D_0+D_s<X \wedge D_0=d)$$

$$=\sum_{d=1}^{40-D_s-D_n-1} \Pr(D_0=d) \times \Pr(D_n+D_0+D_s<H+D_0+D_s \mid D_0+D_s<X \wedge D_0=d)$$

But, $H+D_0+D_s = X$, the random duration chosen for the next wumpus jump. Thus,

$$\Pr(D_n<H \mid D_0+D_s<X)$$

$$=\sum_{d=1}^{40-D_s-D_n-1} \frac{1}{40-D_s} \times \Pr(D_n+D_0+D_s<X \mid D_0+D_s<X \wedge D_0=d)$$

$$=\sum_{d=1}^{40-D_s-D_n-1} \frac{1}{40-D_s} \times \frac{\Pr(D_n+D_0+D_s<X \mid D_0=d)}{\Pr(D_0+D_s<X \mid D_0=d)}$$

$$= \sum_{d=1}^{40-D_s-D_n-1} \frac{1}{40-D_s} \times \frac{40-(D_n+d+D_s)}{40} \times \frac{40}{40-(d+D_s)}$$

$$= \sum_{d=1}^{40-D_s-D_n-1} \frac{1}{40-D_s} \times \frac{40-(D_n+d+D_s)}{40-(d+D_s)}.$$

C.2.2. Policy – Carrier Decides to Wait until Next Finding after Receiving Request

In the case that the carrier will wait until the next time it finds a wumpus, though it does not know the wumpus' location at this moment, the carrier can be sure that it will be inside of the wumpus' hearing radius r_w at the time of finding because $r_c < r_w$. Therefore the probability that the carrier is within the hearing range of the wumpus, Pr_{rc}, is one:

$$Pr_{rc} = 1.$$

Then since $Pr_{hc} = Pr_{rc} \times Pr_{ac}$, $Pr_{hc} = Pr_{ac}$, the probability that the wumpus pays attention to the reply. Pr_{ac} is again determined by Pr_{hf}, the probability that the wumpus heard the request sent by the fighter, by the form defined in *Sub-case 1*:

$$Pr_{ac} = Pr_{an} \times (1 - Pr_{hf}) + 1 \times Pr_{hf}.$$

As noted above $Pr_{hf} = Pr_{rf} \times P_{af}$; hence the carrier needs to calculate 1) Pr_{rf}, the probability that the fighter was inside of the wumpus' hearing radius r_w at the time of sending the request, and 2) Pr_{af}, the probability that the wumpus pays attention to the request that is within its hearing range. For Pr_{rf}, the carrier will not know if the fighter is inside of r_w, because the carrier does not know the wumpus' location at this moment. The carrier can use a method similar to the one it uses to calculate Pr_{rc} in *Sub-case 2* in Section C.2.1. Then $Pr_{rf} = ((2r_w^2 - 2r_w - 1) - (2r_f^2 - 2r_f - 1))/(O - (2r_f^2 - 2r_f - 1))$. $Pr_{af} = 0.1$ because the

wumpus was in "unalerted" status at the time of the fighter sending the request. Therefore, in this case,

$$\begin{aligned}
&Pr_{hc}\\
&=Pr_{rc}\times Pr_{ac}\\
&=1\times Pr_{ac}\\
&=Pr_{af}\times(1-Pr_{hf})+1\times Pr_{hf}\\
&=0.1\times\\
&\left(1-0.1\times\frac{(2r_w^2-2r_w-1)-(2r_f^2-2r_f-1)}{O-(2r_f^2-2r_f-1)}\right)+\\
&1\times\left(0.1\times\frac{(2r_w^2-2r_w-1)-(2r_f^2-2r_f-1)}{O-(2r_f^2-2r_f-1)}\right)\\
&=0.1+0.09\times\frac{(2r_w^2-2r_w-1)-(2r_f^2-2r_f-1)}{O-(2r_f^2-2r_f-1)}.
\end{aligned}$$

In summation, risk will take place for three of carrier's policies: *ProactiveTell*, *Reply* and *WaitUntilNext*. The following table shows Pr_h, the probability that the wumpus overhears the message sent by the carrier, for these policies. Once Pr_h is computed, risk can be easily computed.

Policy \ Probability			Pr_h
ProactiveTell			0.1
Reply	Carrier still see wumpus	Fighter was within r_w	0.19
		Fighter was not within r_w	0.1
	Carrier not see wumpus	Fighter was within r_w	$0.19\times E\{Pr(D_n<J)\}+\frac{(2r_w^2-2r_w-1)-(2r_c^2-2r_c-1)}{O-(2r_c^2-2r_c-1)}\times 0.1\times(1-E\{Pr(D_n<J)\})$
		Fighter was not within r_w	$0.1\times E\{Pr(D_n<J)\}+\frac{(2r_w^2-2r_w-1)-(2r_c^2-2r_c-1)}{O-(2r_c^2-2r_c-1)}\times 0.1\times(1-E\{Pr(D_n<J)\})$
WaitUntilNext			$0.1+0.09\times\frac{(2r_w^2-2r_w-1)-(2r_f^2-2r_f-1)}{O-(2r_f^2-2r_f-1)}$

APPENDIX D

CALCULATING PROBABILITY OF CORRECTNESS IN MULTI-AGENT WUMPUS WORLD

The probability of currency, P, is shown in Section 7.2.3.4 to be

$$\Pr(t_u - t_p < J)$$

where t_u is the time at which the fighter arrives at wumpus' location, t_p is the time at which the carrier finds this wumpus, and J denotes the difference between the time at which the value for *I* was produced and the time at which the wumpus jumps. Since some parameters of P may be unknown, we compute the expected probability $E\{\Pr(t_u - t_p < J)\}$.

In Section 7.2.3.4, we examine the *GeneralCase* and deduce Eq. (7-5) for calculating $E\{\Pr(t_u - t_p < J)\}$ in this case:

$$E\{\Pr(t_u - t_p < J)\}$$

$$= \sum_{d=1}^{40} \frac{1}{40} \times \sum_{j=1}^{40-d} \frac{1}{40-d} \times$$

$$\sum_{X_c=\max(1, X_{cl}-\sqrt{\sigma})}^{\min(20, X_{cl}+\sqrt{\sigma})} \sum_{Y_c=\max(1, Y_{cl}-(\sqrt{\sigma}-|X_c-X_{cl}|))}^{\min(20, Y_{cl}+(\sqrt{\sigma}-|X_c-X_{cl}|))} \frac{1}{2\sigma + 2\sqrt{\sigma} + 1} \times$$

$$\sum_{X_w=\max(1, X_c-r_c)}^{\min(20, X_c+r_c)} \sum_{Y_w=\max(1, Y_c-(r_c-|X_w-X_c|))}^{\min(20, Y_c+(r_c-|X_w-X_c|))} \frac{1}{2r_c^2 + 2r_c + 1} \times f_m(X_w, Y_w, X_f, Y_f, j)$$

where $\sigma = \max(t_p - t_{cl}, 0)$ is the length of time the carrier moves from the time t_{cl} to t_p; t_{cl} is the most recent time at which the decision maker knows the location of the carrier; $L_{cl}=(X_{cl}, Y_{cl})$ is the carrier's location last known by the decision maker; $L_c=(X_c, Y_c)$ is

the carrier's location at time t_p; $L_w=(X_w, Y_w)$ is the wumpus's location at time t_p; $L_f=(X_f , Y_f)$ is a fighter's location at t_n, the time at which the fighter needs a wumpus' location; and r_c is the observable rhombus vertex distance from the carrier.

Based on the *GeneralCase* and Eq. (7-5), this appendix calculates $E\{Pr(t_u - t_p<J)\}$ for each situation/policy combination.

D.1. Situation PA: The Carrier Finds a Wumpus' Location – *ProactiveTell*

In this case, $t_p=T^0_{a,P}$, which is known to the carrier, and $t_n=T_{b,N}$, which is unknown. However, the value for *I* provided at time $T^0_{a,P}$, $I_P(T^0_{a,P})$, may not always be used for the need that arose at $T_{b,N}$ because the wumpus might jump before the fighter arrives at the wumpus' location. P is specified as:

$$Pr(t_u - T^0_{a,P}<J),$$

where t_u, the time at which the fighter arrives at the wumpus' location, will be specified later. This probability can be evaluated conditionally on two sub-cases:

$$Pr(t_u - T^0_{a,P}<J \mid T_{b,N} \le T^0_{a,P}) \times Pr(T_{b,N} \le T^0_{a,P})$$

$$+ Pr(t_u - T^0_{a,P}<J \mid T^0_{a,P} < T_{b,N}) \times Pr(T^0_{a,P} < T_{b,N}).$$

$Pr(T_{b,N} \le T^0_{a,P})$ and $Pr(T^0_{a,P} < T_{b,N})$ are calculated in Appendix A. Below we consider $Pr(t_u - T^0_{a,P}<J)$ for the two sub-cases. Since some variables of this probability are unknown, we calculate $E\{Pr(t_u - t_p<J)\}$.

D.1.1. Sub-case PT–1 $T_{b,N} \leq T^0_{a,P}$

The carrier is the decision maker and $t=t_{cl}=t_p=T^0_{a,P}$. Thus, $\sigma = \max(0, t_p - t_{cl}) = 0$, and the two summations involving σ in Eq. (7-5) reduce to a single point. Also, since the carrier knows L_w, the summations over possible wumpus locations are irrelevant, leaving only the portions involving D_0 (the length of time that the wumpus was in its current location before being sensed by the carrier), J and the fighter location. There are then two further sub-cases to consider: 1) the carrier senses the fighter at this moment, $T^0_{a,P}$, and 2) the carrier does not sense the fighter at $T^0_{a,P}$.

In the first sub-case, L_f is known. Since there is a pending need (because in this case $T_{b,N} \leq T^0_{a,P}$) and the location of the wumpus is being proactively told, the fighter will immediately use the information and start moving toward the wumpus. Hence, $t_u = T^0_{a,P} + D_k$[26]. L_f and L_w are known and hence D_k is known. In addition, since D_k is known, this puts a lower bound on the values of j[27] that are possible, and this in turn, places an upper bound on the range of value of d that is possible. Hence, Eq. (7-5) reduces to:

$$E\{Pr(t_u - t_p < J)\}$$

$$= Pr(T^0_{a,P} + D_k - T^0_{a,P} < J)$$

$$= Pr(D_k < J)$$

[26] Recall that D_k is the distance the fighter must travel to reach the wumpus, and since the fighter moves one step per unit of time, D_k is also the time it take the fighter to reach the wumpus.
[27] Recall from Chapter VII that j and d are random variables representing J and D_0, respectively.

$$= \sum_{d=1}^{40-D_k-1} \frac{1}{40} \times \sum_{j=D_k+1}^{40-d} \frac{1}{40-d} \qquad \text{(D-1)}$$

We call this case *SimpleCase*, because $E\{Pr(t_u - t_p < J)\}$ simply needs one input D_k. Having probabilities of d and j, $E\{Pr(t_u - t_p < J)\}$ can be computed and then all other parameters in expression (7-3) are irrelevant. Therefore for any other case that D_k is known, it can be classified to the *SimpleCase* and the currency can be calculated with Eq. (D-1).

If the carrier does not sense the fighter, L_f is unknown. However, it is still the case that the fighter will use the information as soon as it receives it. Thus, $t_u = T^0_{a,P} + D_k$. The fighter must have moved to the location of the last wumpus that the fighter killed. The last wumpus the fighter killed could be either a wumpus whose location was sent to the fighter by the carrier (which is not necessarily the last one told by the carrier), or a wumpus the fighter found itself. Since the carrier is much more likely to find a wumpus than the fighter, we ignore the latter case, and assume that, at t_n, the fighter killed a wumpus whose location was sent by the carrier. However it would be hard for the carrier to know, among those wumpuses' locations which have been sent to the fighter, which is the last wumpus the fighter killed. It could be the last one the carrier sent or some other one before the last one. Since we assumed that $T_{b,N}$ is the first need that arose after $T^{ls}_{a,P}$ (see Section 6.7.1), we assume the wumpus last killed is the one immediately before the last one the carrier told the fighter about at $T^{ls}_{a,P}$. Hence L_f is approximated by the last wumpus location which the carrier sent it just before

$T^{ls}_{a,P}$. Denote this as L_{wl-1}. So $L_f = L_{wl-1}$. By having estimation for L_f, D_k is known. This case is the *SimpleCase* and E(Pr($t_u - t_p < J$)} can be calculated with Eq. (D-1).

D.1.2. Sub-case PT–2 $T^0_{a,P} < T_{b,N}$

In this case, t_u depends upon the decision the needer will make at $T_{b,N}$ and the provider's response decision at $T_{b,N}$. We consider combinations of cases.

$$\Pr(t_u - T^0_{a,P} < J \mid T^0_{a,P} < T_{b,N} \wedge E_n) \times \Pr(E_n),$$

where E_n, n=1,..,4, denote the following events:

E_1: needer decides to *Wait* at $T_{b,N}$;

E_2: needer decides to keep *Silence* at $T_{b,N}$;

E_3: needer decides to *ActiveAsk* at $T_{b,N}$ $\wedge$ provider decides to *Reply* at $T_{b,N}$;

E_4: needer decides to *ActiveAsk* at $T_{b,N}$ $\wedge$ provider decides to *WaitUntilNext* at $T_{b,N}$.

Based on the analysis in Section 6.7.1.1.2, the needer won't use $I_P(T^0_{a,P})$ for events E_1 and E_4. Hence we do not consider these cases. Specially in this domain the needer will not use $I_P(T^0_{a,P})$ given E_3 either. This is because we are considering *Proactivetell* at $T^0_{a,P}$, so the only condition under which the provider will decide to reply at $T_{b,N}$ is $T_{b,N} = T^1_{a,P}$. Therefore the needer won't use $I_P(T^0_{a,P})$ for E_3 and hence we also do not need to consider it.

Then the case left is E_2. Here we first consider $E(Pr(t_u - t_p < J)\}$ for E_2 and then compute $Pr(E_2)$.

D.1.2.1. Calculating Pr($t_u - T^0_{a,P} < J \mid T^0_{a,P} < T_{b,N}$) $\wedge$ E2)

- *PT–2 E_2: needer decides to keep Silence at $T_{b,N}$*

In this case, $t_u = T_{b,N} + D_k$. Also $t = t_p = t_{cl} = T^0_{a,P}$ which is known, and $t_n = T_{b,N}$ which is unknown. Thus, $\sigma = t_p - t_{cl} = 0$, and the summations in Eq. (7-5) dealing with σ and carrier movement reduce to a single term. Also, since the wumpus location is known, the summations in Eq. (7-5) that deal with wumpus location reduce to a single term. Since $t_n > t_p$, the fighter is chasing a wumpus up until time $T_{b,N}$. Either it is chasing a wumpus whose location was sent to it by the carrier, or it chasing a wumpus it found itself. Since the carrier is much more likely to find a wumpus than the fighter, we ignore the latter case, and assume that the fighter is chasing a wumpus whose location was sent to it by the carrier. With this assumption, what is needed, then, is some way to estimate the unknown time $T_{b,N}$.

From the information supplied by the fighter with active asks (and occasionally with the forced deadlock-breaking protocol), the carrier can determine the average time, τ_n, taken by the fighter to reach and kill a wumpus, measured from the time at which it received the wumpus location (which is not necessarily the time at which it

started chasing the wumpus). Thus, the carrier can estimate $T_{b,N}$ to be $T_{a,P}^{ls} + \tau_n/2$[28] (see Section 6.7.1.1.1). In the current sub-case, though, we are considering $T_{b,N} > T_{a,P}^{0}$, and there is no guarantee that $T_{a,P}^{ls} + \tau_n/2 > T_{a,P}^{0}$. Thus, we will use

$$T_{b,N} = \max(T_{a,P}^{0}+1,\ T_{a,P}^{ls} + \tau_n/2).$$

Since $T_{b,N} > T_{a,P}^{0}$, $T_{b,N} - T_{a,P}^{0}$ would be used to reduce the range of the summations over j and d, and the computation for $Pr(t_u - t_p < J)$ reduces to:

$$Pr(t_u - t_p < J)$$

$$= \sum_{d=1}^{40-(t_n - t_p + D_k + 1)} \frac{1}{40} \times \sum_{j=t_n - t_p + D_k + 1}^{40-d} \frac{1}{40-d} \qquad \text{(D-2)}$$

We call this case *ReducedSimpleCase* because the difference between this case and the *SimpleCase* is that this case uses $t_n - t_p$ to further reduce the ranges of summation. So for any later case if it knows D_k and t_p, and $t_n > t_p$, we will classify it to *ReducedSimpleCase*.

D.1.2.1.1. Calculating Pr(E_2)

- Pr(E_2)*:Pr(needer decides to keep Silence at* $T_{b,N}$*)*

Pr(needer decides to keep *Silence* at $T_{b,N}$)

=Pr(U(e, NA, *Silence*, $T_{b,N}$, t_p, {m}) > Max(U(e, NA, *ActiveAsk*, $T_{b,N}$, t_p, {m}), U(e, NA, *Wait*, $T_{b,N}$, t_p, {m}))),

where t_p will be replaced, for a given policy, by the value called for in that policy.

[28] It would, of course, be theoretically possible to use the distribution obtained from EDF and add another level of summation to the expression for $Pr(t_u - t_p < J)$.

The solution involves evaluating the utility function under each of the possible policies the needer might make at $T_{b,N}$. The most difficult part is to determine the currency. Determinations of currency involve a set of parameters. As before, for an unknown $T_{b,N}$, we use an estimate:

$$T_{b,N} = \max(T^{0}_{a,P}+1,\ T^{ls}_{a,P} + \tau_n/2).$$

Having this estimate, next we will specify all parameters which are need for Eq. (7-5) to calculate the currency. We consider three needer's possible policies separately.

Needer δ = Silence

The evaluation of *Silence* for the needer under the condition that a need has arisen is given in Section D.2.4.6, which requires these inputs for calculating the currency: D_k, D_s, D_n, t_n and t_p. Since we are evaluating the carrier's estimation for the fighter's use of *Silence* at $T_{b,N}$, the carrier needs to use its own knowledge to fill in these inputs. However the carrier's knowledge may be quite different from that of the fighter because of their different observabilities and motions. Moreover for the present case $T_{b,N}$ is in the future so the carrier is estimating the fighter's future decisions, while in Section D.2.4.6, the needer considers using a previous value. This makes the carrier be unable to know some of the inputs for calculating the currency in Section D.2.4.6. For example, in Section D.2.4.6, D_k is known to the fighter because the fighter knows itself's location L_f and the wumpus's location L_w. However for the present case the carrier may has no way to know L_w if this location will be provided in the future. So

the carrier cannot use the form of Section D.2.4.6 to calculate the currency. Instead, it should consider t_p and then decide which form of currency it can use.

Let's consider t_p first. We estimate $T^1_{a,P} = T^0_{a,P} + \tau_P$, where τ_P is the average length of time for producing a new value for *I*. Since we have an estimate for $T_{b,N}$ ($T_{b,N} = \max(T^0_{a,P}+1,\ T^{ls}_{a,P} + \tau_k/2)$), then we are able to decide the order of $T^0_{a,P}$, $T^1_{a,P}$ and $T_{b,N}$. We consider two cases: 1) $T^0_{a,P} < T_{b,N} < T^1_{a,P}$ and 2) $T^1_{a,P} \le T_{b,N}$.

In the case of $T^0_{a,P} < T_{b,N} < T^1_{a,P}$, $t_p = T^0_{a,P}$ so the carrier knows L_w. As before L_f is approximated by the last wumpus location which the carrier sent. So the carrier knows L_f. Therefore D_k is known. This case is exactly the same with Sub-case PT-2 E_2 (*ReducedSimpleCase*), where the fighter also will use $I_P(T^0_{a,P})$. $E\{Pr(t_u - t_p < J)\}$ will be estimated with the same form of Eq. (D-2).

In the case of $T^1_{a,P} \le T_{b,N}$, t_p is the most recent production time before $T_{b,N}$. We assume t_p equals the most recent production time before t_n (see Section 6.7.1.1.1.1 for rationale). Then we use Eq. (6-1) for t_p. Thus $t_p = T^0_{a,P} + Z \times \tau_p$ where $Z = \left\lfloor \frac{T_{b,N} - T^0_{a,P}}{\tau_p} \right\rfloor$) which returns 0 or an positive integer, meaning the number of productions during $T_{b,N}$ and $T^1_{a,P}$. Therefore

$$\sigma = \max(0, t_p - t_{cl}) = t_p - T^0_{a,P} = T^0_{a,P} + Z \times \tau_p - T^0_{a,P} = Z \times \tau_p.$$

The carrier knows L_{cl} its current location at $T^0_{a,P}$. It also knows L_f which is assumed to be the last wumpus location which the carrier sent to the fighter. Since $T_{b,N} > T^0_{a,P}$, $T_{b,N} - T^0_{a,P}$ would be used to reduce the range of the summations over j and d. Then $E\{Pr(t_u - t_p < J)\}$ can be approximated as the follows:

$$E\{Pr(t_u - t_p < J)\}$$

$$= \sum_{d=1}^{40-(t_n - t_p + 1)} \frac{1}{40} \times \sum_{j=t_n - t_p + 1}^{40-d} \frac{1}{40-d} \times$$

$$\sum_{X_c=\max(1, X_{cl}-\sqrt{\sigma})}^{\min(20, X_{cl}+\sqrt{\sigma})} \sum_{Y_c=\max(1, Y_{cl}-(\sqrt{\sigma}-|X_c - X_{cl}|))}^{\min(20, Y_{cl}+(\sqrt{\sigma}-|X_c - X_{cl}|))} \frac{1}{2\sigma + 2\sqrt{\sigma} + 1} \times$$

$$\sum_{X_w=\max(1, X_c - r_c)}^{\min(20, X_c + r_c)} \sum_{Y_w=\max(1, Y_c-(r_c-|X_w - X_c|))}^{\min(20, Y_c+(r_c-|X_w - X_c|))} \frac{1}{2r_c^2 + 2r_c + 1} \times f_m(X_w, Y_w, X_f, Y_f, j) \quad \text{(D-3)}.$$

We call this case *ReducedGeneralCase* because besides requiring σ, L_{cl} and L_f as what the *GeneralCase* does, this case also uses $t_n - t_p$ to reduce summations. For any later case which has the same requirement, we classify it to *ReducedGeneralCase* and the currency can be calculated with Eq. (D-3).

Needer δ = Wait

The evaluation of *Wait* for the needer is given in Section D.2.4.7, which requires these inputs for calculating the currency: σ, L_f and L_{cl}. Since there is a need, we assume t_p is the next production time after $T^0_{b,N}$. Thus we can use Eq. (6-2) to estimate t_p. Thus $t_p = T^0_{a,P} + (Z+1) \times \tau_p$. Also since $T^0_{a,P}$ is the most recent time at which the carrier know is own location (in fact $T^0_{a,P}$ is the current time), so $t_{cl} = T^0_{a,P}$, then:

$$\sigma = \max(0, t_p - t_{cl}) = t_p - T^0_{a,P} = (Z+1)\times\tau_p.$$

Having estimations for σ, and knowing L_{cl}, its own location at $T^0_{a,P}$, which is the current time, and L_f, the last wumpus location which the carrier sent to the fighter, the carrier is able to calculate $E\{Pr(t_u - t_p < J)\}$ with Eq. (7-5) of the *GeneralCase*.

Needer δ = *ActiveAsk*

In this case, t_p depends on the carrier's responding decision at $T_{b,N}$. So the carrier needs to estimate whether it will *Reply* or *WaitUntilNext* at $T_{b,N}$. The carrier needs to calculate Pr(provider decides to *Reply* at $T_{b,N}$) and Pr(provider decides to *WaitUntilNext* at $T_{b,N}$). It is enough to compute one of them because they are complement to each other. We choose Pr(carrier decides to *Reply* at $T_{b,N}$).

Pr(provider decides to *Reply* at t_n)

$$= Pr(U(e, PB, \textit{Reply}, T_{b,N}, t_p, \{m\}) > U(e, PB, \textit{WaitUntilNext}, T_{b,N}, t_p, \{m\})),$$

where t_p will be replaced, for a given policy, by the value called for in that policy.

If *Provider* δ = *Reply*, t_p is the production time just before t_n, and we use Eq. (6-1) to estimate t_p. Thus $t_p = T^0_{a,P} + Z\times\tau_p$. As before, $L_f = L_{wl-1}$. Since $t < T^0_{a,P} < T_{b,N}$, $t_{cl} = T^0_{a,P}$. Then:

$$\sigma = \max(0, t_p - t_{cl}) = Z\times\tau_p.$$

L_{cl} is the carrier's location at the current time $T^0_{a,P}$. L_f is approximated by the last wumpus location which the carrier sent to the fighter at time $T^0_{a,P}$. In conclusion,

the carrier uses estimated $t_p = T^0_{a,P} + Z \times \tau_p$ to calculate σ, uses its current location at $T^0_{a,P}$ as L_{cl}, and uses the wumpus location provided to the fighter at $T^0_{a,P}$ to estimate L_f. Since $T^0_{a,P} < T_{b,N}$, $T_{b,N} - T^0_{a,P}$ would be used to reduce the range of the summations over j and d. This case is *ReducedGeneralCase* because it requires σ and L_{cl} and L_f as inputs and has $t_p < t_n$. $E\{Pr(t_u - t_p < J)\}$ can be calculated with Eq. (D-3).

If *Provider* δ = *WaitUntilNext*, t_p is the production time just after $T^0_{b,N}$, we use Eq. (6-2) to estimate t_p. Thus $t_p = T^0_{a,P} + (Z+1) \times \tau_p$. Since $t < T^0_{a,P} < T_{b,N}$, $t_{cl} = T^0_{a,P}$. Then:

$$\sigma = t_p - t_{cl} = (Z+1) \times \tau_p.$$

L_{cl} is the carrier's location at $T^0_{a,P}$ and L_f is approximated by L_{wl}, which is the wumpus location provided to the fighter at $T^0_{a,P}$. This case is the *GeneralCase* because it requires σ and L_{cl} and L_f as inputs. $E\{Pr(t_u - t_p < J)\}$ can be calculated with Eq. (7-5).

Once the currency for three possible policies can be computed, their utilities can be computed and then Pr(needer decides to keep *Silence* at $T_{b,N}$) is deterministic either 1 or 0.

D.2. Situation PA: the carrier finds a wumpus' location – *Silence*

In this case,

$$Pr(t_u - t_p < J),$$

where t_p may be before, equal to, or after $T^0_{a,P}$, depending upon decisions the agents make, and t_u will be the time at which the fighter arrives at the wumpus' location. This probability can be evaluated conditionally on two sub-cases:

$$\Pr(t_u - t_p < J) \times \Pr(T_{b,N} \leq T^0_{a,P})$$

$$+ \Pr(t_u - t_p < J) \times \Pr(T^0_{a,P} < T_{b,N})$$

$\Pr(T_{b,N} \leq T^0_{a,P})$ and $\Pr(T^0_{a,P} < T_{b,N})$ are calculated in Appendix A. We now consider $E\{\Pr(t_u - t_p < J)\}$ for two sub-cases.

D.2.1. Sub-case PS–1 $T_{b,N} \leq T^0_{a,P}$

In this case, the needer could not have chosen *ActiveAsk* because otherwise, the needer would have asked and the provider would be obligated to provide the value at $T^0_{a,P}$ and could not choose *Silence*. Therefore the needer must be either *Wait*ing for a proactive tell from the provider or keeping *Silence,* so we use $I_P(T^{ls}_{a,P})$. We consider combinations of these two cases:

$$\sum_{n=5}^{6} \Pr((t_u - t_p < J \mid T_{b,N} \leq T^0_{a,P} \wedge E_n) \times \Pr(E_n),$$

where E_n, n=5 and 6, denote the following events:

E_5: needer decides to *Wait* at $T_{b,N}$;

E_6: needer decides to keep *Silence* at $T_{b,N}$.

We first calculate $E\{\Pr(t_u - t_p < J)\}$ for E_5 and E_6 and then compute $\Pr(E_5)$ and $\Pr(E_6)$.

D.2.1.1. Calculating $E\{\Pr(t_u - t_p < J)\}$ for E_5 and E_6

- *PS–1 E_5: needer decides to Wait at $T_{b,N}$*

In this case, we have $t = t_{cl} = T^0_{a,P}$, and $t_n = T_{b,N}$, which is unknown. Since the carrier knows t_{cl}, so L_{cl} is known. t_p is the next *ProactiveTell* time, which will be at some future production time (not necessarily the next production time). Since by assumption there is a need at t, the fighter must have chased the wumpus whose location was most recently sent to it. This wumpus location was denoted by L_{wl} which is the wumpus' location at time $T^{ls}_{a,P}$. As before, we approximate L_f by L_{wl}. Since t_p is the production time just after $T^0_{b,N}$, we use Eq. (6-2) to estimate t_p. Thus $t_p = T^0_{a,P} + (Z+1)\times\tau_p$. We approximate σ as:

$$\sigma = t_p - t = T^0_{a,P} + (Z+1)\times\tau_p.$$

All of the parameters needed for evaluating Eq. (7-5) (the *GeneralCase*), σ, L_{cl} and L_f, are thus estimated and this equation can be used to estimate $E\{Pr(t_u - t_p < J)\}$.

- *PS–1 E_6: needer decides to keep Silence at $T_{b,N}$*

In this case, $t = t_{cl} = T^0_{a,P}$, $t_n = T_{b,N}$, which is unknown, and $t_p = T^{ls}_{a,P}$, the known time of the most recent value for *I* the needer has. $T^{ls}_{a,P}$ must be less than t, and must correspond to the time at which some previous proactive tell occurred. As an estimate, we will consider the two most recent wumpus locations sent to the fighter and assume that the fighter has just killed the next-to-last wumpus and is about to chase the most recent wumpus, whose location was sent. While this situation is not guaranteed, it is the most likely situation. We also take the wumpus location that will be sought to be $L_w = L_{wl}$, and hence it is known. Having this estimate for L_f and L_w, D_k turns out to be

a fixed number. Since $t_n > t_p$, we also need to consider unknown t_n. Therefore $t_n - t_p$ would be used to reduce the range of the summations over j and d. We approximate t_n by

$$T_{b,N} = \max(T^{ls}_{a,P}+1, \min(T^{ls}_{a,P} + \tau_n/2, T^{0}_{a,P})).$$

Since D_k and t_p are known, and $t_n > t_p$, this is *ReducedSimpleCase* and then $E\{Pr(t_u - t_p < J)\}$ can be estimated with Eq. (D-2).

D.2.1.2. Calculating Pr(E_n)

- $Pr(E_6)$: Pr(needer decides to keep *Silence* at $T_{b,N}$)

Pr(needer decides to keep *Silence* at $T_{b,N}$)

=Pr(U(e, NA, *Silence*, $T_{b,N}$, $T^{ls}_{a,P}$, {m}) > U(e, NA, *Wait*, $T_{b,N}$, $T^{0}_{a,P}$, {m}),

where as before, we estimate $T_{b,N} = \max(T^{ls}_{a,P}+1, \min(T^{ls}_{a,P} + \tau_n/2, T^{0}_{a,P}))$. We assume $L_{wl\text{-}1}$ is the wumpus' location the carrier sent just before $T^{ls}_{a,P}$ and the carrier knows this time.

The solution will involve (the provider) evaluating the utility function under each of the possible policies the needer might make at $t_n = T_{b,N}$. The evaluation of the utility function includes cost, timeliness and currency (risk is only associated with the carrier). Having t_p and t_n, determinations of cost and timeliness are straightforward. Next we consider the currency for each policy.

Needer δ = Silence

The evaluation of *Silence* for the needer under the condition that a need has arisen is given in Section D.2.4.6, which requires the following inputs for calculating the currency: D_k, D_s, D_n, t_n and t_p. Since we are evaluating the carrier's estimation for the needer's use of *Silence* at t_n, the carrier will use its own knowledge to fill in these inputs. Let's consider these inputs one by one.

D_k. Under our assumption, the fighter will be at L_{wl-1} at time t_n. This is also the time at which the fighter begins to chase the wumpus at L_{wl}, thus $L_f = L_{wl-1}$. We also take the wumpus location that will be sought to be $L_w = L_{wl}$, and hence L_w is known. Having this estimate for L_f and L_w, D_k turns out to be a fixed number.

D_s and D_n. During the time interval $[T^{ls}_{a,P}, t_n]$, where $t_n = \max(T^{ls}_{a,P}+1, \min(T^{ls}_{a,P} + \tau_n/2, T^0_{a,P}))$, the carrier is able to determine D_s, the length of time between when the carrier first saw this wumpus and when it last saw this wumpus, and D_n, the time duration since the carrier last saw the wumpus. Having these inputs, the carrier is able to estimate the currency given in Section D.2.4.6.

Needer δ = Wait

The evaluation of *Wait* for the needer under the condition that a need has arisen is given in Section D.2.4.7, which requires the following inputs for calculating the currency: σ, L_f and L_{cl}. However, in this case, $t_p = t_{cl} = T^0_{a,P}$, and $\sigma = \max(0, t_p - t_{cl}) = 0$. Since $\sigma=0$, the carrier did not move so L_{cl} is irrelevant. Also as before L_f is approximated by L_{wl-1}. Having estimations for L_f, this fits the form of Sub-case PT-1,

which also has σ=0. The currency will be estimated with the same form of Sub-case PT-1 (Eq. (D-1) for the *SimpleCase*).

Once the currency for three possible policies can be computed, their utilities can be computed. Consequently the estimated needer's decision is deterministic, thus the needer will choose a policy which has the max utility. So the probability that the needer will choose this policy is 1. Since the needer only can make one decision at a decision point, so the probability of choosing other policies is 0. Then Pr(needer decides to keep *Silence* at $T_{b,N}$) equals either 0 or 1.

- *Pr(E_5):* Pr(needer *decides to Wait at* $T_{b,N}$*)*

  The event E_5 is complement of E_6. $Pr(E_5)=1–Pr(E_6)$.

D.2.2. Sub-case PS–2 $T^0_{a,P} < T_{b,N}$

In this case, t_p and t_u depend upon the decision the needer will make at $T_{b,N}$ and the provider's decision at $T_{b,N}$. We consider combinations of cases:

$$\sum_{n=7}^{10} \Pr(t_u - t_p < J \mid T^0_{a,P} < T_{b,N} \wedge E_n) \times \Pr(E_n),$$

where E_n, n=7,..,10, denote the following events:

E_7: needer decides to *Wait* at $T_{b,N}$;

E_8: needer decides to keep *Silence* at $T_{b,N}$;

E_9: needer decides to *ActiveAsk* at $T_{b,N}$ ∧ provider decides to *Reply* at $T_{b,N}$;

E_{10}: needer decides to *ActiveAsk* at $T_{b,N}$ $\wedge$ provider decides to *WaitUntilNext* at $T_{b,N}$.

Based on Section 6.7.1.2.2, $I_P(t_p)$ will be used for all events. So we first must consider $E\{Pr(t_u - t_p < J)\}$ for all events and then compute $Pr(E_n)$. Generally these computations need parameters: t_p and t_{cl} to calculate $\sigma=\max(0, t_p - t_{cl})$, L_{cl} the carrier's most recent known location, and L_f the fighter's location at time $T_{b,N}$.

D.2.2.1. Calculating $E\{Pr(t_u - t_p < J)\}$ for E_7-E_{10}

- *PS–2 E_7: needer decides to Wait at $T_{b,N}$*

This case is very similar to Sub-case PS–1 E_5. Sub-case PS–1 E_5 also requires that the needer be waiting when the need occurs. Sub-case PS–1 E_5 uses Eq. (7-5) which requires three parameters: σ, L_{cl} and L_f. In order to calculate σ, the carrier needs t_p and t_{cl}. In the present case, t_p can be estimated by the same way in Sub-case PS–1 E_5: $t_p = T^0_{a,P} + (Z+1)\times\tau_P$, and t_{cl} equals to the current time $T^0_{a,P}$. Then L_{cl} is the carrier's current location at $T^0_{a,P}$. As before L_f is approximated by L_{lw-1}. Having σ, L_{cl} and L_f, this is *GeneralCase* and then Eq. (7-5) may be used for currency.

- *PS–2 E_8: needer decides to keep Silence at $T_{b,N}$*

In this case, $t = t_{cl} = T^0_{a,P}$. Since the carrier knows t_{cl}, so L_{cl} is known. As before, $L_f = L_{wl-1}$. Since $t_p < t_n$, $t_n - t_p$ would be used to reduce the range of the summations over j and d. As in Sub-case PS–2, E_5, we approximate t_n by

$$T_{b,N} = \max(T^0_{a,P} + 1,\ T^{ls}_{a,P} + \tau_n/2).$$

t_p is the time at which the last wumpus' location the carrier will have sent to the fighter (not necessarily $T^{ls}_{a,P}$). We use an estimation $T^1_{a,P} = T^0_{a,P} + \tau_P$. Having estimations for t_n and $T^1_{a,P}$, we are able to determine their order. If $T^0_{a,P} < t_n < T^1_{a,P}$, $t_p = T^{ls}_{a,P}$; otherwise we assume t_p equals the most recent production time before t_n: $t_p = T^0_{a,P} + Z \times \tau_P$.

If $t_p = T^{ls}_{a,P}$, this case is very similar to Sub-case PS–1 E_6, the *GeneralCase*, which also required that the needer keep silence when the need occurred. The only difference in the present case is that the range of possible values for $T_{b,N}$ is different. Then, Eq. (7-5) may be used as in Sub-case PS–1 E_6 with the revised value for $T_{b,N}$.

If $t_p = T^0_{a,P} + Z \times \tau_P$, this case is the *ReducedGeneralCase* because it requires σ, L_{cl} and L_f, and has $t_p < t_n$.since $t_{cl} < T^0_{a,P}$, then

$$\sigma = t_p - t_{cl} = Z \times \tau_P + T^0_{a,P} - T^0_{a,P} = Z \times \tau_P.$$

$E\{\Pr(t_u - t_p < J)\}$ can be calculated with Eq. (D-3).

- *PS–2 E_9: needer decides to ActiveAsk at $T_{b,N}$ ∧ provider decides to Reply at $T_{b,N}$*

In this case, $t_{cl} = t = T^0_{a,P}$, and the carrier knows its own location L_{cl}. The fighter will be at L_{wl-1} at time $T_{b,N}$; thus $L_f = L_{wl-1}$. Since $t_p < t_n$, $t_n - t_p$ would be used to reduce the range of the summations over j and d. As in Sub-case PS–2 E_7, we approximate t_n by

$$T_{b,N} = \max(T^0_{a,P} + 1,\ T^{ls}_{a,P} + \tau_n/2).$$

t_p could be either $T^0_{a,P}$ or some production time greater than $T^0_{a,P}$. Different values for t_p will lead to σ equal to zero or not and this consequently will result in using different forms to calculate $E\{Pr(t_u - t_p < J)\}$.

As before we use an estimation $T^1_{a,P} = T^0_{a,P} + \tau_P$. Having estimations for t_n and $T^1_{a,P}$, we are able to determine their order. If $T^0_{a,P} < t_n < T^1_{a,P}$, $t_p = T^0_{a,P}$. This case is exactly the same with Sub-case PT-2 E_2, where the fighter also will use $I_P(t_p)$. $Pr(t_u - t_p < J)$ will be estimated with the same form of Eq. (7-5) in Sub-case PT-2 E_2 (the *GeneralCase*).

If $T^1_{a,P} \le t_n$, t_p is the production time just before t_n; we use Eq. (6-1) to estimate t_p: $t_p = T^0_{a,P} + Z \times \tau_P$. Then:

$$\sigma = t_p - t_{cl} = Z \times \tau_P + T^0_{a,P} - T^0_{a,P} = Z \times \tau_P.$$

This case is *ReducedGeneralCase* because it requires σ, L_{cl} and L_f as inputs, and uses $t_n - t_p$ to reduce summations. $E\{Pr(t_u - t_p < J)\}$ will be estimated with Eq. (D-3).

- *PS–2 E_{10}: needer decides to ActiveAsk at $T_{b,N}$ ∧ provider decides to WaitUntilNext at $T_{b,N}$*

In this case, $t_{cl} = t = T^0_{a,P}$, and the carrier knows its own location L_{cl}. The fighter will be at L_{wl-1} at time $T_{b,N}$, thus $L_f = L_{wl-1}$. As in Sub-case PS–2, E_7, we approximate t_n by

$$T_{b,N} = \max(T^0_{a,P} + 1,\ T^{ls}_{a,P} + \tau_n/2).$$

The carrier will reply the next wumpus' location found after t_n. t_p can be approximated by Eq. (6-2): $t_p = T^0_{a,P} + (Z+1)\times\tau_P$. Then:

$$\sigma = t_p - t_{cl} = T^0_{a,P} + (Z+1)\times\tau_P - T^0_{a,P} = (Z+1)\times\tau_P.$$

This case is *GeneralCase* because it requires σ, L_{cl} and L_f as inputs. Eq. (7-5) may be used for currency.

D.2.2.2. Calculating Pr(E_n)

We consider Pr(E_n) (n=7-10):

E_7: needer decides to *Wait* at $T_{b,N}$;

E_8: needer decides to keep *Silence* at $T_{b,N}$;

E_9: needer decides to *ActiveAsk* at $T_{b,N}$ $\wedge$ provider decides to *Reply* at $T_{b,N}$;

E_{10}: needer decides to *ActiveAsk* at $T_{b,N}$ $\wedge$ provider decides to *WaitUntilNext* at $T_{b,N}$.

These events cannot occur simultaneously. In order to estimate the needer's decisions, the provider must compute utilities for all needer's possible policies. Once the utilities can be computed, the estimated needer's decision is deterministic, thus the needer will choose a policy which has the max utility. So the probability that the needer will choose this policy is 1. Since the needer only can make one decision at a decision point, so the probability of choosing other policies is 0. For example, if the utility of *Silence* is greater than that of *Wait* and *ActiveAsk*, then Pr(needer decides to

keep *Silence* at $T_{b,N}$) = 1 and Pr(needer decides to *Wait* at $T_{b,N}$) = 0 and Pr(needer decides to *ActiveAsk* at $T_{b,N}$) = 0. Therefore we can identify that Pr(E_n) is deterministic either 1 or 0. This means these probabilities are dependent with each other. For example, in order to compute Pr(E_7) which is about policy *Wait*, we must also consider the other two possible policies *Silence* and *ActiveAsk*, and if Pr(E_7)=1 then the other two policies become impossible decision for the needer. Since consideration of the probability about one of the needer's decisions includes evaluating utilities for all policies, here we only show the probability about one of the needer's decisions and the probability about one of the provider's responding decisions, and then the probabilities for all four events can be determined. We choose Pr(needer decides to keep *Silence* at $T_{b,N}$) and Pr(provider decides to *Reply* at $T_{b,N}$).

In order to compute Pr(needer decides to *Wait* at $T_{b,N}$), again we need to compute utility for all *Wait*, *Silence* and *ActiveAsk*. This process is very similar to what has been done in Sub-case PT–2 where we computed Pr(needer decides to *Wait* at $T_{b,N}$) under the condition that $T_{b,N}$ > t. There is only one difference between Sub-case PT–2 and the present case. For the present case, since the provider won't provide $I_P(T^0_{a,P})$, so for the case of needer's δ= *Silence*, if $T^0_{a,P} < T_{b,N} < T^1_{a,P}$, $t_p = T^{ls}_{a,P}$ ($t_p = T^0_{a,P}$ for the previous case). Therefore in this case $\sigma = \max(0, t_p - t_{cl})$ will be computed with $t_p = T^{ls}_{a,P}$. Except this difference, any other computation for this case is exactly the same with that of Sub-case PT–2. So Pr(needer decides to *Wait* at $T_{b,N}$) is computable.

In order to compute Pr(provider decides to *Reply* at $T_{b,N}$), again we need to compute utility for both *Reply* and *WaitUntilNext*. This process is exactly the same as Sub-case PT–2 needer δ=*ActiveAsk*, where we computed Pr(provider decides to *Reply* at $T_{b,N}$) under the condition that the needer *ActiveAsk*s for *I* at t_n and $t_n > t$. So we can compute Pr(provider decides to *Reply* at $T_{b,N}$) with the same manner.

D.3. Situation PB: the carrier receives a request from a fighter – *Reply*

In this case, $t_n = t = T_{b,q}$, $t_u = T_{b,q}+D_k$ and $t_p = T_{a,P}^{q0}$, which is the latest production time before the request time, $T_{b,q}$. Also, $t_p < t_n$, and the carrier knows both of them. Because the carrier is assumed to reply with the last observed wumpus' location, it also knows L_w. We presume that in addition to asking the carrier the location of the wumpus, the fighter will also tell the carrier its own location so that the carrier can use it as a point of reference. Therefore L_f is known, and hence D_k is determined.

During the time interval $(t_p, t]$, the carrier may still be able to sense the wumpus some of the time. Therefore $(t_p, t]$ could be divided into two time durations: D_s, the length of time between the carrier's first sight of this wumpus and its last sight of this wumpus, and D_n, the time duration from the carrier last saw the wumpus to the current time. D_s will decrease the hypothesis space of D_0 from [1, 40] to $[1, 40–D_s]$. D_n, plus $t_n – t_p$, will increase the lower bound on the values of j and decrease the upper bound on the range of value of d. This case is similar to the *ReducedSimpleCase* because D_k and t_p are known and $t_n > t_p$. $E\{Pr(t_u–t_p<J)\}$ is calculated with

$$E\{Pr(t_x - t_p < J)\}$$

$$= \sum_{d=1}^{40-D_s-(t_n-t_p+D_n+D_k+1)} \frac{1}{40-D_s} \times \sum_{J=t_n-t_p+D_n+D_k+1}^{40-d-D_s} \frac{1}{40-d-D_s}.$$

D.4. Situation PB: the carrier receives a request from a fighter – *WaitUntilNext*

In this case, since t_p is a future time but t_{cl} is the current time, so $\sigma \neq 0$. We need to use Eq. (7-5) to compute currency. Eq. (7-5) requires three parameters σ, L_{cl} and L_f. For this case, $t = t_{cl} = t_n = T_{b,q}$, $t_p = T_{a,P}^{q1}$, and $t_u = T_{a,P}^{q1} + D_k$; where $T_{a,P}^{q1}$ is the next production time following the request time, $T_{b,q}$. Since the carrier knows t_{cl}, so L_{cl} is known. We presume that the fighter will attach its location to the request at $T_{b,q}$, so L_f is known. And $\sigma = \max(0, t_p - t_{cl}) = t_p - T_{b,q}$. We can use an estimation $t_p = \max(T_{a,P}^{q0} + \tau_P, T_{b,q}+1)$. Then:

$$\sigma = t_p - T_{b,q} = \max(T_{a,P}^{q0} + \tau_P, T_{b,q}+1) - T_{b,q}.$$

This case is the *GeneralCase* because it needs σ, L_{cl} and L_f. Eq. (7-5) will be used for computing currency.

D.5. Situation NA: the fighter needs a wumpus' location – *ActiveAsk*

In this case, $t_n = T_{b,N}^0$. t_p and t_u depend upon the provider's responding decision at $T_{b,N}^0$. We consider the following expression which can be evaluated by considering two sub-cases:

$$\sum_{n=11}^{12} Pr(t_u - t_p < J \mid E_n) \times Pr(E_n),$$

where E_n, n=11 and 12, denote the following events:

E_{11}: provider decides to *Reply* at $T^0_{b,N}$;

E_{12}: provider decides to *WaitUntilNext* at $T^0_{b,N}$.

Below we first consider $Pr(t_u - t_p < J \mid E_n)$ and then $Pr(E_n)$.

D.5.1. Calculating $E\{Pr(t_u - t_p < J)\}$ for E_{11} and E_{12}

- *E_{11}: provider decides to Reply at $T^0_{b,N}$*

In this case, the fighter is the decision maker. The fighter knows $t = t_n = T^0_{b,N}$, but $t_p = T^{a0}_{a,P} < t_n$ and t_p is unknown; $T^{a0}_{a,P}$ is the time at which the carrier most recently produced a wumpus' location.

The fighter will know a set of previous locations of the carrier, because we presume that, in addition to telling the fighter the location of the wumpus, the carrier also tells the fighter its own location. The most recent location of the carrier that the fighter knows is at time $T^{ls}_{a,P}$, which denotes the time at which the carrier last sent a wumpus location to the fighter. Note that $T^{ls}_{a,P}$ must be less than t_p, because the carrier will not choose *Reply* if no new wumpus' location has been produced after t_s. So $t_{cl} = T_{b,r}$[29] and the fighter knows L_{cl}. As before we assume that t_p is the most recent production time just before t_n. Thus we estimate t_p as $t_p = T_{b,r} + Z \times \tau_P$, where $Z = \left\lfloor \frac{T^0_{b,N} - T_{b,r}}{\tau_p} \right\rfloor$. Then:

$$\sigma = \max(0, t_p - t_{cl}) = Z \times \tau_P + T_{b,r} - T_{b,r} = Z \times \tau_P.$$

[29] Recall from Chapter VI that $T_{b,r}$ is the most recent time at which the needer received a message from the carrier.

The fighter can obtain τ_P from the data about the average time between (new) wumpus findings sent from the carrier.

This case is *ReducedGeneralCase* because it needs σ, L_{cl} and L_f, and has $t_p < t_n$. $E\{Pr(t_u - t_p < J)\}$ will be calculated with Eq. (D-3).

- *E_{12}: provider decides to WaitUntilNext at* $T^0_{b,N}$

In this case, $t = t_n = T^0_{b,N}$, which is known, and $t_p = T^{a1}_{a,P}$, the next time at which the carrier finds a wumpus, which is unknown. Thus we estimate t_p by $t_p = (Z+1)\times\tau_P + T_{b,r}$. Then:

$$\sigma = \max(0, t_p - t_{cl}) = (Z+1)\times\tau_P + T_{b,r} - T_{b,r} = (Z+1)\times \tau_P.$$

This case is the *GeneralCase* because it needs σ, L_{cl} and L_f. $E\{Pr(t_u - t_p < J)\}$ will be calculated with Eq. (7-5).

D.5.2. Calculating Pr(E_n)

Next we consider $Pr(E_{11})$ and $Pr(E_{12})$, the needer's estimate to the provider's responding decision.

- *$Pr(E_{11})$: Pr(provider decides to Reply at* $T^0_{b,N}$*)*

$$\Pr(\text{provider decides to } \textit{Reply} \text{ at } T^0_{b,N})$$

$$= \Pr(U(e, PB, \textit{Reply}, T^0_{b,N}, t_p, \{m\}) > U(e, PB, \textit{WaitUntilNext}, T^0_{b,N}, t_p, \{m\})),$$

where $t_n = T^0_{b,N}$ for both policies.

If provider δ = *Reply*, t_p is the production time just before $t_n = T^0_{b,N}$. We estimate t_p as $t_p = T_{b,r} + Z \times \tau_P$, where $Z = \left\lfloor \frac{T^0_{b,N} - T_{b,r}}{\tau_p} \right\rfloor$. Since the fighter may not know the carrier's location from the fighter's limited observability. Hence, we use the location sent at time $T_{b,r}$. If Z=0 which means that no new production after the last told, Pr(provider decides to *Reply* at $T^0_{b,N}$) = 0, because the carrier won't resend the last told. Otherwise $\sigma = \max(0, t_p - t_{cl}) = t_p - t_{cl}$. Also the fighter knows its own location L_f at t_n. Since $t_p < t_n$, $t_n - t_p$ would be used to reduce the range of the summations over j and d. Therefore this case is *ReducedGeneralCase* because it needs σ, L_{cl} and L_f, and has $t_p < t_n$. $Pr(t_u - t_p < J)$ can be estimated with Eq. (D-3).

If provider δ = *WaitUntilNext*, t_p is the production time just after $T^0_{b,N}$. We estimate t_p as $T_{b,r} + (Z+1) \times \tau_P$, where $Z = \left\lfloor \frac{T^0_{b,N} - T_{b,r}}{\tau_p} \right\rfloor$. Also $\sigma = \max(0, t_p - t_{cl}) = t_p - t_{cl}$. Meanwhile the fighter knows its own location L_f at t_n. This case is the *GeneralCase* because it needs σ, L_{cl} and L_f. $Pr(t_u - t_p < J)$ can be estimated with Eq. (7-5).

After computing the currency, the risk can be estimated using Table 2 and hence *Pr(provider decides to Reply at* $T^0_{b,N}$*)* be estimated to be either 1 or 0.

- *Pr(E_{12}): Pr(provider decides to WaitUntilNext at* $T^0_{b,N}$*)*

  The event E_{12} is the complement of E_{11}. So $Pr(E_{12}) = 1 - Pr(E_{11})$.

D.6. Situation NA: the fighter needs a wumpus' location – *Silence*

In this case, $t=t_n=T^0_{b,N}$, $t_p=T_{b,r}$, the time at which agent *b* most recently received a wumpus location, and $t_u=T^0_{b,N}+D_k$. The fighter knows L_f and L_w; hence D_k is determined. Likewise, t_p, and t_n are known, hence all other variables including L_c, L_w and L_f are known. This fits the *ReducedSimpleCase* and we can calculate the concurrency with Eq. (D-2).

D.7. Situation NA: the fighter needs a wumpus' location – *Wait*

In this case, $t_n=T^0_{b,N}$. t_p is the next *ProactiveTell* time after $T^0_{b,N}$. Since there is a need, we assume t_p is the next production time after $T^0_{b,N}$. Thus we t_p as $T_{b,r}+(Z+1)\times\tau_P$. Then:

$$\sigma = \max(0, t_p - t_{cl}) = (Z+1)\times\tau_P + T_{b,r} - T_{b,r} = (Z+1)\times \tau_P.$$

This case is the *GeneralCase* because it needs σ, L_{cl} and L_f. $E\{Pr(t_u-t_p<J)\}$ can be estimated with Eq. (7-5).

D.8. Situation NB: the fighter receives a wumpus' location

In this situation, the fighter will *Accept* the wumpus location received. The decision is deterministic, so we do not consider this case.

VITA

Yu Zhang received a B.S. and a M.S., both in computer science, from Central South University, P.R.China in 1995 and 1998 respectively, and a Ph.D in computer science from Texas A&M University in December 2005.

She has been employed as a research assistant in the Department of Computer Science at Texas A&M University from 1999 to fall 2003 and 2004. During the fall of 2003, she was a lecturer of an undergraduate core course, CPSC206 Programming in C, for the same department.

She has over 10 papers published in journals and refereed conferences during her study at Texas A&M University. Her research interests include distributed intelligent systems, uncertainty in AI, machine learning, human-computer interaction, knowledge representation and modeling, and robotics automation and tele-operation. She currently is an assistant professor in the Department of Computer Science at Trinity University. She can be reached at:

Yu Zhang
Department of Computer Science
Trinity University
San Antonio, TX 78212-7200.