Reasoning Effectively Under Uncertainty for Human-Computer Teamwork

A dissertation presented
by

Ece Semiha Kamar

to
The School of Engineering and Applied Sciences
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of

Computer Science

Harvard University
Cambridge, Massachusetts
May 2010

©2010 - Ece Semiha Kamar

All rights reserved.

Thesis advisor Author

Professor Barbara J. Grosz **Ece Semiha Kamar**

Reasoning Effectively Under Uncertainty for Human-Computer Teamwork

Abstract

As people are increasingly connected to other people and computer agents, forming mixed networks, collaborative teamwork offers great promise for transforming the way people perform their everyday activities and interact with computer agents. This thesis presents new representations and algorithms, developed to enable computer systems to function as effective team members in settings characterized by uncertainty and partial information.

For a collaboration to succeed in such settings, participants need to reason about the possible plans of others, to be able to adapt their plans as needed for coordination, and to support each other's activities. Reasoning on general teamwork models accordingly requires compact representations and efficient decision-theoretic mechanisms. This thesis presents Probabilistic Recipe Trees, a probabilistic representation of agents' beliefs about the probable plans of others, and decision-theoretic mechanisms that use this representation to manage helpful behavior by considering the costs and utilities of computer agents and people participating in collaborative activities. These mechanisms are shown to outperform axiomatic approaches in empirical studies.

The thesis also addresses the challenge that agents participating in a collaborative activity need efficient decision-making algorithms for evaluating the effects of their actions on the collaboration, and they need to reason about the way other participants perceive these actions. This thesis identifies structural characteristics of settings in which computer agents and people collaborate and presents decentralized decision-making algorithms that exploit this structure to achieve up to exponential savings in computation time. Empirical studies with human subjects establish that the utility values computed by this algorithm are a good indicator of human behavior, but learning can help to better understand the way these values are perceived by people.

To demonstrate the usefulness of these teamwork capabilities, the thesis describes an application of collaborative teamwork ideas to a real-world setting of ridesharing. The computational model developed for forming collaborative rideshare plans addresses the challenge of guiding self-interested people to collaboration in a dynamic setting. The empirical evaluation of the application on data collected from the real-world demonstrates the value of collaboration for individual users and environment.

Contents

List of Figures

List of Tables

Citations to Previously Published Work

Portions of Chapters 2 and 3 have appeared in the following paper:

> "Incorporating Helpful Behavior into Collaborative Planning", E. Kamar, Y. Gal, and B.J. Grosz, In Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2, pages 875882. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

The mobile opportunistic commerce domain used in Chapters 2 and 3 first appeared in the following paper:

> "Mobile Opportunistic Commerce: Mechanisms, Architecture, and Application", E. Kamar, E. Horvitz, and C. Meek, In Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2, pages 10871094. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

Portions of Chapter 5 have appeared in the following paper:

> "Applying MDP Approaches for Estimating Outcome of Interaction in Collaborative Human-Computer Settings", E. Kamar and B.J. Grosz, Multi-Agent Sequential Decision Making in Uncertain Domains, pages 2532, 2007.

Portions of Chapters 5 and 6 have appeared in the following paper:

> "Modeling User Perception of Interaction Opportunities for Effective Teamwork", E. Kamar, Y. Gal, and B.J. Grosz, In Proceedings of the 2009 International Conference on Computational Science and Engineering-Volume 04, pages 271277. IEEE Computer Society, 2009.

Significant portions of Chapter 7 have appeared in the following papers:

> "Collaboration and Shared Plans in the Open World: Studies of Ridesharing", E. Kamar and E. Horvitz, In Proceedings of the 21st International Joint Conference on Artificial Intelligence, pages 187194, 2009.
>
> "Generating Shared Transportation Plans Under Varying Preferences: Ridesharing Models and Mechanisms", E. Kamar and E. Horvitz, Technical Report, MSR-TR-2009-2011, Microsoft Research, 2009.

The research presented in Chapter 7 was done during an internship at Microsoft Research.

Acknowledgments

I owe the greatest debt of gratitude to my advisor Barbara J. Grosz. Since the first day I met her, Barbara has provided me with invaluable insights about research, technical writing and life, and has encouraged me every day to study what is exciting and become better at what I do. She has provided me with the freedom to become an independent researcher yet has always been there when I needed her. This thesis would not be possible without her guidance.

I consider myself extraordinarily fortunate to have Eric Horvitz as my mentor. Eric is an amazing and inspirational researcher with exciting ideas and endless passion for real-world systems. Working with him has shown me infinite opportunities for using Artificial Intelligence techniques to help people with everyday tasks, and has motivated me even more to pursue my research. Knowing him and working with him have been a pleasure.

I would like to thank David Parkes for his excellent guidance on my research. His feedback has helped to better formulate my ideas and present them in this thesis. Thanks also to former and current faculty members of the Harvard Artificial Intelligence Research Group (AIRG), in particular to Avi Pfeffer for helping with mathematical models, Stuart Shieber for advising me during my first year, Radhika Nagpal for relaxing me when I get stressed, and Krzysztof Gajos for introducing me to the rich literature on Human-Computer Interaction.

I am grateful to Kobi Gal and Dudi Sarne for their contributions to this thesis. Kobi has been a collaborator and friend over the past five years. He has always made time to discuss research ideas, go over papers, and talk about the ever-changing situation in the Middle East. I was also fortunate to have Dudi as my mentor during the first two years at Harvard. The research we did together served as the starting point of this thesis.

During my five-years as a member of AIRG, I had many friends whom I learnt so much from and hope to work with in the future. Philip Hendrix, who was my officemate for four years, taught me a lot about American culture, made conferences a lot of fun and proofread countless versions of my papers. The time I spent in the office became even more enjoyable with the arrival of my Greek officemate Dimitrios Antos. The other two Turkish members of AIRG, Emir Kapanci and Elif Yamangil, made me feel at home in the department. Heather Pon-Barry, whom I enjoyed spending time with, helped me with technical writing.

I would like to thank my friends in Providence for their all the fun times we spent together. They have been my family away from home and my life has been so much richer thanks to them. In particular, I would like to thank to Alptekin Kupcu, Gavril Bilev, Feryaz Ocakli and Onur Domanic.

Lastly, and most importantly, I wish to thank my parents Adalet and Tuncel and my life partner Ibrahim for their continued support, encouragement and for enduring my complaints. My parents have devoted their life to my brother and me, and highly prioritized our education. They have given me all the opportunities in the world to choose what I want to be and supported me all the way. Ibrahim has believed in me more than I believed in myself. He held my hand through big transitions in our lives and helped me to achieve the best that I could.

This thesis work was supported by Microsoft Research Fellowship Program, the Robert L. Wallace Prize Fellowship, a subcontract to SRI International's DARPA Contract No. FA8750-05-C-0033and NSF grants IIS-0705406, and IIS-IIS-0705587.

Annem Adalet Kamar'a, babam Tuncel Kamar'a
ve hayat arkadasim Ibrahim Eden'e ithaf edilmistir.

Dedicated to my parents Adalet and Tuncel Kamar
and to my life partner Ibrahim Eden.

Chapter 1

Introduction

Collaboration is a special kind of group activity in which participants work together towards the achievement of a shared goal. Working as a team may help participants to bring together diverse capabilities and accomplish tasks that they could not do as easily alone. Collaboration occurs frequently in daily life, usually among people situated together or connected with an organizational or social structure; for instance people coming together to cook a pot-luck dinner, musicians in an orchestra, rescue teams working together in a rescue site. With the increasing connectivity provided by computer networks, there is opportunity for collaborations among computer agents and people who may be located in diverse locations without organizational or social structure. Enabling collaboration in such "mixed networks" of computer agents and people offers great promise for transforming the way people perform their everyday activities and interact with computer agents.

This thesis looks beyond individual intelligent systems and considers mixed networks of computer agents and people working together. For a successful partnership with people in real-world settings, computer agents need to reason about the world and the way their partners are performing their part of the collaborative activity, even though they have only partial observations about the world and their partners' activities and the world is dynamically changing. They need efficient decision-making algorithms for determining how to contribute to the collaborative activity and support their partners' efforts.

This thesis defines new representations that enable agents to reason about their partners' plans without knowing all the details of the collaborative activity. It describes several efficient decision-making algorithms for determining how to contribute to the collaborative activity and support their partners when needed, and it presents theoretical and empirical investigations of these representations and algorithms in settings analogous to the real-world.

These representations and decision-making algorithms address the challenges of representing uncertainty and making decisions effectively in settings with partial information and uncertainty. The representations handle the uncertain and partially observable nature of real-world domains. The decision-making strategies make use of these representations to make decisions in uncertain environments with partial knowledge by considering the costs and benefits of actions on the collaborative utility. They enable computer systems to carry out collaborative activities effectively in mixed networks. This research thus contributes to the overarching goal of designing computer agents as problem solving partners in settings in which they interact with people.

This thesis also addresses the novel challenges introduced by the inclusion of people in collaborative activities with computer agents. Understanding the way people perceive collaborative activities will lead to the design of computer agents that can work better with people. In particular, the thesis considers the undesirable and expensive nature of interrupting and interacting with people frequently. It introduces the approach of nearly decomposable multi-agent planning models that achieve efficiency in reasoning by eliminating, when they are not needed, close coupling and continuous communication among computer agents and people.

When people participate in collaborative activities with computer agents, it is also necessary for the agents to reason about the way people make decisions and perceive the utility of the collaboration and the actions performed by the agents. People's decision making may differ from decision-theoretic models because they may not be computationally unbounded and fully rational; they may not always make decisions that maximize utility. This thesis presents empirical investigations of the mismatch between the utility of collaboration computed by the decision-theoretic planning models and people's perception of it. It also explores the way learning from human responses may help to improve the interactions between computer agents and people.

The development of more complete theories of collaboration, decision-making models and representations for collaboration-capable computer agents offers practical impact and value for people's everyday activities. Throughout the thesis, the presentations of the theoretical work is supported with real-world examples and the empirical evaluations use settings that are analogues of the real-world. The thesis concludes with a description of an application of these collaborative teamwork theories to a real-world setting that brings self-interested people together in collaborative rideshare plans.

1.1 Reasoning about Teammates' Plans

The participants of a collaborative activity form coordinated, mutually supportive plans. They make commitments to the group activity, to doing some of the constituent tasks of that activity, and to other participants' ability to accomplish other constituent actions. As in most multi-agent task settings, the collaborative activity is carried out in a world that is constantly changing, the participants' knowledge about the world is inherently incomplete, individuals have (sensory) access to different parts of the world, and their beliefs–including their beliefs about how best to perform an action–may differ. Although participants of a collaborative activity are not required to know all the details of the plans for certain constituent actions followed by other members of the group to accomplish the activity, they still need to reason about the way their partners are performing these constituent actions to succeed in doing the collaborative activity.

Throughout this thesis, various aspects of collaborative teamwork will be illustrated with an example of two agents, Alice and Bob, who are cooking for a dinner party. Alice is committed to making the entree and Bob is committed to making an appetizer. Bob may not know which entree Alice is making, but he is committed to the success of the collaborative activity as a whole. While preparing the entree, Alice may have incomplete information about the world. For instance, she may not know that some ingredients are not available. Moreover, the conditions in which the collaborative activity is being done may change stochastically. The oven that Alice plans to use for her entree may break down unexpectedly. To fulfil his commitment to the success of the dinner-making plan and to be able to help Alice with her entree plan when needed, Bob needs to reason about the way Alice is making the entree. For instance, informing Alice about the broken oven may be beneficial if Alice's plan includes using the oven, but doing so may be costly otherwise. For a successful partnership analogous to the one between Bob and Alice, computer agents need representations for handling uncertainty and partial information, and decision-making models that can use these representations to reason about the costs and utilities of actions on the collaborative activity.

This thesis expands prior formal teamwork models to treat the uncertain and partially observable nature of real-world domains. Formal models of teamwork specify requirements for having a successful collaborative activity, but do not provide implementations that handle uncertainty (Cohen and Levesque, 1991; Grosz and Kraus, 1996). The representations and decision-making models defined in the thesis bridge a gap in these teamwork theories by incorporating costs and utilities in a principled and general way, and integrating decision-theoretic reasoning. These models enable a computer system to weigh both the costs and utilities of a helpful action on the collaborative activity to reason about the utility of helping a partner analogous to the way Bob would do if he was making a decision about helping Alice in the dinner-making example.

Key to this integration is Probabilistic Recipe Trees (PRTs), a structured tree representation for agents' beliefs about the way a collaborative activity is being done by the members of a group. This representation can represent Bob's beliefs about different recipes Alice and he can use to make dinner and their likelihoods. The PRT representation is exponentially more compact than an exhaustive probability distribution over the possible ways for doing an action. It represents agents' beliefs about the constituent actions of a collaborative activity on separate branches. This modular structure enables the definition of operations on PRTs for adding an agent's beliefs about a new constituent action, for removing some part of the beliefs and for updating an agent's beliefs about the way some part of the activity is being done without the need to update a PRT as a whole. By applying these operations, PRTs can be updated to reflect the changes in agents' beliefs about the way a collaborative activity is being done. The hierarchical structure of the representation allows associating actions on different levels of decomposition with utility and cost values. Propagating these values provides means for computing the expected utility of a collaborative activity based on an agent's beliefs about the way the activity is being accomplished.

We demonstrate the usefulness of PRTs for decision-theoretic reasoning on formal teamwork models in a decision-theoretic mechanism for deciding whether to undertake helpful behavior. Although the participants in a collaborative activity have incentive to help each other, by the nature of their commitments to the shared goal and to each others' actions in service of satisfying that goal, a decision about whether to help still requires deliberation. Helpful actions typically incur cost to the agent helping, and they may incur costs for the group activity as a whole. Typical costs include resources consumed in communicating, lost opportunities to do other activities, and the need for group members to adapt their individual plans to the helpful act or its effects. Thus, even in collaborative settings, agents must weigh the trade-off between the potential benefit to the group of some helpful behavior and its associated costs.

In the dinner-making example, Bob may consider helping Alice by informing her that there are no fresh tomatoes in the kitchen. Bob incurs costs for informing Alice because his recipe may be ruined if he takes too long to find Alice and inform her. To make this decision in a way that is beneficial for the dinner-making activity, Bob needs to reason about the likelihood that Alice's recipe involves tomatoes, the way Alice's plan for the entree would change after knowing about tomatoes, and the cost of informing her. Other possible ways Bob can help Alice may be responding to her if she asks about tomatoes and buying some tomatoes for her as a way of helpful act.

The decision-theoretic mechanism presented in this thesis for managing helpful behavior addresses the intertwined problems of recognizing when help is needed in a collaboration and determining whether to help, taking into account the costs of a helpful action and its possible effects on the beliefs and commitments of group members. The mechanism considers three distinct types of helpful behavior: adopting commitment to perform a helpful action that is outside the scope of an agent's individual responsibility in the collaborative activity, abandoning a commitment to perform a constituent action of the collaborative activity to improve the success of the activity, and communicative actions, including two sub-types: informing actions and asking actions. Agents using this mechanism decide whether to undertake helpful behavior by trading off the cost and utility of doing so.

We investigate empirically the value of the PRT representation for decision-theoretic reasoning in a collaborative activity in a dynamic setting. The mechanism which uses this representation for reasoning about helpful behavior is evaluated on a game setting developed on the Colored Trails infrastructure (Grosz et al., 2004). The game setting is a realistic analogue of the ways in which goals, tasks and resources interact. Across various costs of helpful behavior, agents' uncertainty about the world and each other's capabilities, agents performed better using this mechanism to make helpful-behavior decisions than using purely axiomatic methods.

1.2 Understanding Human Perception in Teamwork

To collaborate successfully with people, a computer agent needs to reason about the effect of its individual actions on the collaboration, and it needs to be able to decide when and how to act jointly with its partners. Collaboration-capable computer agents must have efficient multi-agent planning algorithms that are effective in domains with uncertainty and partial information. Prior work has shown that optimally solving general multi-agent planning problems under uncertainty and partial information is infeasible even for small sized problems (Pynadath and Tambe, 2002).

This thesis introduces a new approach for efficient collaborative planning for settings in which computer agents work with people. This approach takes advantage of structural characteristics typical of these settings: computer agents and people have individual tasks that they accomplish, but they also collaborate to complete both tasks efficiently. When they are not interacting or acting jointly, they are performing their individual tasks without considering the progress of their partners. In this case, the individual tasks are essentially independent. However, the decisions about when and how to coordinate, to communicate with each other, and to support each other requires reasoning about the joint utility.

Nearly-Decomposable Markov Decision Process (ND-MDP), by defining the properties of transition, reward and observation functions, formalizes this nearly-decomposable structure in a multi-agent planning problem. The ND-DECOP algorithm exploits this nearly-decomposable structure for efficient planning. It distinguishes the set of individual and joint actions and decouples an ND-MDP into individual models that are connected through a type sequence for coordinating joint actions. A type sequence represents a sequence of joint and individual action types that agents agree on together. The algorithm finds the type sequence that maximizes the collaborative utility function. The optimal type sequence determines when and how agents should act jointly, interact with each other and support each other.

For example, a collaborative computer assistant for meeting scheduling and a user who is working on a report may collaborate to complete both tasks as efficiently as possible. If they are not acting jointly or communicating with each other, the agent's decisions about meeting scheduling are independent of the report the user is working on. Similarly, the user does not reason about the meetings scheduled by the agent. However, to determine when and how to interrupt the user to ask for feedback or to acquire the user's preferences, the agent needs to consider the cognitive state of the user, whether the user is in the middle of a paragraph or talking on the phone, and the effect of an interruption on the joint utility. If the assistant continuously interrupts the user, it will disrupt the user's writing process.

The investigations presented in this thesis use interruption management as an example of a decision making capability needed for collaborative activities in which agents are distributed, conditions may be rapidly changing and decisions are made under uncertainty. This interruption management problem has the nearly-decomposable characteristics and thus can be modeled as an ND-MDP. The ND-DECOP algorithm can be used to determine when it is beneficial for the computer agent to interrupt the user. However, the way a person makes decisions about accepting an interruption request initiated by the agent may differ from the ND-DECOP algorithm. Similarly, the user's perception of the collaborative utility may not match the values computed by the algorithm. The failure to consider this mismatch may cause the user to reject a valuable interruption opportunity, thereby turning what could have been a beneficial interaction for the collaboration into a performance degrading disturbance.

When people participate in collaborative activities with computer agents, it is no longer sufficient for computer agents to act on the fully-rational estimate of the collaborative utility. Rather, they need to reason about the ways in which people perceive the utility of the collaboration and its constituent actions. This thesis empirically investigates the mismatch between the actual utility of an action in a collaborative context and people's perception of it, exploring the different factors that may influence people's perception of this utility. A myopic variant of the ND-DECOP algorithm is applied to the interruption domain to construct a computer agent for investigating the way people perceive interruptions. These experiments investigated the effect of three factors on human perception of the usefulness of interruption requests: the magnitude of the interruption utility, the timing of interruptions and the perceived type of the partner (human or computer agent). The results revealed that the benefit of interruptions to both computer agents and people is the major factor affecting the likelihood that people will accept interruption requests. However, for those cases in which the benefit of interruption is ambiguous, people prefer to accept those interruptions that originate from other people.

Based on the empirical results showing that some additional factors other than the collaborative utility may affect the way people make interruption decisions, this thesis investigates whether learning from human responses can help to better interact with people. To investigate this problem, several well-known learning algorithms were applied to the data collected from the human studies to build predictive models of the way subjects responded to interruptions. The results showed that learning improves the prediction accuracy when collaborative utility values computed by the planning algorithms are provided as features for learning. They also show that a hybrid model that can learn both the personal and general characteristics of human decision making can predict human responses in a better way.

1.3 An Application of Collaboration in Open World

Self-interested agents have incentives to participate in a collaborative activity if doing so improves their individual utilities. Generalizing formal models of teamwork, which assume the participants of a collaborative activity share a joint utility function, to settings with self-interested agents is not trivial. It requires reasoning about incentives to bring them together in collaborative activities. This thesis addresses the challenges of guiding self-interested people to collaboration in dynamic settings. In particular, it explores different payment mechanisms under real-world considerations by addressing their limitations and computational requirements.

The investigations of teamwork models for self-interested agents use the domain of ridesharing to demonstrate the value of collaboration in real-world settings. The participants of a rideshare plan bear uneven costs for accomplishing the plan and generating value for the whole group. For sustaining the collaboration, they need to be compensated properly. This thesis presents a complete computational model for forming collaborative rideshare plans and for providing fair incentives to self-interested participants efficiently under the limitations of a dynamic system.

The operation of the computational model for ridesharing is empirically evaluated on GPS traces collected from a community of commuters. They indicate significant reductions on number of trips and on total cost of transportation, and they show promise for generating efficiency by bringing self-interested agents together in real-world domains.

1.4 Contributions and Thesis Overview

The major contributions of this thesis arise from its creation of representations and decision-making models for reasoning effectively under uncertainty for successful teamwork among computer agents and people. In particular, this thesis makes the following contributions:

- It provides a probabilistic representation, Probabilistic Recipe Trees (PRTs), for agents' beliefs about the way a collaborative activity is being accomplished. This representation fills a gap in formal teamwork theories by incorporating costs and uncertainty in a principled and general way and enabling tractable decision-theoretic reasoning on teamwork theories despite incomplete information. (Chapter 2)

- It demonstrates the usefulness of PRTs for reasoning about helpful behavior. The decision-theoretic mechanism for managing helpful behavior presented in this thesis determines whether to undertake helpful behavior by taking into account that agents may have only partial information about their partners' plans for sub-tasks of the collaborative activity, the effectiveness of helping may not be known a priori, and helping actions have some associated cost. The empirical evaluations of this decision-theoretic mechanism show that agents using this mechanism to decide whether to help outperform agents using purely axiomatic rules. (Chapter 3)

- It identifies the special structural characteristics that arise in settings in which computer agents and people collaborate. It presents a multi-agent planning algorithm that exploits these characteristics to achieve up to exponential savings in computation time in comparison to general multi-agent planning algorithms while generating optimal and complete joint policies. (Chapter 4)

- It models interruption management as a collaborative decision-making problem and applies efficient multi-agent planning algorithms to compute the utility of interruption by considering both the costs and utilities of an interruption to all participants of a collaborative activity. (Chapter 5)

- It presents an empirical evaluation of the way people perceive collaborative utility when they participate in collaborative activities with computer agents. It explores different factors that may affect the way people make decisions in such settings. In particular, it shows that the magnitude of the utility is the major factor affecting the likelihood that people accept interruptions from computer agents, and that the timing of interruptions and the partner type also matters. It demonstrates that learning from human responses by using decision-theoretic features helps to better predict the way people interact with computer agents. (Chapter 6)

- It presents a complete computational model to guide self-interested agents to collaboration in a real-world application setting. It explores different incentive mechanisms and highlights challenges and trade-offs that arise in applying these mechanisms in such real-life domains. The value of collaboration for users and the environment is demonstrated in the domain of ridesharing. Empirical studies of the ridesharing model on data collected from a community of commuters show that generating collaborative ridesharing plans significantly reduce the number of vehicles on the road, the cost of transportation and gas emissions. (Chapter 7)

Chapter 8 describes related work, and Chapter 9 presents conclusions and a discussion of possible future work.

Chapter 2

Reasoning about Collaborative Plans under Uncertainty

For a successful collaborative activity in a world of uncertainty and partial information, team members need to reason about the possible plans of others, revise their plans accordingly, and support each other when needed. This chapter presents Probabilistic Recipe Trees, a probabilistic representation of agents' beliefs about the probable plans of others. This representation formally integrates costs and utilities into general teamwork models, and enables tractable decision-theoretic reasoning despite having only partial information. Probabilistic Recipe Trees can be used for commitment reconciliation, meshing plans, reasoning about future and managing helpful behavior.

The chapter is organized as follows: Section 2.1 presents an overview of formal teamwork models and introduces the basic constructs used in this chapter. Section 2.2 presents Probabilistic Recipe Trees. Section 2.3 introduces algorithms for evaluating PRTs in terms of their success likelihood, cost and utility.

2.1 Combining Axiomatic and Decision-theoretic Methods

Several formalizations have been proposed to model collaboration and teamwork (Cohen and Levesque, 1991; Dignum and Weigand, 1995; Grosz and Kraus, 1996; Jennings, 1995; Kinny et al., 1992; Levesque et al., 1990). They all use intentions, desires, and beliefs about the world and about each other as the building blocks of collaborative activities. None of these formalizations formally incorporate costs, utilities and probabilities, nor do they include means for agents' beliefs about their partners' plans for the collaborative activity to be represented and used for reasoning under uncertainty.

Unlike other prominent formalizations, the SharedPlan formalization embraces the dynamic nature of real-world teamwork by handling the partiality of collaborative plans, and it is comprehensive in its description of means-end reasoning and constituents of collaborative plans (Grosz and Kraus, 1996, 1999). Further, the SharedPlan formalization does not focus merely on the requirements for agents performing their own part of the activity, but also on the way agents need to reason about each other and support each other when needed. Going beyond the other formalizations, the SharedPlan formalization suggests evaluating utilities and costs to better manage helpful behavior. However, it does not provide representations or algorithms for handling uncertainty regarding the world or agents' capabilities nor does it provide insight about how costs and utilities can be formally integrated into the axiomatic rules of the specification.

The work presented in this chapter builds on the SharedPlan formalization and expands it by defining new predicates, functions and basic constructs that together enable the incorporation of probabilities, costs and utilities into the axiomatic formalization of teamwork.

This section gives an overview of the SharedPlan formalization as well as some examples of teamwork. It concludes with basic definitions for expanding the formalization to enable decision-theoretic reasoning.

2.1.1 Overview of SharedPlan Formalization

The SharedPlan formalization of collaborative activities provides a specification of the capabilities and mental attitudes required to have a successful collaborative plan (Grosz and Kraus, 1996, 1999). A SharedPlan is a collection of individual and mutual beliefs and intentions agents have about a collaborative activity. In addition to the intentions agents hold for performing their own part of the activity, they also have intentions toward the success of the collaborative activity.

The formalization requires agents to form mutual beliefs and to agree on collaborative plans to follow. As agents may have partial information about the world and each other, communication plays a crucial role in forming a successful collaborative plan. Communication requirements emerge from agents' commitments to the success of the collaborative activity. The formalization includes axioms specifying when to communicate or to help in terms of intentions, beliefs and mutual beliefs of agents.

Table 2.1, taken from the original SharedPlan description (Grosz and Kraus, 1996), lists the key requirements for having a collaborative plan for a group action. First, agents need a recipe for accomplishing the action; the recipe defines the set of actions and constraints agents need to satisfy to accomplish the action (Item 1). This requirement assumes that agents have the necessary capabilities for sharing knowledge about recipes they have and

Table 2.1: Key components of collaborative plans.

To have a collaborative plan for an action, a group of agents must have

1. mutual belief of a (partial) recipe

2a. individual intentions that the action be done

2b. individual intentions that collaborators succeed in doing the (identified) constituent subactions

3. individual or collaborative plans for the subactions

for choosing which recipe to use to accomplish the action. The second requirement summarizes the intentions agents should hold to have a SharedPlan. For the actions that agents are performing on their own, they need to commit to performing the actions (Item 2a). As a special requirement for a successful collaborative plan, agents also need to commit to the success of the group plan and consequently to their partners' success (Item 2b). Finally, agents need collaborative and individual plans for doing the constituent actions of the group activity (Item 3). The formalization also requires that agents have decision-making capabilities for forming the plans.

The SharedPlan formalization allows a SharedPlan to be partial to reflect the uncertain and partially observable nature of real-world domains. Agents may initially start with a partial plan and extend it until they have a complete plan. Plans may become incomplete because of changes in the world and thus need to be extended again. Thus, building a collaborative plan is defined as an evolving process in which agents communicate with each other, update the plan as they make new observations about the world and about each other, expand the plan or abandon some parts of the plan when needed. Forming and performing a collaborative plan requires interaction and supportive behavior among the

Table 2.2: Summary of SharedPlan notations.

Type	Notation	Meaning
Modal Operators	Int.To	intend-to
	Int.Th	intend-that
	Bel	belief
Meta-predicates	CBA	can bring about
	CBAG	can bring about group
Act-types for Planning Actions	Select-Rec	agent selects or revises recipe
	Select-Rec-GR	group of agents selects or revises recipe

members of the group as well as capabilities for dynamically updating agents' beliefs and plans individually or as a group.

The operators *Int.To* and *Int.Th* represent intentions that are adopted by agents. *Int.To* is used for representing an agent's commitment to do some action. This type of intention leads to means-end reasoning about how to perform some particular activity. *Int.Th* represents an agent's intention that some proposition will hold. The *Int.Th* operator is key to the success of a collaborative plan. This intention results in agents avoiding conflicting intentions, meshing plans with other agents, helping each other and coordinating with each other when needed. Several axioms presented in the SharedPlan formalism describe the way *Int.Th* results in actions in collaborative activities. (See Grosz and Kraus (1996) for details.)

The SharedPlan formalization requires agents to agree on decision-making procedures for forming collaborative plans, including procedures for selecting recipes, timing actions and assigning agents to do particular actions. The formalization defines two complex actions to refer to these decision-making procedures; *Select_Rec* and *Select_Rec_GR* represent individual and collective planning capabilities agents need to specify the way to perform an action. The formalization does not, however, specify these decision-making procedures, but leaves to agent designers the definition of procedures for agents to use depending on the characteristics of the domain in which they are deployed.

The SharedPlan formalization stipulates that only the agents performing some subactivity know the details of the way the subactivity is being accomplished. Under this formalization, the participants of a collaborative activity are not required to communicate all the details of their part of the plan with each other continuously. Not requiring an agent to know all the details of the subplans that are accomplished by its partners relaxes the communication requirements among agents. It also makes the coupling among their decision-making procedures flexible.

The formalization defines two meta-predicates for representing an agent's belief about its and its collaborators' abilities to perform actions in a SharedPlan; *CBA (can bring about)* for single agent, *CBAG (can bring about group)* for group capabilities. *CBA* is true for a basic level action (i.e., an action that is executable at will) and an agent, if it is believed that the agent can successfully perform the action. For a complex action (i.e., an action decomposable to constituent actions), *CBA* is true if it is believed that the agent has a recipe for performing the action and the agent is able perform all of its constituent actions. *CBAG* is defined similarly for group capabilities. *CBA* and *CBAG* are strong meta-predicates that are defined to be either true or false. They do not handle the uncertainty or doubt an agent may have about its or its collaborators' capabilities. As stated by Grosz and Kraus, the SharedPlan formalization would be better served by a probabilistic approach to the modeling of ability, but the authors did not identify a suitable computational model (Grosz and Kraus (1996) pg. 11).

The SharedPlan formalization uses modal operators to define the beliefs of an agent about the world and about the capabilities of agents. The standard modal operator *Bel* is used for representing beliefs. $Bel(G_q, prop)$ represents the fact that agent G_q believes that proposition $prop$ is true. Thus, like all logical formalizations, the SharedPlan formalization assumes that an agent has beliefs that are either true or false.

The way the SharedPlan formalization represents agents' beliefs about the capabilities of the participants of a collaborative activity and reasons about them is overly strong. Relying on logical representations alone does not allow to represent uncertainty. Not being able to do so results in a gap in formal models of teamwork for not handling the uncertain nature of the real-world properly. The following sections present new predicates, functions, operators and representations to fill this important gap in the SharedPlan formalization (and consequently in teamwork theories).

2.1.2 Examples of Collaborative Plans

Throughout this thesis, two examples are used to illustrate various aspects of teamwork and the representations and algorithms defined in this thesis. The first example is inspired by the dinner-making example presented in the original SharedPlan work (Grosz and Kraus, 1996). The second example, the opportunistic commerce example, involves a computer agent working collaboratively with a human driver to provide assistance during the daily commute from work to home. This example demonstrates the main characteristics of fast-paced domains in which computer agents and people collaborate.

Even though dinner-making is unlikely as a computer-human collaboration, this domain provides a good example to motivate various challenges and aspects of the formalization. Dinner-making is an intuitive and easy to understand domain for which almost every reader has knowledge and experience. The example shares the general characteristics of collaborative activities, which include decision-making requirements, limited resources, partial information and uncertainty of the world and needs for coordination, supportive behavior and communication. The domain is classical in that it is used extensively by prior work in plan recognition (Kautz, 1990; Litman and Allen, 1990) and teamwork (Grosz and Hunsberger, 2006; Grosz and Kraus, 1996).

The dinner-making example includes two agents, Alice and Bob, who are collaborating on cooking for a dinner party. It is decided that Alice will make an entree and Bob will make an appetizer. To successfully accomplish this group plan, each agent needs to form a plan individually for making an entree and an appetizer, respectively. These individual plans specify the recipes that will be used by Alice and Bob, the timing of their actions and the resources they will be using to accomplish the actions. Alice and Bob are not required to know all the details of the way the collaborative activity is being performed. Alice may not know which appetizer Bob is preparing, and Bob may have limited information about Alice's entree. To have a successful collaborative plan, Alice and Bob need to reason about each other's actions to discover potential conflicts, to notify each other about a possible failure or success, to inform and to help each other when needed. For instance, if Alice believes that Bob may be preparing either stuffed mushrooms or some type of salad, and she discovers that one of the guests is allergic to mushrooms, she needs to reason about the way the success of Bob's plan may be affected by this new information and then decide whether to communicate this information to Bob. Her entree recipe may be ruined if she takes too long to find Bob and inform him. She needs to reason not only about this cost, but also about the likelihood that Bob's appetizer recipe involves mushrooms.

The *opportunistic commerce* example is illustrative of situations in which a computer agent is actively involved in a collaborative activity in a fast-paced setting. Fast-paced settings are ones in which agents are distributed, conditions may be rapidly changing, actions occur at a fast pace, and decisions must be made within tightly constrained time frames. The example considers a collaboration between a driver and a computer agent to have the most effective plan for a trip from work to home. In this example, the driver is responsible for forming and performing commute plans which may include decisions about the timing of the trip and the route from work to home, and the computer agent is responsible for assisting the driver by creating opportunistic plans for daily errands. The design of the agent is based on the work by Kamar et al. (2008). The computer agent infers the active goals of the driver for purchasing products or services (e.g., fuel purchase), predicts the possible routes for her (Krumm and Horvitz, 2006) and performs an ongoing search over the possible feasible business locations that may satisfy her goals. The agent generates an opportunistic plan for the driver by adding the business with the highest expected value as a waypoint to the route. For instance, the driver may choose between two possible routes, Route A and Route B to follow for the evening commute. At the same time the computer agent may realize that the driver needs to buy gas and may search the area around Routes A and B for the best deal with the gas station offering the highest expected value. The expected value takes into consideration not only the advertised price of the product or service but also the additional costs in time and distance to access the waypoints.

The opportunistic commerce example demonstrates the essential elements of fast-paced domains. Both the computer agent and the driver need to plan and act under uncertainty. The traffic conditions may change unexpectedly, which may cause plan failure and the need to replan. The driver and the agent have partial information about the world and each other. As the driver is choosing a route to follow, she may not know about a recent accident in Route A that may affect the duration of her trip. Similarly, while the agent is generating an opportunistic errands plan for the driver, it may be uncertain about the route she has selected. The environment in which the driver and the agent act is open to interactions with other agents and with events not under their control. Given such uncertainty and partial information, the driver and the agent need to reason about each other and the world. Computer agents need representations and reasoning mechanisms to successfully accomplish collaborative activities under uncertainty.

For purposes of simplification, both the dinner-making and opportunistic commerce examples contain two agents. However, the formalization, representations and decision-making models presented in this thesis are not limited to the collaboration of a group of two. The representations are able to represent an agent's beliefs about the way a collaborative activity is being done by a group of many (more than 2). Similarly, agents can use the decision-making models to make decisions about a collaborative activity of many agents. An emergency rescue example in which multiple rescue teams (fire fighters, ambulances and security groups) work collaboratively to rescue as many survivors as possible (Keogh et al., 2009) could be handled by the techniques described in this thesis but would have been more complex to explain. The examples used in this thesis are able to represent many situations that arise in collaborative activities, whereas they take less to explain and motivate.

2.1.3 Basic Definitions

This section describes the key constructs of the SharedPlan formalization of collaborative activity and defines additional ones needed to accommodate uncertainty. The terminology is based on the original SharedPlan work, but is extended to express the probabilistic nature of real-life domains. In addition to the entities, operators, functions and predicates introduced by the SharedPlan formalism, it includes new predicates and functions that associates actions with costs, utilities and success probabilities to enable decision-theoretic reasoning.

The SharedPlan formalization models the mental state of agents that are collaborating to carry out actions needed to achieve a common goal. We will use $\mathcal{A}$ to represent the universe of agents, $GR \subset \mathcal{A}$ to represent a group of agents involved in a collaborative activity and forming a SharedPlan together.

In a dynamically changing and uncertain world, a SharedPlan may fail unexpectedly. Agents need to form the plan that offers the highest possible expected utility. In this thesis, it is assumed that every agent involved in a collaborative activity by group GR is committed to maximizing the same utility function, where the function is a linear combination of the utilities of individual agents in GR. This utility function may be the linear combination in which the same weight is given to the individual utilities of all agents in the group. For example, the utility function that Alice and Bob maximize for dinner-making may be the aggregate of Alice's and Bob's individual utilities of the collaborative activity. The utility function may also be designed to give more weight to the utilities of some agents in the group. The utility function that the driver and the agent maximize in the opportunistic commerce example may consider the driver's individual utility more important than the agent's. Maximizing a single utility function in a collaborative activity follows the joint reward function principle that is often used in the literature of decentralized decision-making under uncertainty (Bernstein et al., 2002) and leads agents involved in a collaborative plan to be truthful and not to exhibit malicious behavior.

Generalizing the decision-making models presented in this thesis to the cases in which agents in a collaborative activity maximize different utility functions introduces new challenges. An agent maximizing its own utility function may perform an action that is not beneficial for its partners or for the social good. A possible solution to this challenge is providing agents side payments that align their individual utilities with the social good. An example application of this idea to the case of self-interested agents (i.e. agents maximizing their individual utilities) is presented in Chapter 7.

Actions are abstract, complex entities that have a variety of properties including the type of the action, the agents assigned for the action, the time at which the action is performed and the objects used in doing the action. We use lower case Greek letters (e.g., α, β, γ) to represent actions and define a set of functions to access the various properties of an action: $time(\alpha)$, T_α for short, refers to the execution time of action α; $type(\alpha)$ represents the type of α; $agents(\alpha)$ refers to the agent or group of agents committed to accomplishing action α.

The recipe for an action is defined as the set of subactions and constraints such that performing those subactions under those constraints constitutes completing the action (Pollack, 1990). The function $recipe(\alpha)$ represents the set of recipes for accomplishing action α. A recipe for action α is defined as $R_\alpha = \{\Delta, \Upsilon\}$, where $\Delta = \{\beta_1, \ldots \beta_n\}$ is a set of subactions for accomplishing α, and $\Upsilon = \{\rho_1, \ldots, \rho_m\}$ is a set of constraints that need to be satisfied for successful completion of α. Agents may not have identical recipe libraries. Their recipe libraries may get updated over time. Agents forming a SharedPlan for accomplishing a complex action use their recipe libraries to derive recipes for the joint activity. To agree on how to perform an action, agents may integrate recipes from different agents' recipe libraries or discover new recipes.

A subsidiary action in a recipe, $\beta_i \in \Delta$, may be either a *basic-level* or *complex* action. A *basic-level* action does not decompose into lower-level actions, but is executable at will by only a single agent if the necessary conditions are satisfied. [1] The predicate $basic.level(\alpha)$ is true if α is a basic-level action. A *complex* action is decomposable to other complex actions or basic-level actions. A recipe for a complex action defines the subsidiary actions and conditions needed to perform the complex action. A complex action can be performed either by a group of agents or a single agent.

Decomposing a complex action recursively with the recipes given in a recipe library results in a multi-level action decomposition hierarchy. [2] An example of a multi-level action decomposition hierarchy is presented in Figure 2.1. In the example, the complex action α decomposes into subactions $\beta_1, \ldots, \beta_k$. The complex action β_1 decomposes into the complex action δ_{11} and basic-level actions $\delta_{12}, \ldots, \delta_{1m}$. The action decomposition

[1] The way basic-level actions are defined and used in this thesis differs from the work in the philosophy literature in that whether an action is basic-level or complex is determined by the designer of an application depending on the level of description needed for the application. An action may be defined as basic-level in one application and complex in another if the level of description needed by the applications is different.

[2] We use a standard assumption from planning that a complex action can be hierarchically decomposed into basic-level actions. Although complex actions such as iteration present challenges, such issues are beyond the scope of this thesis.

hierarchy cannot grow further as all the leaves are basic-level actions. γ_{111}, δ_{12}, β_2 represent basic-level actions at different levels of decomposition.

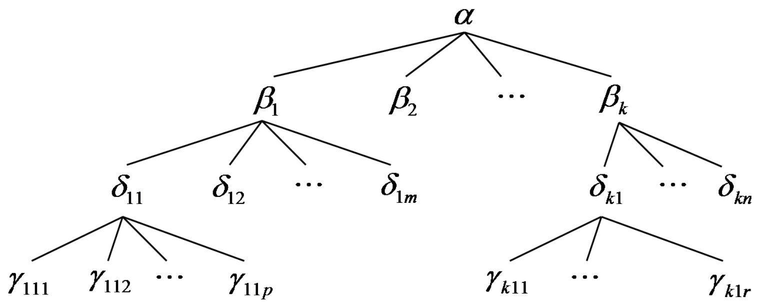


Figure 2.1: An example of an action decomposition hierarchy.

An action decomposition hierarchy is completely instantiated in the planning process if the decomposition of each action fully instantiates all of the action parameters (e.g., agents performing each action, timing of actions and constraints under which the complex action succeeds) and if all leaf nodes of the hierarchy are basic-level actions. When agents are developing their SharedPlans, an action hierarchy may not be completely instantiated. If an action decomposition hierarchy is not completely instantiated, agents involved in a SharedPlan for accomplishing a complex action have beliefs about the set of possible complete action hierarchies for that action.

The differences between complex and basic-level actions and the role of recipes in developing SharedPlans may be demonstrated with the dinner-making example. “Dinner-making” is a complex action that Alice and Bob are committed to doing together by performing a set of lower-level actions, for example making an ”appetizer” and an “entree”. Agents assigned to a complex action may have multiple recipes to choose from. If Bob commits to making an appetizer, he may choose “mushroom puffs”, a simplified recipe that decomposes the complex action “making appetizer” into two basic-level actions; “chopping mushrooms" and "baking". He may also choose a "lettuce salad" recipe that decomposes "making appetizer" into basic-level actions "chopping lettuce" and "adding dressing". The level of decomposition needed to decompose a complex action into basic-level actions may change with respect to the characteristics of the domain and the goal of the designer of the system. If desired, the action "chopping mushrooms" can be defined as a complex action that further decomposes into basic-level actions (e.g., actions for chopping each mushroom individually).

A SharedPlan is created and accomplished in some context. Context is relevant aspects of the state of the world in which agents act. Operators, functions and predicates defined over the actions of a SharedPlan need to refer to this context. For the purposes of this thesis, the context in which an action is performed by an agent or a group of agents is defined to include all of the information that agents need to make decisions about that action at a particular time. For example, Alice's context for *dinner-making* includes Alice's beliefs about the world (e.g., whether there are fresh tomatoes in the house) and Alice's beliefs about Bob's context for dinner-making (e.g., Alice's belief about Bob's belief about whether there are tomatoes). Similarly for the *opportunistic commerce* example, the computer agent's context includes its beliefs about the traffic conditions in the city and its belief about the route taken by the driver. The agent needs to use both pieces of information to generate the most appropriate errands plan for the driver.

Definition 2.1.1. $\mathcal{P} = \{prop_1, \ldots, prop_n\}$ is the set of propositions representing the complete state of the world. $\mathcal{P}_\alpha \subset \mathcal{P}$ is the set of all propositions that may relate to constructing a plan for action α. Importantly, α does not necessarily appear as an argument in the propositions in P_α.

$\Upsilon \subset \mathcal{P}_\alpha$ is the set of constraints that needs to be satisfied to accomplish action α with respect to a recipe for it. Agents need to consider the constraints for accomplishing α to construct a plan for it. These constraints may include resource constraints and temporal constraints. For instance, $Exists(r, t)$ is true if resource r is available for use at time t and $Completed(\beta, t)$ is true if action β is completed by time t.

Definition 2.1.2. For any agent G_q and action α, G_q's **context** for action α at time t, denoted as $C^\alpha_{G_q}$, [3] is

$$C^\alpha_{G_q} = \{Bel(G_q, prop_i), \ldots, Bel(G_q, prop_j)\}$$

the set of all of G_q's beliefs at time t about propositions in P_α, where $prop_i, \ldots, prop_j \in \mathcal{P}_\alpha$.

G_q may not have beliefs about all the propositions in P_α. For instance, G_q may not believe some propositions in P_α relate to doing action α. For any $prop_k \in P_\alpha$ that is not included in G_q's context for action α, G_q does not have beliefs about $prop_k$. For instance, if Alice does not even consider that some guests may be allergic to mushrooms, she may not have beliefs about the proposition $Allergic(guest, mushroom)$ although reasoning about this proposition may change the way Alice is making the entree. In that case, Alice's context for entree-making does not include her beliefs about this proposition. Thus, for any $prop_k \in P_\alpha$ that $Bel(G_q, prop_k) \notin C^\alpha_{G_q}$, it is the case that $\neg Bel(G_q, prop_k)$.

$C^\alpha_{G_q}$ may include not only agent G_q's beliefs about the world, but also G_q's beliefs about other agents' beliefs about the world and other agents. $C^\alpha_{G_q,G_r} \subset C^\alpha_{G_q}$ denotes all of G_q's

[3] Unless not clear, the arguments G_q or α of context notation $C^\alpha_{G_q}$ may be omitted to simplify the notation.

beliefs about G_r's beliefs about propositions in $\mathcal{P}_\alpha$.

$$C^{\alpha}_{G_q,G_r} = \{Bel(G_q, Bel(G_r, prop_l)), \ldots, Bel(G_q, Bel(G_r, prop_m))\}$$

where $prop_l, \ldots, prop_m \in \mathcal{P}_\alpha$.

The function $d\text{-}Bel(C^{\alpha}_{G_q}, prop_i)$ maps agent G_q's context for action α and proposition $prop_i$ to a degree of belief. [4] $d\text{-}Bel(C^{\alpha}_{G_q}, prop_i) \in [0, 1]$ represents the likelihood that $prop_i$ is true in $C^{\alpha}_{G_q}$, where $prop_i \in \mathcal{P}_\alpha$ and $Bel(G_q, prop_i) \in C^{\alpha}_{G_q}$. In other words, $d\text{-}Bel(C^{\alpha}_{G_q}, prop_i)$ represents G_q's confidence that $prop_i$ holds. For any $prop_i \in \mathcal{P}_\alpha$ such that $Bel(G_q, prop_i) \notin C^{\alpha}_{G_q}$, the value of the function $d\text{-}Bel(C^{\alpha}_{G_q}, prop_i)$ is assigned to a predetermined constant in $[0, 1]$ interval.

For example, the set of propositions related to a particular action of "dinner-making" may include a variety of facts about the resources available for cooking (e.g., $Exists(tomatoes, 5pm)$), about the physical environment (e.g., $Market\text{-}closed(7pm)$), about the progress of agents (e.g., $Completed(Appetizer)$), and about decisions made by others. Alice's context for this dinner-making action would include her beliefs about these propositions. If Alice believes that the likelihood of Bob selecting the mushroom-puffs recipe is 70% and his selecting the lettuce salad recipe is 20%, the $d\text{-}Bel$ function defined over Alice's context (C_{Alice}) would represent these likelihoods as follows:

$$d\text{-}Bel(C_{Alice}, Selected\text{-}Recipe(Bob, \textit{mushroom puffs}, \textit{appetizer}, C_{Bob})) = 0.7,$$

$$d\text{-}Bel(C_{Alice}, Selected\text{-}Recipe(Bob, \textit{lettuce salad}, \textit{appetizer}, C_{Bob})) = 0.3$$

where $Selected\text{-}Recipe(Bob, \textit{mushroom puffs}, \textit{appetizer}, C_{Bob}) \in \mathcal{P}_{appetizer}$, and $Selected\text{-}Recipe(Bob, \textit{lettuce salad}, \textit{appetizer}, C_{Bob}) \in \mathcal{P}_{appetizer}$.

[4] The definition of *degree of belief* is based on but slightly modified from the way it is defined by Davis (1990).

The predicate $Selected\text{-}Recipe(G_q, R_\alpha, \alpha, C^\alpha_{G_q})$ is true if recipe R_α is the recipe selected by agent G_q in context $C^\alpha_{G_q}$ for accomplishing action α.

The definition of context given in this section is only one of the possible ways for defining context formally. The literature on Decentralized Markov Decision Processes combines all the information about the world that are relevant to the decentralized planning into a state representation (Bernstein et al., 2002). Interactive POMDPs extend this regular state representation to interactive states that include agents' possibly infinitely nested beliefs about other agents and other agents' beliefs as well as beliefs about the physical state of the world (Gmytrasiewicz and Doshi, 2004). Formal models of teamwork define context as a collection of propositions that are believed to hold, semantic rules specifying what these propositions mean, how they relate to agents' intentions and goals, and how they get updated in time as agents act in the world (Cohen and Levesque, 1990; Grosz and Kraus, 1996).

The way context is defined formally in this thesis combines essential elements of both of these approaches. The context of an agent at a given time is defined over the propositions that may hold in the world at the given time, accompanied with their corresponding probabilities representing the confidence of the agent about the validity of these propositions. It combines the logical and probabilistic features needed for reasoning about collaborative plans. In particular, it incorporates essential features of the possible world semantics context definition used in formal logical teamwork models with a representation of probabilistic distribution of agent beliefs needed to reason in environments with uncertainty.

Table 2.3 lists several new predicates and functions used in this thesis. The predicate $Context$ is true if agent G_q believes at time T that action α is being done in context C^α.

Table 2.3: Summary of predicates and functions.

$Context(C^{\alpha}, G_q, \alpha, T)$	the believed context of action α
$cba.basic(G_q, \beta, C^{\beta})$	probability that G_q can bring about action β
$cost.basic(G_q, G_r, \beta, C^{\beta})$	cost of action β
$V(G_q, \alpha, C^{\alpha})$	value of action α

(T may be different than T_{α}, the execution time of α.)

In an uncertain, partially observable and dynamically changing world, an agent may fail to execute an action that it is in general capable of doing because the world is uncertain or the agent's model of its own or another agent's ability to perform an action may be incorrect. For example, Alice's plan for entree making may fail if the oven breaks down unexpectedly. She may fail to perform her entree plan also because she lacks a key ingredient she thought she had. The deterministic functions CBA and $CBAG$ of the SharedPlan formalization are not able to represent these uncertainties. Thus, this thesis uses a different function to represent the probability of an agent performing a basic level action successfully. The function $cba.basic$ represents the probability that agent G_q can bring about (i.e., successfully complete) the basic-level action β in context C^{β}.

When they perform basic level actions, agents incur costs from such expenses as resources being used or time and energy being consumed. The function $cost.basic$ denotes the cost of performing a basic-level action. In particular, $cost.basic(G_q, G_r, \beta, C^{\beta})$ represents the cost incurred by agent G_q when basic-level action β is executed by agent G_r in context C^{β}. (G_q and G_r may refer to the same agent. C^{β} is the context of the agent reasoning about this cost, which may be either agent G_q or G_r.)

The agents participating in a SharedPlan have incentives for the success of their plans because they benefit (perhaps in different ways) from successful completion of their planned activity. The function $V(G_q, \alpha, C^{\alpha})$ represents the (non-negative) utility for agent G_q of the successful completion of action α in context C^{α}. If an action is basic level, the utility of performing the action equals the corresponding value of the V function. The utility for carrying out a complex action with respect to a particular recipe for the action is the sum of the value of the function V for that action and the sum of the utilities for performing each of its subactions because the utility of a complex action may differ from "the sum of its parts". For example, if the recipe chosen by Alice and Bob for making the entree is composed of grilling the chicken and cooking a sauce for the chicken, successfully completing the entree recipe provides higher utility to the agents than the sum of the utilities for having the sauce and the grilled chicken separately. Consequently, the value of the function V for the action "entree making" is larger than zero. On the other hand, if Alice and Bob fail to make dinner, because Alice does not make the entree, but Bob prepares the appetizer, they still have utility for having something to eat. In this case, the value of the function V for the action "appetizer making" is larger than zero; the value of the dinner-making plan is higher than if Bob also failed to prepare his dish. In contrast, the value of a constituent action may be zero, if accomplishing that action alone does not provide any utility. For example, if the recipe selected for making the appetizer is a mushroom puffs recipe which includes chopping onions as a constituent action, the utility of agents for chopping onions is zero if the plan for mushroom puffs fails. Thus, the value of the function V is zero for the constituent action "chopping onions".

This thesis leaves the specification of the values for $cba.basic$, $cost.basic$ and V functions for each basic-level action and agent couple to the domain modeler. It is assumed that the domain is modeled such that all agents know these values or have estimates of them in their mental models. The functions $cba.basic$ and $cost.basic$ are defined only for basic-level actions. Recursive algorithms for computing the success probability, the cost and the utility of complex actions are presented in Section 2.3.

2.2 Probabilistic Recipe Trees

Key features of the SharedPlan formalism are that agents may have incomplete information about the way to accomplish a group activity, and thus their plans may be partial and they may know little about how the actions for which they are not responsible are being done. These features interact in ways that present challenges for multi-agent planning. For example, even though Alice is committed to Bob's success as a part of their plan to cook dinner together, she may not know which appetizer he is making. She needs to make decisions about helping and supporting Bob without knowing his recipe. Agents cannot reason about the benefit to the group from engaging in helpful behavior when they have no information about the recipes that other group members are considering. To bridge this gap, this section provides a way to represent agents' beliefs about the recipes that may be selected by other group members to complete a constituent action of their collaborative activity. To do so, it defines a novel representation, Probabilistic Recipe Trees (PRTs).

2.2.1 Formal Definition

Definition 2.2.1. A Probabilistic Recipe Tree (PRT) is a structured AND-OR tree representation that is either

- a single node representing a basic-level action,
- an AND node representing a recipe for a complex action and one or more subtrees, each of which has a root node representing a subaction (constituent action) of the complex action,
- an OR node representing a complex action and one or more subtrees, each of which has a root representing a recipe for accomplishing the complex action. Each branch leaving an OR node is associated with a probability such that the sum of the probabilities of all branches leaving a single OR node is 1.

The Probabilistic Recipe Tree (PRT) for action α defines a complete probability distribution over the possible action decomposition hierarchies and recipes for accomplishing α.

Figure 2.2 illustrates a PRT. Each node in a PRT represents either an action (e.g., β_1) and associated with the properties of the action (e.g., type of action, agents committed to the action, time of action), or it represents a recipe for accomplishing an action (e.g., recipe $R^1_{\beta_1}$ for action β_1) and associated with the properties of that recipe (e.g., constraints of the recipe). Leaf nodes of a PRT represent basic-level actions (e.g., γ_1). Intermediate nodes may be either AND or OR nodes. Each child of an AND node represents a constituent subaction for accomplishing the complex action represented by the AND node. Each child of an OR node represents a possible choice of recipe for the complex action represented

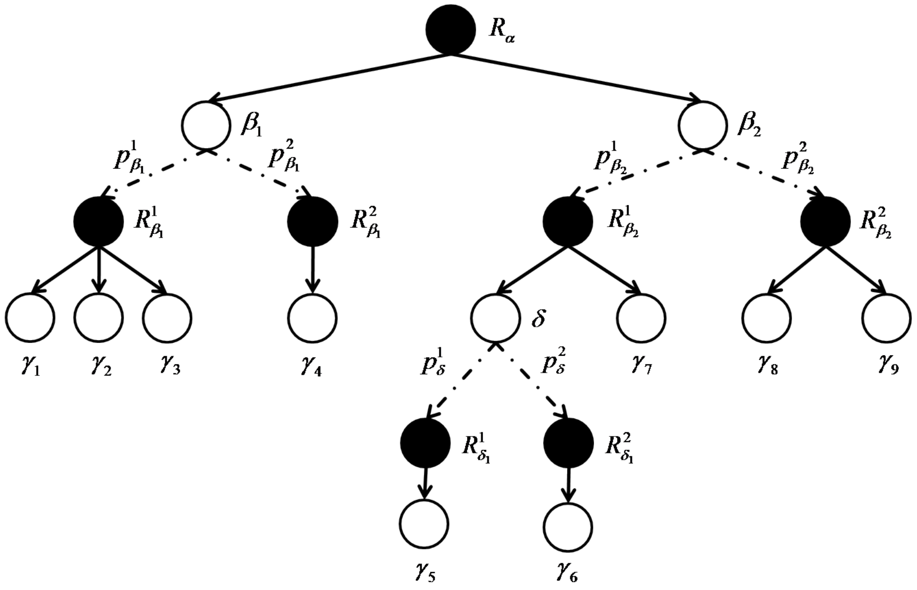


Figure 2.2: A probabilistic recipe tree. Black nodes are AND nodes. White nodes are OR nodes.

by the OR node, where the choice is non-deterministic. Each branch from an OR node to a child node has an associated probability representing the likelihood that the recipe represented by the child node is chosen as the recipe for accomplishing the action represented by the OR node. In Figure 2.2, probability $p^1_{\beta_1}$ represents the probability that recipe $R^1_{\beta_1}$ is selected for action β_1.

Various functions and predicates defined over Probabilistic Recipe Trees are presented in Table 2.4. If node θ refers to a recipe, function $\Upsilon(\theta)$ represents the set of constraints that needs to hold for the successful completion of the recipe. Otherwise, $\Upsilon(\theta)$ is an empty set.

The probability function Pr represents the branching probability of a child PRT from a parent node θ. It is defined as follows for all $PRT_\beta \in children(\theta)$:

$$Pr(PRT_\beta) = \begin{cases} \in [0,1] & \text{if } OR(\theta) = 1 \\ = 1 & \text{otherwise} \end{cases} \tag{2.1}$$

Table 2.4: Summary of predicates and functions defined over PRTs.

$action(\theta)$	the action represented by node θ
$agents(\theta)$	the agent or the group of agents committed to doing $action(\theta)$
$OR(\theta)$	is true if node θ is an OR node
$AND(\theta)$	is true if θ is an AND node
$children(\theta)$	the set of PRTs that are children of θ
$\Upsilon(\theta)$	the set of constraints associated with node θ
$Pr(PRT_\beta)$	branching probability of PRT_β from its parent node
$top(PRT_\alpha)$	root node of PRT_α

PRTs are an efficient way to represent agents' beliefs about the possible ways they or other agents may perform a complex action. They are exponentially more compact than an exhaustive representation over a set of possible action decomposition hierarchies for a complex action. [5] That this is the case may be demonstrated by considering the space of recipes with up to n potential recipes for each action, each recipe having up to m constituent actions, and d representing the number of levels of decomposition needed to fully transform the top-level action into basic-level actions. The size of a single action decomposition hierarchy in which a particular recipe is selected for performing each complex action is $O(m^d)$. Because there are n possible recipes for each action, the number of possible action decomposition hierarchies is $O(n^{m^d})$, and a distribution over recipes will have to assign

[5] This is true even though constituent actions of a collaborative activity may not be independent. For example, there may be temporal constraints among them. Such constraints can be represented with an edge between two nodes representing these actions, and thus without increasing the number of nodes of a PRT.

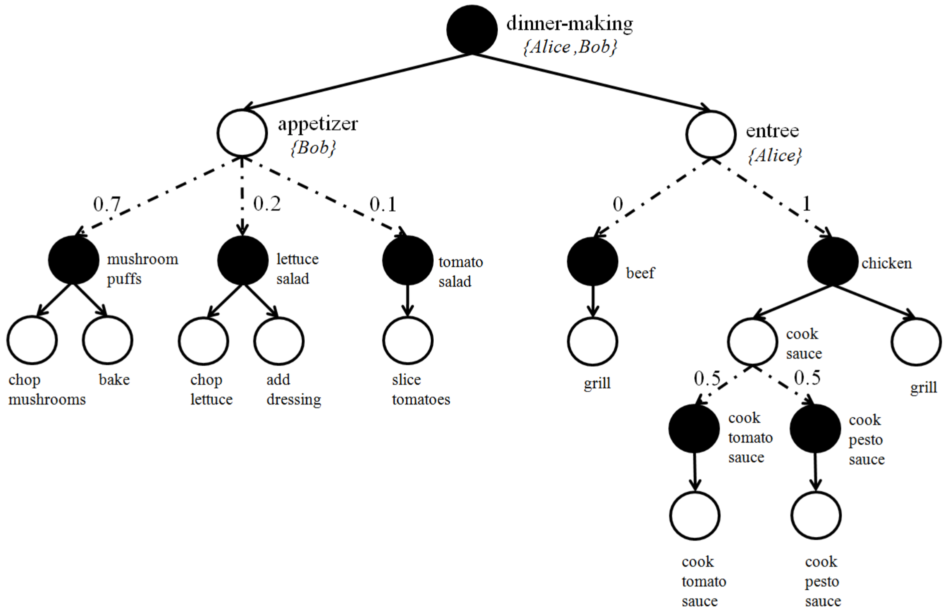


Figure 2.3: A Probabilistic Recipe Tree for the dinner-making example. Black nodes are AND nodes. White nodes are OR nodes.

a separate probability for each of them. Thus, the size of an exhaustive representation is exponential in both the number of constituent actions and the level of decomposition. In contrast, the size of the PRT is limited to $O((nm)^d)$, which is exponentially smaller.

The PRT representation is not only efficient in terms of its size, but also its modular structure provides a computationally efficient way for updating PRTs to reflect changes in agents' beliefs. The operations defined on PRTs for updating them efficiently are presented in Section 2.2.3.

2.2.2 Examples of PRTs

Figure 2.3 presents the PRT for the *dinner-making* example, which represents Alice's beliefs about the way dinner-making plan is being accomplished in collaboration with Bob.

This PRT represents her belief that the dinner-making consists of making an appetizer and an entree. Consequently the top node of the tree (dinner-making), an AND node, branches into constituent actions of making an appetizer and making an entree. The children of the appetizer node represent the possible recipes for making an appetizer. In this example, these recipes are making mushroom puffs, lettuce salad, or tomato salad. According to Alice's beliefs as represented in this example, the likelihood of selecting mushroom puffs as the appetizer is 0.7.

The PRT representation allows representing agents' possibly different beliefs about the way a collaborative activity is being accomplished. Bob's beliefs about the way dinner-making plan is being done may differ from Alice's beliefs about the plan. For instance, he may be sure about the recipe he is following to do the appetizer and he may believe that Alice is doing the beef with 100% chance. In addition to the differences in the likelihoods of recipes, Alice's and Bob's PRTs may not include the same set of actions or recipes. For example, Bob may only consider the tomato salad recipe as a possible way for making the appetizer. Chapter 3 addresses the possible difference in agents' beliefs and presents helpful behavior models that can reason about this difference to determine whether and how agents can support each other.

The PRT presented in Figure 2.3 represents a probability distribution over 9 possible action decomposition hierarchies for dinner-making. A choice among recipes for each of the OR nodes constitutes one deterministic recipe for achieving this action. For example, one possible way to make dinner is to make mushroom puffs and chicken with tomato sauce. The probability of choosing this decomposition hierarchy is 0.35.

Another sample PRT, this time in the *opportunistic commerce* domain, is presented in

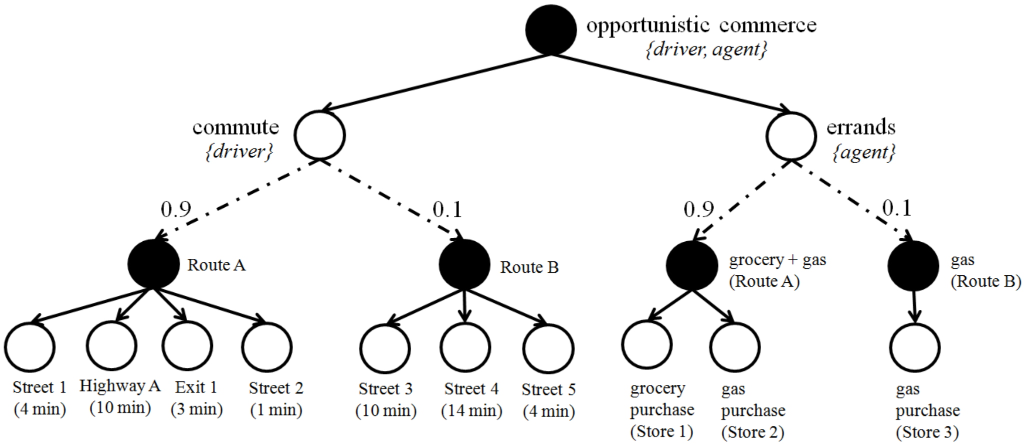


Figure 2.4: A Probabilistic Recipe Tree for the mobile opportunistic commerce example. Black nodes are AND nodes. White nodes are OR nodes.

Figure 2.4. This PRT represents the computer agent's beliefs about the way the complex action *opportunistic commerce* is being accomplished. The AND node representing the opportunistic commerce action branches into the constituent actions of commuting and doing errands. The plan for errands includes decisions about services or products that can be purchased on the way while driving. The commute node stochastically branches into possible routes the driver can take to get home. In this example, the agent expects the driver to take Route A with 90% chance. The AND node representing the Route A recipe branches into leaf nodes representing basic-level actions of the recipe.

This opportunistic commerce PRT represents a probability distribution over four possible action decomposition hierarchies. One possible deterministic action decomposition consists of taking Route A for driving and stopping at Stores 1 and 2 for grocery and gas purchases. The likelihood of this decomposition is 0.81. In contrast, the choice of a decomposition consisting of taking Route B and having a stop at Store 3 for gas purchases is highly unlikely; it has only 1% chance.

2.2.3 Operations on PRTs

In a partially observable and dynamically changing environment, collaborative plans and agents' beliefs about these plans are not static, but changing over time. Probabilistic Recipe Trees may need to get updated to reflect such changes in agents' beliefs. The modular structure of the PRTs enables easy and efficient updating of the parts of a PRT. A plan constructed for a subaction may get updated while plans selected for other subactions remain the same. The modularity of PRTs allows updating a PRT partially without needing to consider the trees for the subtasks that are unchanged. This efficiency is not possible if uncertainty is represented by a probability distribution over complete SharedPlans.

To enable efficient updating of a PRT using this modular structure, we define the following operators on PRTs:

- **Addition:** The operator $PRT_\alpha \bigcup PRT_\beta$ adds PRT_β as a child of PRT_α; as illustrated in Figure 2.5. If α is an OR node, the probability distribution over the branches leaving the OR node is normalized after integrating PRT_β into PRT_α.

Applying the addition operator on a PRT represents a change in an agent's beliefs about the way a complex action is being accomplished. Adding a PRT as a child to an AND node represents a change in an agent's belief that a new constituent action needs to be performed in order to accomplish the complex action represented by the AND node. Addition of a PRT as a child to an OR node represents an agent discovering a new way to perform the complex action represented by the OR node. The branching probability of the added PRT

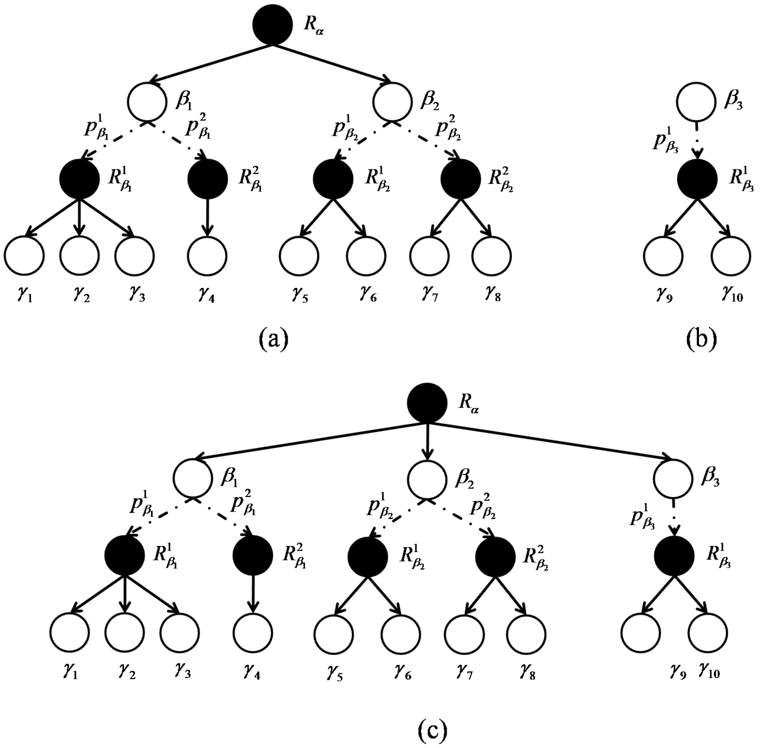


Figure 2.5: (a) PRT_α, a probabilistic recipe tree for action α. (b) PRT_{β_3}, a probabilistic recipe tree for action β_3. (c) $PRT_\alpha \cup PRT_{\beta_3}$, the updated probabilistic recipe tree with a new action added.

is associated with the agent's belief about the likelihood of this new recipe being chosen for performing the complex action.

For the dinner-making example, adding a PRT for "desert-making" as a child of the AND node for "dinner-making" would represent a change in Alice's beliefs from dinner being a combination of an appetizer and an entree to being a combination of an appetizer, an entree and a desert. The addition of a PRT for "fish fillet" as a child to the OR node for "entree-making" would represent Alice believing that fish fillet is a possible entree. With this addition, Alice's beliefs about dinner-making as represented by the PRT would include her beliefs about making "fish fillet" and the likelihood of taking that action.

- **Subtraction:** The operator $PRT_\alpha \setminus PRT_\beta$ removes the sub-tree PRT_β from PRT_α, if such a sub-tree exists, as illustrated in Figure 2.6. If α is an OR node, then the probability distribution over the branches leaving the OR node is normalized after this detach.

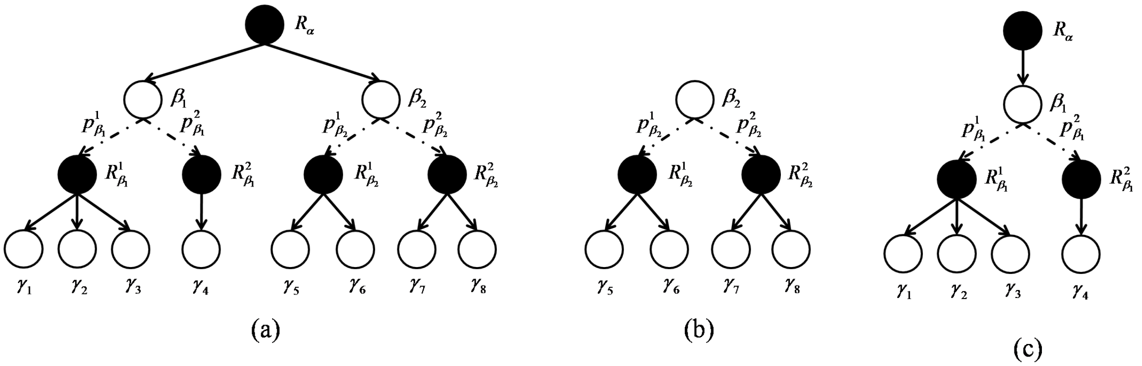


Figure 2.6: (a) PRT_α, a probabilistic recipe tree for action α. (b) PRT_{β_2}, a probabilistic recipe tree for action β_2. (c) $PRT_\alpha \setminus PRT_{\beta_2}$, the updated probabilistic recipe tree with action β_2 removed.

Removing one of the children of an AND node represents a change in an agent's beliefs that the constituent action associated with the detached child node is no longer needed to accomplish the complex action represented by the AND node. Subtracting a PRT that is one of the children of an OR node is the same as setting the branching probability of that PRT to 0; it represents a change in an agent's beliefs that the recipe associated with the detached child is not a valid recipe for the complex action represented by the OR node.

Subtracting the PRT for "entree-making" from the "dinner-making" node would represent a change in Alice's beliefs to making dinner consisting only of making an appetizer.

After this detachment, Alice would believe that the collaborative plan will be complete once the appetizer is done. Subtracting the PRT for “beef-making” from the “entree-making” OR node would eliminate one of the two possible recipes for the entree. The resulting PRT would represent Alice’s belief that the only way to make an entree is “chicken-making”.

- **Replacement:** The operator $PRT_\alpha \bigotimes PRT'_\beta$ removes PRT_β (the original PRT for β in PRT_α) from PRT_α, and adds PRT'_β to the parent node of PRT_β as illustrated in Figure 2.7. If this parent node is an OR node, the probability distribution over the branches leaving the node is normalized after the replacement. The implementation of the PRT replacement operation is different from simply doing a subtraction followed by addition in that the subtraction is done at the node at which the new subtree will be added. Operationally, this means that the position for the addition is identified without the need for search.

Replacing a part of a PRT with a new PRT represents updating an agent’s beliefs about the way a complex action is performed in the collaborative activity. This update may include a change in beliefs about the set of constituent actions needed to be done for accomplishing a complex action. It may also include a change in beliefs about the likelihoods of different recipes being chosen to perform a complex action.

Alice’s updated beliefs about making “lettuce salad” would be represented by replacing the corresponding PRT with a new PRT with “open package” and “add olive oil” actions. Alice being sure about the recipe being used for making appetizer would be represented by replacing the PRT for the appetizer with one in which the branching probability of “mushroom-puffs” is 1.

For simplicity of presentation, the definitions of PRT operations given in this section

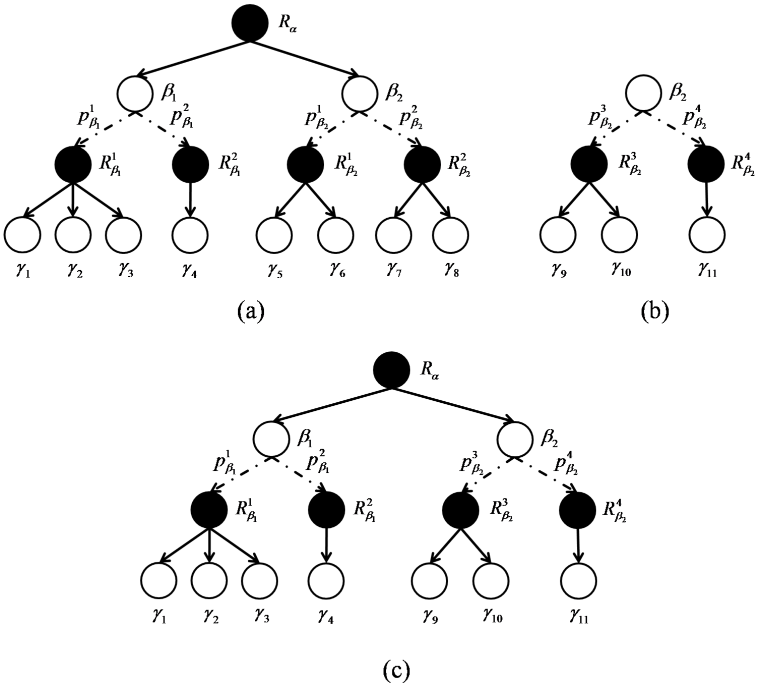


Figure 2.7: (a) PRT_α, a probabilistic recipe tree for action α. (b) $PRT^{'}_{\beta_2}$, an updated probabilistic recipe tree for action β_2. (c) $PRT_\alpha \otimes PRT^{'}_{\beta_2}$, the updated probabilistic recipe tree with a recipe replaced.

assume that the constituent actions of a collaborative activity are independent. These independence assumptions provide for a modular PRT structure enabling efficient updates. The formal definition of PRTs makes no such assumptions, but, in fact, provides for representing the dependencies that may arise. Section 2.2.4 presents an example of representing dependencies among constituent actions on PRTs. The algorithms presented in Section 2.3 for evaluating PRTs consider these dependencies and reason about the way updating a part of a PRT may affect the success of other constituent actions. Reasoning about such dependencies increases the complexity of applying operations on PRTs, since updating one part of a PRT may affect the way other constituent actions are being done.

2.2.4 Representing Constraints

The PRT representation may be extended to incorporate constraints among the recipes and actions chosen for a collaborative plan. This section briefly demonstrates the way PRTs may be used to represent two types of constraints; resource and temporal constraints.

As defined in Section 2.1.3, a recipe has a resource constraint if a particular resource is required at a particular time for the successful accomplishment of the recipe. A resource constraint exists between two nodes of a PRT representing two different recipes, if (1) the two recipes are not alternative ways of doing the same action, (2) they may belong to the same action decomposition hierarchy (i.e., nodes representing the two recipes do not have two different ancestors that are children of the same OR node) and (3) they have an overlapping resource constraint (e.g., they require the same resource at the same time).

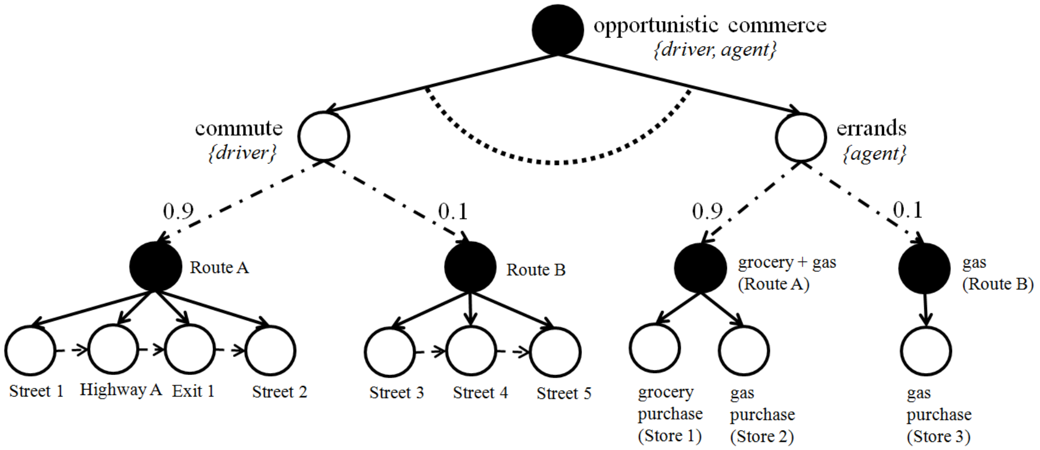


Figure 2.8: A probabilistic recipe tree for the mobile opportunistic commerce domain incorporating resource and temporal constraints.

Figure 2.8 illustrates resource and temporal constraints in the opportunistic commerce example. Both the recipe for commuting and the recipe for opportunistic commerce require the driver as a resource at the same time. The driver is required to be physically inside the car for driving on the road and required to be in a store for purchasing gas and grocery on the way. The dashed line in Figure 2.8 between nodes “commute” and “errands” represents this resource constraint.

A temporal constraint exists between two nodes of a PRT if the action represented at one of the nodes requires the successful completion of the action represented by the other node at a particular time. In Figure 2.8, the dashed lines among the leaves of the PRTs for routes A and B represent temporal constraints among these basic-level actions. For example, entering the Highway can only be done after completing the route through Street 1.

PRTs represent temporal and resource constraints in a way that can easily be used by planning algorithms to discover dependency relationships among constituent actions of a collaborative activity. Because the focus of this thesis is not on planning algorithms, the challenge of dynamically updating agents’ beliefs and consequently PRTs efficiently under these constraints is left to future work.

Representing these constrains within PRTs helps to identify some of the problems to be addressed by planning algorithms. If the actions represented by two subtrees of a PRT are not independent of each other, building and updating one of the subtrees requires reasoning about the other subtree. In the mobile opportunistic commerce example, the planning algorithm for constructing the commute plan needs to consider the plan chosen for errands. To ensure consistency among these two constituent actions, the algorithm needs to reason about the way the plan chosen to do the errands would be affected by the plan chosen for commuting. Similarly, generating a successful plan for a complex action requires reasoning about the temporal constraints among its constituent actions. In a successful driving plan generated for the opportunistic commerce example, the timings of the commute actions (e.g., Street 1, highway) depend on the timings of the preceding actions.

2.3 Evaluating Probabilistic Recipe Trees

To enable decision-theoretic reasoning on formal teamwork models, it is necessary to have methods for evaluating the expected utility of a collaborative activity. Being able to evaluate the utility of a collaborative activity enables to reason about the effect of performing a communication action or a helpful action on the activity. PRTs offer a structured representation that can be used to compute the expected utility of collaborative activities based on agents' beliefs. By using PRTs, agents are able to estimate the expected utility of an activity without knowing all the details about the constituent actions other participants of the activity are performing.

This section presents algorithms for analyzing the success likelihood, cost and expected utility of collaborative activities by propagating $cost.basic$, $cba.basic$ and V values from basic-level actions to the top of the tree.

Table 2.5: Summary of notations for Sections 2.3.1, 2.3.2 and 2.3.3.

T_α	time of execution for action α
T_β	time of execution for action β
$C^\alpha \in \mathcal{C}$	$Context(C^\alpha, G_q, \alpha, T_\alpha)$
$C^\beta \in \mathcal{C}$	$Context(C^\beta, G_q, \beta, T_\beta)$

2.3.1 Success Probability of a PRT

The function $p\text{-}CBA(PRT_\alpha, C^\alpha)$ denotes the probability of successfully performing action α in context C^α given the recipes represented in PRT_α. Algorithm 1 presents a method for calculating $p\text{-}CBA$. The algorithm makes the following assumptions: (1) to successfully perform a recipe for an action, the constraints of the recipe must hold in the context in which the action is being performed; (2) to successfully execute a complex action, all the constituent actions of the complex actions must be successfully executed.

Algorithm 1 computes the probability that a PRT is successful by propagating $cba.basic$ values of basic-level actions from the leaf nodes to the top. For a leaf node representing a basic action, the algorithm returns a value that equals the function $cba.basic$ applied to the leaf. For an internal node, cba_α represents the probability of satisfying the constraints required by the recipe referred by the node. cba_β represents the success probabilities propagated up from the children. For AND nodes, cba_β is the product of the probabilities that the children nodes will succeed. For OR nodes, cba_β is an average of the likelihood that the child nodes will succeed, weighted by the probability assigned to each of the children. Finally, the algorithm returns the success probability as the product of the probabilities the constraints will be satisfied and that the children nodes will succeed.

2.3.2 Cost of a PRT

The function $Cost(G_i, PRT_\alpha, C^\alpha)$ denotes the expected cost to agent G_i for the group carrying out the recipes represented in PRT_α in context C^α. Algorithm 2 presents a method for calculating the $Cost$ function by a top-down traversal of the tree. For leaf nodes, the algorithm returns the value of the function $cost.basic$ applied to the leaf. For AND nodes

Algorithm 1 : $p\text{-}CBA(PRT_\alpha$**,** $C^\alpha)$, represents the success likelihood of PRT_α.

```
$\theta$ = $top(PRT_\alpha)$
**if** $(basic.level(action(\theta)))$ **then**
    return $cba.basic(agents(\theta), action(\theta), C^\alpha)$
**else**
    $cba_\alpha$ = 1.0
    **for** $\rho_i \in \Upsilon(\theta)$ **do**
        $cba_\alpha = cba_\alpha \times d\text{-}Bel(C^\alpha, \rho_i)$
    **end for**
    $cba_\beta$ = 1.0
    **if** $AND(\theta)$ **then**
        **for** $PRT_\beta \in$ children($\theta$) **do**
            $cba_\beta = cba_\beta \times p\text{-}CBA(PRT_\beta, C^\beta)$
        **end for**
    **else if** $OR(\theta)$ **then**
        $cba_\beta$ = 0.0
        **for** $PRT_\beta \in$ children($\theta$) **do**
            $cba_\beta = cba_\beta + (Pr(PRT_\beta) \times p\text{-}CBA(PRT_\beta, C^\beta))$
        **end for**
    **end if**
    return $cba_\alpha \times cba_\beta$
**end if**
```

the value returned by the algorithm is a summation of the cost of its children nodes. For OR nodes it is an average of the costs for the children nodes, weighted by the probability assigned to each child.

Algorithm 2 : Cost(G_i, PRT_α, C^α), representing the cost of agent G_i from the execution of PRT_α.

```
$\theta$ = top($PRT_\alpha$)
**if** (basic.level(action($\theta$))) **then**
    return cost.basic($G_i$, agents($\theta$), task($\theta$), $C^\alpha$)
**else if** AND($\theta$) **then**
    cost = 0.0
    **for** $PRT_\beta \in$ children($\theta$) **do**
        cost = cost + Cost($G_i$, $PRT_\beta$, $C^\beta$)
    **end for**
    return cost
**else if** OR($\theta$) **then**
    cost = 0.0
    **for** $PRT_\beta \in$ children($\theta$) **do**
        cost = cost + ($Pr(PRT_\beta) \times Cost(G_i$, $PRT_\beta$, $C^\beta$))
    **end for**
    return cost
**end if**
```

2.3.3 Utility of a PRT

The function $Utility(G_i, PRT_\alpha, C^\alpha)$ denotes the expected utility to agent G_i for carrying out the recipes represented in PRT_α in context C^α. Algorithm 3 calculates the $Utility$ function. The utility function is computed by traversing the PRT in a top-down fashion. The algorithm combines the expected utility of the parent node with the expected utilities of its children. The expected utility of a node is the value of the action that the node represents multiplied by the success probability (p-CBA) of the node. If the node is an OR node, the expected utility of each child node is weighed by its branching probability. Computation of the $Utility$ function requires traversing the entire PRT, because a recipe that is selected for one of the subactions in a PRT may affect the utility of a recipe for another subaction in the tree. For example, if Bob decides to change the main course to include tomatoes, this may affect the resources available for making the appetizer, and thus it may affect Alice's decision about which appetizer to do. If she were planning to do tomato soup, her plan may fail as a result of not having enough tomatoes. The calculations for evaluating the expected utility of dinner-making needs to consider the dependencies among the constituent actions, thus needs to consider the entire plan.

To compute the value of a PRT, its cost and utility are computed. As defined in Equation 2.2, the benefit of a PRT for a group is the difference between the expected utility for the group for carrying out α and the expected cost, given the recipes represented in PRT_α.

$$Eval(GR, PRT_\alpha, C^\alpha) = \Sigma_{G_i \in GR}(Utility(G_i, PRT_\alpha, C^\alpha) - Cost(G_i, PRT_\alpha, C^\alpha)) \tag{2.2}$$

Algorithm 3 : $Utility(G_i, PRT_\alpha$**,** $C^\alpha)$, estimates the utility G_i obtains when PRT_α is executed in context C^α.

```
$utility = p\text{-}CBA(PRT_\alpha, C^\alpha) \times V(G_i, \alpha, C^\alpha)$
$\theta = top(PRT_\alpha)$
if $AND(\theta)$ then
  for $PRT_\beta \in children(\theta)$ do
    $utility = utility + Utility(G_i, PRT_\beta, C^\beta)$
  end for
else if $OR(\theta)$ then
  for $PRT_\beta \in children(\theta)$ do
    $utility = utility + (Pr(PRT_\beta) \times Utility(G_i, PRT_\beta, C^\beta))$
  end for
end if
return $utility$
```

2.3.4 Computing Utilities, Probabilities and Costs

To use PRTs in practice, agent designers need to determine the success probabilities and costs for all basic-level actions as well as values for all actions in the domain of interest. A designer needs algorithms and models to predict these values based on the dynamic context. This section provides an example in the domain of opportunistic commerce of how such values can be determined.

To construct its context about the opportunistic commerce action, the agent can draw on traffic prediction systems, information coming from satellites for weather prediction, the driver's calendar data and predictive models for the driver for estimating her time cost and value for accomplishing parts of the opportunistic commerce plan successfully. These information sources are used to predict the likelihood that propositions relevant to opportunistic commerce action hold. For example, information coming from the satellites may help to predict the likelihood of rain during the commute.

The agent uses its information sources to infer the costs of doing basic-level actions. For example, it accesses the traffic prediction services and the driver's time-cost function generated based on her historical data to predict the cost for driving. To compute the expected utility of a procurement plan, the agent trades-off the driver's value for a particular purchase (e.g., gas) with the additional cost for driving to the store (Kamar et al., 2008).

The branching probabilities from the "driving" OR node, representing the likelihood of choosing a particular route, can be retrieved from a system for predicting where the user is driving to based on data collected from prior trips (Krumm and Horvitz, 2006).

Given these costs, utilities and probabilities, evaluating the PRT involves combining the success likelihood of different driving and opportunistic purchase plans with the driver's expected value for these plans. The expected utility to the driver for a PRT for the mobile opportunistic commerce domain is the difference between the value to the driver for the plans given in the PRT and her cost.

Chapter 3

General Models of Helpful Behavior

This chapter presents a decision-theoretic mechanism for agents to decide whether to undertake helpful behavior. The mechanism demonstrates the way the PRT representation can be used to enable decision-theoretic reasoning on formal teamwork models. The representation allows agents to make decisions about performing helpful behavior in a way that accommodates others' possibly different beliefs about the world and about each other's plans.

We consider three types of helpful behavior: communicating with a partner, adopting a commitment to perform a helpful act and abandoning a commitment to perform a specific domain action. These different types of behavior change the recipes agents use, but in different ways. Abandoning and adopting commitments to perform an action are active approaches to changing the group plan, whereas communicating an observation to a partner is more passive. An agent that decides to communicate information to a partner will cause the partner to update its context to reflect this information, which may lead this partner to adopt a new recipe that is more likely to succeed. For example, Bob informing Alice about the lack of tomatoes may result in Alice updating her plan so that it does not contain tomatoes and is more likely to succeed. An agent that performs a helpful domain action is doing so to directly improve the way the group activity is being done. Similarly, an agent that abandons a commitment for doing a domain action is doing so to improve the collaborative utility of a SharedPlan. For example, Bob buying tomatoes for Alice is a direct attempt to improve the likelihood that her plan will succeed.

The chapter is organized as follows: Section 3.1 defines agents' commitment to the success of a collaborative activity. Sections 3.2, 3.3 and 3.4 present decision-theoretic rules for determining whether to adopt or abandon a commitment to do an action, and for deciding whether to communicate with partners to improve the utility of the collaborative activity. Section 3.6 describes the empirical evaluation of these rules.

3.1 Commitment to Helpful Behavior

One of the key requirements for having a successful collaborative activity is agents' commitment to the success of the activity. As discussed in Section 2.1.1, this requirement is addressed in the SharedPlan formalization by the modal operator *Int. Th.* This intention results in agents avoiding conflicting intentions, meshing plans with other agents, helping each other when needed.

This section presents a new definition for agents' commitment to the success of a collaborative activity that motivates agents not only to succeed in performing the activity, but to maximize the collaborative utility associated with doing the activity. Thus, this definition forms the basis for agents that use the decision-theoretic algorithms presented in this chapter to reason about helpful behavior in a way that improves the utility of the collaborative activity. Table 3.1 presents the notation used in this definition of commitment.

Definition 4 : Committed(G_q, GR, α), is true if agent $G_q \in GR$ is committed to GR's success in doing α.

Committed(G_q, GR, α) iff

[($G_q \in GR$) $\wedge$ SP(GR, α, C_α) $\wedge$

($\exists PRT_\alpha \in \mathcal{PRT}$)

[Bel(G_q, ($\forall PRT_i \in \mathcal{PRT}_{-\alpha}$)

[Eval(GR, PRT_i, C_α) $\leqslant$Eval$(GR, PRT_\alpha, C_\alpha)$]) $\wedge$

Int.Th(G_q, Selected-PRT(GR, PRT_α, C_{GR}))]]

Definition 4 defines an agent's commitment to a collaborative activity in a way that reflects that this commitment must include the agent's intention that a recipe that is optimal for the group (given the agent's beliefs) is selected. The clause $Committed(G_q, GR, \alpha)$ refers to the commitment of agent G_q to the success of the group GR for achieving α, when there exists a recipe PRT_α that G_q believes will maximize the group utility, and G_q intends that all group members intend to carry out PRT_α at execution time, where $\mathcal{PRT}_{-\alpha}$ denotes the universe of PRTs excluding PRT_α.

For example, $Committed(Alice, \{Alice, Bob\}, \textit{dinner-making})$ refers to Alice's commitment to the success of the dinner-making plan. Alice intends that the way dinner-making is accomplished in collaboration with Bob will maximize their collaborative utility. Among the recipes available for dinner-making, if doing mushroom-puffs as the appetizer and beef as the entree offers the highest expected utility, Alice intends that the plan chosen for dinner-making consists of making these two dishes.

Table 3.1: Summary of notations used in Definition 4.

Type	**Notation**	**Meaning**
Action	$\alpha \in \Omega$	top level action
Agents	$GR \subset \mathcal{A}$ $G_q \in GR$	agents involved in SharedPlan for α agent committed to GR's success
Time	T_α	time of execution for α
Contexts	$C_\alpha \in \mathcal{C}$ $C_{GR} \in \mathcal{C}$	$Context(C_\alpha, G_q, \alpha, T_\alpha)$ $Bel(G_q, Context(C_{GR}, GR, \alpha, T_\alpha))$
$\mathcal{PRT}$	$PRT_\alpha \in \mathcal{PRT}$ $\mathcal{PRT}_{-\alpha} \subset \mathcal{PRT}$	PRT selected for action α universe of PRTs excluding PRT_α

Predicates	SP(GR, α, C_α)
	Selected-PRT(GR, PRT_α, C_{GR})
		is true if GR has a SharedPlan for α in C_α
		is true if PRT_α is selected by GR in C_{GR}

3.2 Adopting a Commitment to Do a Helpful Action

An agent commits to performing a helpful action if doing so improves the way its partner is doing a constituent action of the collaborative activity and thus improves the expected utility of the collaborative activity.

Algorithm 5 specifies the process for making the decision whether to perform a helpful action γ. The first conditional establishes that agent G_q is committed to the success of the group's achieving α. The second conditional holds if G_q believes that it can improve the utility of doing α by doing the helpful act γ. To compute the utility of adding helpful

Algorithm 5 : Helpful-Act(G_q**,** GR**,** α**,** γ), G_q helps GR in doing α by doing γ if doing so increases the expected utility of GR for doing action α, and G_q is committed to GR's success in doing α.

```
if Committed($G_q$, $GR$, $\alpha$) then
    $PRT_\alpha$ := Predict-PRT($G_q, GR, \alpha, C_{GR}$)
    $PRT_\gamma$ := Select-PRT($G_q, \gamma, C_\gamma$)
    $PRT_\alpha^T$ := $PRT_\alpha \bigcup PRT_\gamma$
    $PRT_\alpha^{Add}$ := Update-PRT($PRT_\alpha^T$, $C_\alpha$)
    utility := Eval($GR$,$PRT_\alpha^{Add}$,$C_\alpha$) - Eval($GR$,$PRT_\alpha$,$C_\alpha$)
    if utility $>$ 0 then
        Int.To($G_q$, $\gamma$, $C_\gamma$)
    end if
end if
```

action γ, G_q reasons about the recipes it will adopt for helpful act γ and the improvement it can generate by adding these recipes to the ones selected by GR for action α. Agent G_q intends to do γ, if it believes that doing γ will lead to an improvement in the group's utility for carrying out α.

The algorithms presented in this chapter make use of several functions that refer to the decision-making models used in collaborative activities. The function $Select\text{-}PRT(G_q, \alpha, C_\alpha)$ refers to the PRT that represents G_q's belief about the possible recipes it will select to perform action α in context C_α. The function $Predict\text{-}PRT(G_q, G_r, \alpha, C_\alpha)$ refers to the PRT that represents G_q's belief about the possible recipes G_r will select to perform action α in context C_α. If the subtrees of a PRT are not independent but connected through

Table 3.2: Summary of notations used in Algorithm 5 and Algorithm 6.

Type	**Notation**	**Meaning**
Action	$\alpha \in \Omega$	top level action
	$\gamma \in \Omega$	helpful act
Time	T_i	current time
Agents	$GR \subset \mathcal{A}$	agents involved in SharedPlan for α
	$G_q \in GR$	agent committed to GR's success in doing α
Contexts	$C_\alpha \in \mathcal{C}$	$Context(C_\alpha, G_q, \alpha, T_i)$
	$C_{GR} \in \mathcal{C}$	$Bel(G_q, Context(C_{GR}, GR, \alpha, T_i))$
	$C_\gamma \in \mathcal{C}$	$Context(C_\gamma, G_q, \gamma, T_i))$
$\mathcal{PRT}$	$PRT_\alpha \in \mathcal{PRT}$	PRT selected for action α
	$PRT_\gamma \in \mathcal{PRT}$	PRT selected for action γ

temporal or resource constraints, updating a subtree of the PRT may require updating the other subtrees respectively to propagate the changes. For example, if the desert-making action is added to the PRT for dinner-making and Alice is committed to making the desert, the timing of entree-making action and its constituent actions may need to get updated to accommodate time for the new commitment. The function $Update\text{-}PRT$ refers to this additional update that may be required after an addition, subtraction or replacement operator is applied to a PRT. $Update\text{-}PRT(PRT_\alpha, C_\alpha)$ represents the PRT that is updated from PRT_α in context C_α by propagating the changes introduced by the PRT operators and resolving possible conflicts that may have occurred as a result of these changes.

The way Algorithm 5 can be used to decide whether to perform a helpful act can be demonstrated on the dinner-making example. If Bob believes Alice may intend to make pasta with tomato sauce, but he knows there are no fresh tomatoes left in the kitchen, he can perform a helpful action by adopting an intention to go to the market and buy some tomatoes. Having tomatoes in the kitchen may help Alice to perform her entree plan successfully and thus improve the expected utility of the group for the dinner-making plan. However, going to the market and buying tomatoes are associated with a cost for time spent in the store and the charge for the tomatoes. Bob should commit to going to the market and buying tomatoes only if the cost of the helpful action is lower than the potential benefit to the dinner they are making.

Figure 3.1 illustrates the use of Algorithm 5 in the opportunistic commerce example to determine whether it is beneficial for the agent to perform an action for getting the car fixed in a way that may help the driver in her commute plan. [1] To decide whether to do the helpful act, the agent reasons about how the plan for doing the opportunistic commerce action would change with the addition of the action. Part (a) of Figure 3.1 represents the agent's beliefs about the way the collaborative activity is being accomplished. Based on the agent's beliefs, the current plan is expected to take 19 minutes. Part (b) of the figure represents the agent's beliefs about the updated plan which includes fixing the car and then commuting and doing errands. If the agent hires a repairman to get the car fixed, the commute plan of the driver is expected to take 4.6 minutes less than the original plan. The agent adopts an intention to get the car fixed if the expected utility of the updated PRT with a helpful action added (Figure 3.1 part (b)) is higher than the expected utility of the original

[1] The ordering of nodes in a PRT can be arbitrary. It does not necessarily reflect the temporal ordering among constituent actions. The nodes of a PRT are associated with the properties of the actions they represent. These properties include the timing of the actions.

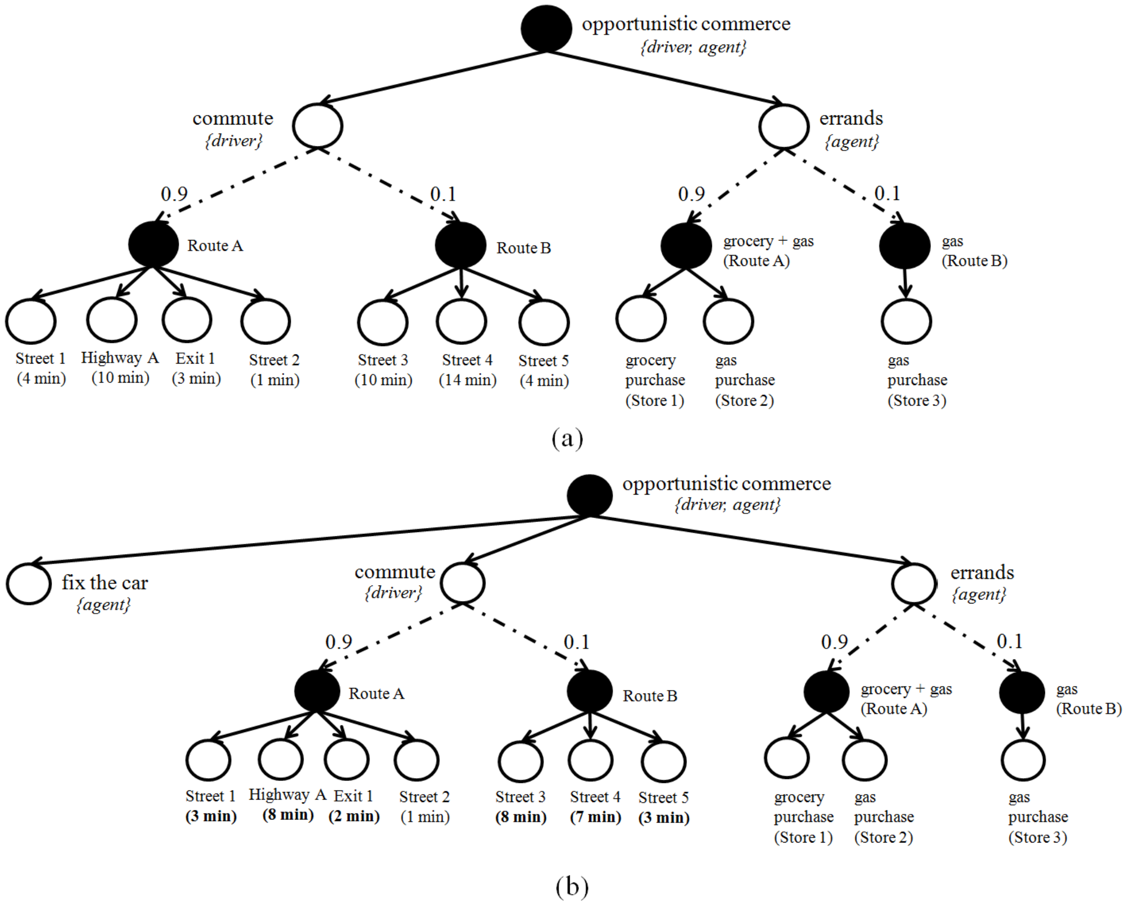


Figure 3.1: (a) A PRT reflecting the agent's beliefs about the original plan. (b) An updated PRT that reflects the agent's beliefs about the way opportunistic commerce action is done after adopting a commitment to fix the car.

plan (Figure 3.1 part (a)).

3.3 Abandoning Commitment to Do an Action

After making observations about the world and about other agents' plans, an agent may realize that a constituent action that the agent is committed to doing no longer improves the utility of the collaborative activity. Therefore, an agent may consider dropping its commitment to do an action to improve the expected utility of the activity. This section presents a decision-theoretic algorithm that agents can use to decide whether to abandon a commitment to do an action.

Algorithm 6 : Abandon(G_q**,** GR**,** α**,** γ**)**, G_q helps GR in doing α by dropping its commitment to do γ if doing so increases the expected utility of GR for doing action α, and G_q is committed to GR's success in doing α.

```
if Committed($G_q$, $GR$, $\alpha$) then
  if Int.To($G_q$, $\gamma$, $C_\gamma$) then
    $PRT_\alpha$ := Predict-PRT($G_q, GR, \alpha, C_{GR}$)
    $PRT_\gamma$ := Select-PRT($G_q, \gamma, C_\gamma$)
    $PRT_\alpha^T$ := $PRT_\alpha \setminus PRT_\gamma$
    $PRT_\alpha^{Drop}$ := Update-PRT($PRT_\alpha^T, C_\alpha$)
    utility := Eval($GR$,$PRT_\alpha^{Drop}$,$C_\alpha$) - Eval($GR$,$PRT_\alpha$,$C_\alpha$)
    if utility $> 0$ then
      $\neg$Int.To($G_q$, $\gamma$, $C_\gamma$)
    end if
  end if
end if
```

The process for making the decision whether to abandon commitment to do action γ is specified by Algorithm 6. The first conditional establishes that agent G_q is committed to the success of the group for achieving α. The second conditional holds if G_q has already adopted an intention to do γ. The third conditional checks whether G_q believes that it can improve the utility of the action α by dropping its commitment for doing γ. To compute the utility of G_q abandoning its commitment, G_q reasons about the recipes selected for doing γ and the improvement it can generate by subtracting these recipes from the ones selected by GR for action α. Agent G_q intends to abandon doing γ, if it believes that doing so will lead to an improvement in the group's utility for carrying out α.

Figure 3.2 illustrates the way Algorithm 6 can be used in the opportunistic commerce example to determine whether to abandon an intention for doing a constituent action of the collaborative plan. Part (a) of Figure 3.2 represents the agent's belief about the way opportunistic commerce action is being accomplished. Based on its beliefs about the world and other agents, the agent has adopted a commitment to purchase gas and groceries if Route A is chosen by the driver. If the agent observes that the gas station on Route A is crowded and the duration for purchasing gas from that station is higher than expected, the agent may consider abandoning its commitment for purchasing gas from that gas station. If the agent drops its commitment, the errands plan on Route A is expected to take 10 minutes less. Part (b) of Figure 3.2 represents the agent's belief about the way opportunistic commerce action would be accomplished after abandoning this commitment. The agent abandons its commitment for purchasing gas if the utility of the updated PRT with fewer actions to do is higher than the expected utility of the original plan.

3.4 Deciding to Communicate

The ability to communicate information allows agents to convey information about and ask about changes in the world and in the way the collaborative activity is being done. As a result of sharing information, agents update their context and consequently their own part

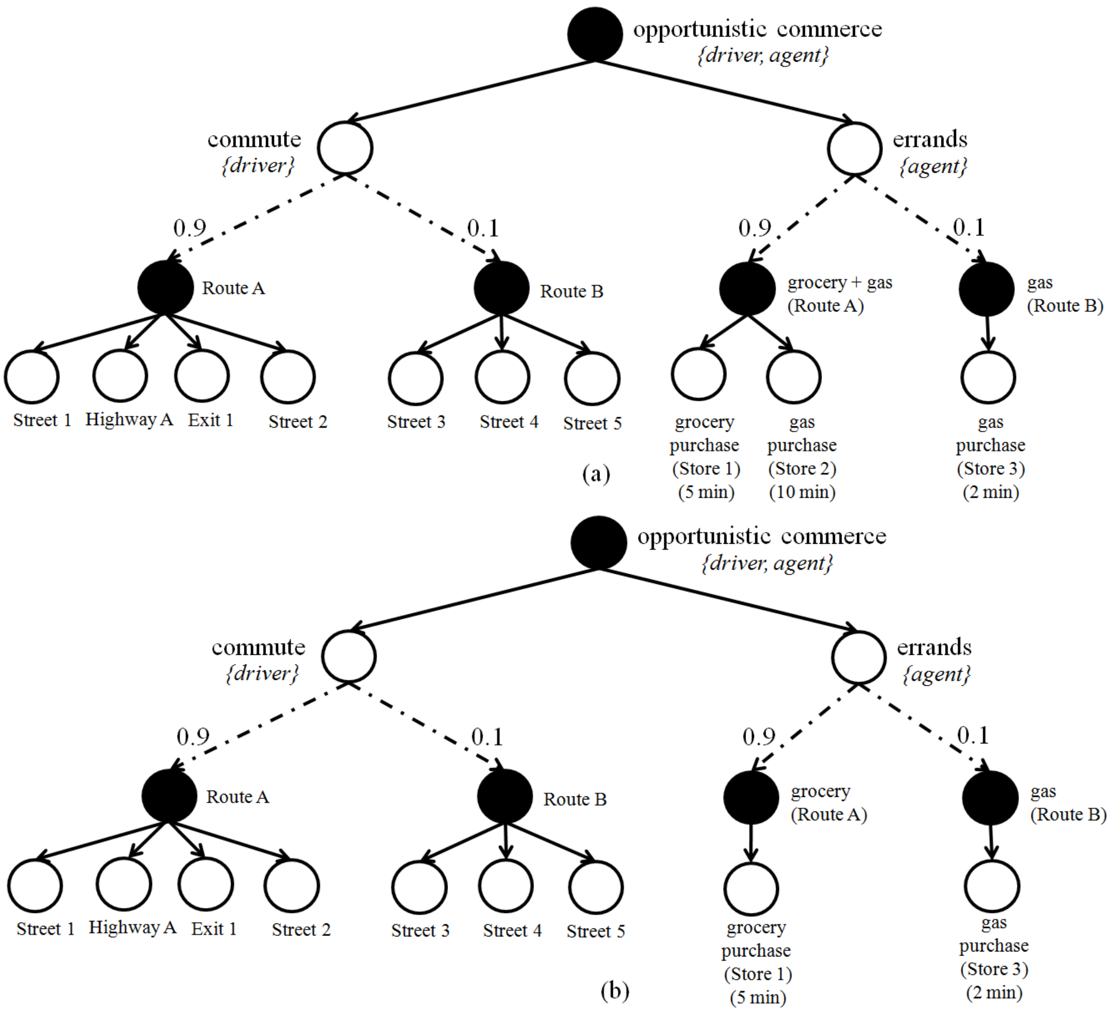


Figure 3.2: (a) A PRT reflecting the agent's beliefs about the original plan. (b) An updated PRT that reflects the agent's beliefs about the way opportunistic commerce action is done after abandoning its commitment for purchasing gas.

of the plan in a way that is more likely to succeed. This section considers two types of communication, informing and asking for information, and presents algorithms for deciding whether to communicate.

3.4.1 Conveying Information

After making an observation about the world and about other agents' plans, an agent may consider informing its partner about this observation. To decide whether to inform its partner, the agent considers the way the partner's context and plan would get updated as a result of hearing about the observation and reasons about the effect of this updated plan on the expected utility of the collaborative plan.

Algorithm 7 specifies the process by which an agent can reason about the trade-off between the utility and cost of informing its partner about an observation. In situations in which two agents G_q and G_r are committed to the success of a collaborative activity, when G_q makes an observation, it needs to reason about informing G_r about this observation. The decision to communicate may improve the utility of the group, but communication is associated with a cost. G_q reasons about the recipes that G_r would adopt for doing subaction β if G_q has communicated observation o. If the utility gain to the group from the adoption of this recipe is higher than the cost of communication, then G_q will communicate o to G_r.

In Algorithm 7, $Comm(G_q, G_r, o)$ refers to the inform action, $COC(G_q, G_r, o)$ represents the cost of G_q communicating o with G_r. $Context\text{-}Update(C_\beta^{Bel}, o)$ represents context C_β^{Bel} updated with observation o.

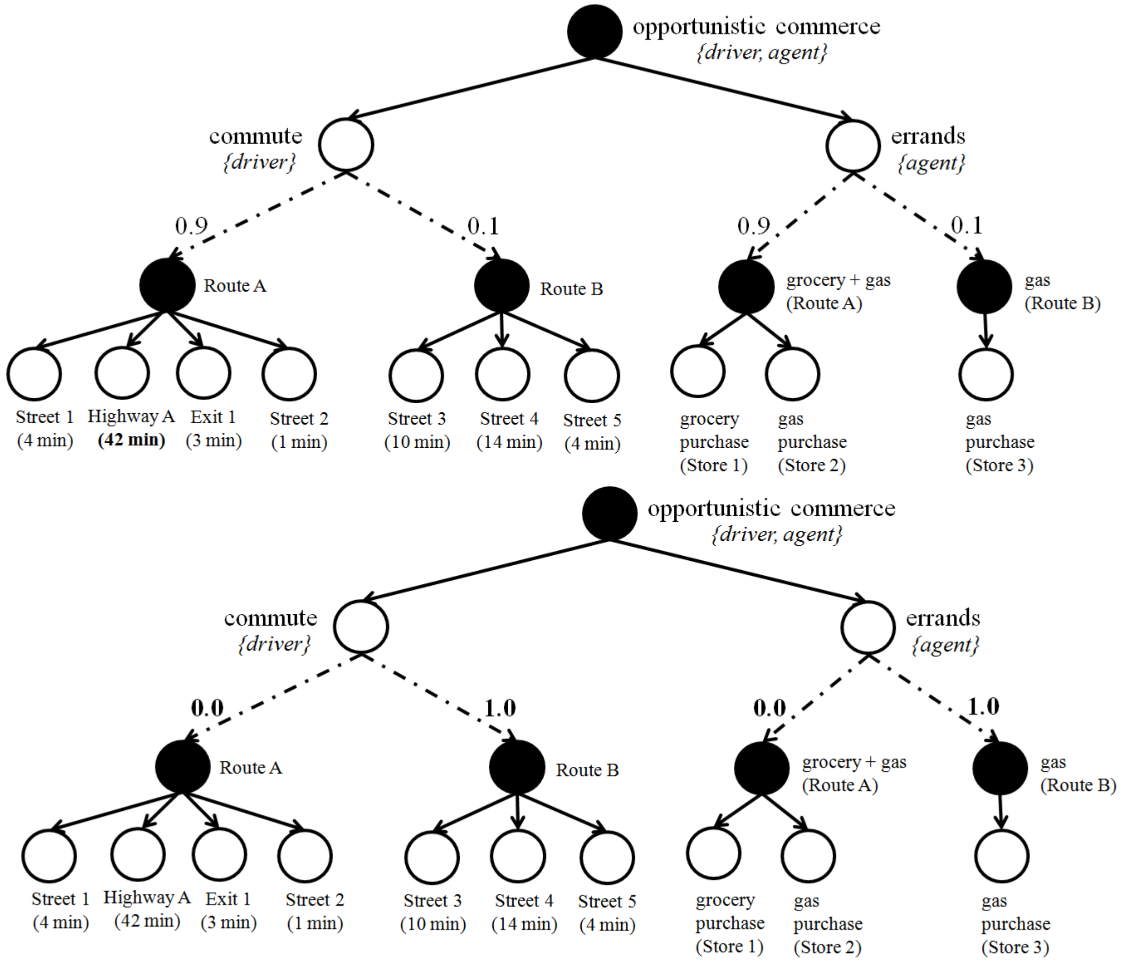
In the dinner-making example, if Bob sees that there are no tomatoes in the kitchen, and he thinks that Alice is making a tomato sauce, he would conclude that their dinner-making is likely to fail. If he informs Alice about this observation, Alice can update her recipe so that it does not contain tomatoes. If Bob forecasts that the utility improvement generated by Alice updating her recipe given the observation is higher than communication costs, Bob

Algorithm 7 : Inform(G_q, G_r, α, β, o), G_q informs G_r about observation o, if doing so increases the expected utility of GR for α, and G_q is committed to GR's success in doing α, where G_r has intention to do β.

```
if Committed($G_q$, $GR$, $\alpha$) then
  $PRT_\alpha$ := Predict-PRT($G_q, GR, \alpha, C_{GR}$)
  $C^o_\beta$ := Context-Update($C^{Bel}_\beta$, $o$)
  $PRT^o_\beta$ := Predict-PRT($G_q, G_r, \beta, C^o_\beta$)
  $PRT^T_\alpha$ := $PRT_\alpha \otimes PRT^o_\beta$
  $PRT^o_\alpha$ := Update-PRT($PRT^T_\alpha$, $C_\alpha$)
  utility := Eval($GR$, $\alpha$, $PRT^o_\alpha$, $C_\alpha$) $-$ Eval($GR$, $\alpha$, $PRT_\alpha$, $C_\alpha$)
  if utility $> COC(G_q, G_r, o)$ then
    Int.To($G_q$, Comm$(G_q, G_r, o)$, $C_\alpha$)
  end if
end if
```

will inform Alice. However, if Bob believes that Alice is likely not to be using tomatoes or the communication cost is very high, then he would not inform Alice.

Figure 3.3 illustrates the way Algorithm 7 can be used in the opportunistic commerce example to determine whether it is beneficial for the agent to inform the driver about a traffic congestion on Route A. To evaluate the utility of informing, the agent reasons about the way this observation changes the utility of the original plan for opportunistic commerce. The agent's beliefs about the original plan are represented by the PRT shown in Part (a) of Figure 3.3. Having a congestion on Route A increases the expected duration of the commute plan to 47.8 minutes. Next, the agent reasons about the way the opportunistic



(b)

Figure 3.3: (a) A PRT reflecting the agent's beliefs about the plan that would be followed, if there is a traffic congestion and the driver does not know about it. (b) An updated PRT that reflects the agent's beliefs about the way opportunistic commerce action would be done, if there is traffic congestion and the driver knows about it.

Table 3.3: Summary of notations used in Algorithms 7, 8 and 9.

Type	**Notation**	**Meaning**
Action	$\alpha \in \Omega$	top level action
	$\beta \in \Omega$	$\beta \in sub\text{-}actions(\alpha)$
Time	T_i	current time
Agents	$GR \subset \mathcal{A}$	agents involved in SharedPlan for α
	$G_q \in GR$	agent committed to GR's success in doing α
	$G_r \in GR$	partner(s) of G_q
Contexts	$C_\alpha \in \mathcal{C}$	$Context(C_\alpha, G_q, \alpha, T_i)$
	$C_{GR} \in \mathcal{C}$	$Bel(G_q, Context(C_{GR}, GR, \alpha, T_i))$
	$C_\beta \in \mathcal{C}$	$Context(C_\beta, G_q, \beta, T_i))$
	$C_\beta^{Bel} \in \mathcal{C}$	$Bel(G_q, Context(C_\beta^{Bel}, G_r, \beta, T_i))$
$\mathcal{PRT}$	$PRT_\alpha \in \mathcal{PRT}$	PRT selected for action α
Cost	$COC(G_q, G_r, o)$	cost of G_q communicating o with G_r

commerce plan would change after the driver hears about the congestion. Part (b) of the figure represents the agent's beliefs about the updated plan that would be generated if the driver finds out that there is congestion on the highway. After hearing about the congestion, the driver would choose Route B to commute home, which would decrease the expected duration of the commute to 28 minutes. It is beneficial for the agent to inform the driver about the congestion, if the cost of communication is less than the cost of driving for 19.8 minutes.

3.4.2 Asking for Information

If an agent believes that it can improve the expected utility of a collaborative activity by obtaining some information about the world or the way the activity is being done, the agent may consider asking its partner for information. To decide whether to ask its partner for information, the agent considers the set of possible answers it can receive from its partner, and the way its beliefs about the activity being done would change according to each possible answer.

Agent G_q reasons about asking G_r for ψ, if this information would beneficially change G_q's recipe for doing subaction β. To compute the utility of asking G_r about ψ, G_q needs to consider how it will adapt its own belief about the recipes it will select, for each possible answer that is provided by G_r. Agent G_q computes the difference in expected group utility for α between its initial belief about recipes to select for α and any refined belief that is a result of the answer given by G_r.

In algorithm 4, C_α^o and C_β^o are contexts updated by G_q with answer o, $Comm(G_q, G_r, \psi)$ refers to the ask action, $COC(G_q, G_r, \psi)$ represents the cost of communicating with G_r, $\Psi(\psi)$ is the set of observations that are answers to ψ, $d\text{-}Bel(G_q, Bel(G_r, o))$ is G_q's prediction of the probability of receiving o from G_r as an answer to ψ. G_q believes there exists an observation $o \in \Psi(\psi)$ that G_r believes to be the correct answer to question ψ.

For example, Alice can reason about asking Bob about tomatoes to improve her plan for entree making. If Alice has a plan for making tomato sauce, but believes that her plan may fail as a result of there being no good tomatoes, she can ask Bob if he knows the availability of tomatoes. For each possible answer Alice may receive from Bob, she updates her belief about recipes to select that incorporate that answer. After weighting each possible updated

Algorithm 8 : Ask(G_q, G_r, α, β, ψ), G_q committed to doing β, asks G_r question ψ, if doing so increases the expected utility of GR's doing α, and G_r is committed to GR's success in doing α.

```
if Committed($G_r$, $GR$, $\alpha$) then
    $PRT_\alpha$ := Predict-PRT($G_q, GR, \alpha, C_{GR}$)
    for $o \in \Psi(\psi)$ do
        $C^o_\beta$ := Context-Update($C_\beta, o$)
        $C^o_\alpha$ := Context-Update($C_\alpha, o$)
        $PRT^o_\beta$ := Select-PRT($G_q, \beta, C^o_\beta$)
        $PRT^T_\alpha$ := $PRT_\alpha \otimes PRT^o_\beta$
        $PRT^o_\alpha$ := Update-PRT($PRT^T_\alpha$, $C^o_\alpha$)
        utility := utility + $d\text{-}Bel(G_q, Bel(G_r, o)) \times$
                (Eval($GR$,$\alpha$,$PRT^o_\alpha$,$C^o_\alpha$) $-$ Eval($GR$, $\alpha$, $PRT_\alpha$, $C^o_\alpha$))
    end for
    if utility $> COC(G_q, G_r, \psi)$ then
        Int.To($G_r$, Comm$(G_q, G_r, \psi)$, $C_\alpha$)
    end if
end if
```

recipe with the probability of receiving that answer, Alice computes the expected utility for asking. If it is higher than the communication cost, Alice considers asking. However, if Alice believes that the answer will not improve the recipe she selects, or the cost of communication is very high, then she would not consider communicating with Bob.

When an observer agent informs a recipient agent about an observation, this commu-

Algorithm 9 : Ask-MB(G_q, G_r, α, β, γ, ψ), G_q committed to doing β, asks G_r, which is committed to doing γ, question ψ, if doing so increases the expected utility of GR's doing α, where G_r is committed to GR's success in doing α.

```
if Committed($G_r$, $GR$, $\alpha$) then
  $PRT_\alpha$ := Predict-PRT($G_q, GR, \alpha, C_{GR}$)
  for $o \in \Psi(\psi)$ do
    $C_\beta^o$ := Context-Update($C_\beta, o$)
    $C_\gamma^o$ := Context-Update($C_\gamma, o$)
    $C_\alpha^o$ := Context-Update($C_\alpha, o$)
    $PRT_\beta^o$ := Select-PRT($G_q, \beta, C_\beta^o$)
    $PRT_\gamma^o$ := Predict-PRT($G_q, G_r, \gamma, C_\gamma^o$)
    $PRT_\alpha^{T_1}$ := $PRT_\alpha \otimes PRT_\beta^o$
    $PRT_\alpha^{T_2}$ := $PRT_\alpha^{T_1} \otimes PRT_\gamma^o$
    $PRT_\alpha^o$ := Update-PRT($PRT_\alpha^{T_2}$, $C_\alpha^o$)
    utility := utility + $d\text{-}Bel(G_q, Bel(G_r, o)) \times$
          (Eval($GR$,$\alpha$,$PRT_\alpha^o$,$C_\alpha^o$) $-$ Eval($GR$, $\alpha$, $PRT_\alpha$, $C_\alpha^o$))
  end for
  if utility $> COC(G_q, G_r, \psi)$ then
    Int.To($G_r$, Comm$(G_q, G_r, \psi)$, $C_\alpha$)
  end if
end if
```

nication action leads to mutual belief among the agents. After sharing an observation, not only do both agents believe that the observation holds in the world, but they also believe that the other agent believes that the observation holds in the world. Algorithm 8 assumes that the shared observation only affects the action the recipient agent is committed to doing, and computes the benefit of informing in terms of the recipient agent's updating its beliefs about doing its part of the collaborative activity. However, if the shared information affects the action of the observer agent, the recipient agent should also update its beliefs about the way the observer agent performs its part of the activity given the observation. The algorithm for calculating the benefit of informing should also consider the effect of this update on the expected utility of the activity. Algorithm 9 is an extension of Algorithm 8 that reasons about the recipes G_r would adopt for doing subaction β and also the way G_r's belief would change about the way G_q is doing subaction γ if G_q asks G_r about question ψ.

The extent to which decisions to ask a partner are profitable depends on how well agents are able to model how the world changes. If G_q believes the world is uncertain but it is not, it will keep asking needlessly. If G_q is not expecting any changes but, the world is changing, it fails to ask when needed.

The complexity of Algorithms 8 and 9 depends heavily on the size of $\Psi(\psi)$, the number of observations that are answers to question ψ. Recent work has provided techniques to facilitate this computation for situations with large numbers of possible answers in collaborative settings (Sarne and Grosz, 2007).

The way Algorithm 9 can be used to determine whether to for ask information from a partner can be demonstrated on the example presented in Figure 3.3 in the opportunistic commerce domain. The example is modified so that the PRTs presented in Figure 9 now represent the driver's beliefs about the way opportunistic commerce action is being accomplished. In this example, the driver reasons about asking the agent about the traffic conditions in the city. The driver believes that the likelihood of a traffic congestion on Route A is 30%. To determine whether to ask, the driver needs to reason about the way knowing about this congestion changes the success of the original plan for each possible answer she can receive from the agent. Figure 2.4 represents the driver's belief about the original opportunistic commerce plan, if there is no congestion. Part (a) of Figure 3.3 represents the way the driver's belief would change about the success of the original plan if she finds out that there is a congestion on the highway. In the case of congestion, the expected duration of the original plan would be 28.8 minutes higher. As formalized in Algorithm 9, if the driver finds out about the traffic conditions, the driver's beliefs about the opportunistic commerce plan would be updated to reflect the way she would change her commute plan and also to reflect the driver's updated beliefs about the errands plan. Part (b) of the figure represents the driver's beliefs about the updated plan that would be generated, if she finds out that there is congestion on the highway. The expected duration of the opportunistic commerce plan decreases 8.4 minutes if she asks the agent about the traffic and updates her commute plan to Route B. It's beneficial to ask for information if the cost of asking is lower than this additional cost for driving.

3.5 Other Uses of Helpful Behavior Algorithms

The set of recipes agents have for forming collaborative plans provides guidelines on performing constituent parts of a collaborative activity. A recipe may need to be enriched with supporting actions if doing them helps to satisfy some constraints of the recipe and improves its likelihood of success. The decision-theoretic algorithm for adopting commitment to do a helpful action, as presented in Algorithm 5, modifies the way an action is being done in a way that makes it more likely to succeed.

Algorithm 5 is primarily designed to determine whether to adopt an intention to do an action to help a partner in doing a constituent action of a collaborative plan. But, the algorithm can also be used for intention reconciliation in a more general way: to determine whether to adopt an intention to do a new individual constituent action that directly improves the utility of a collaborative activity as a result of accomplishing more. Adding constituent actions to a recipe may lead to the discovery of new ways for doing an action.

Figure 3.4 illustrates the way Algorithm 5 can be applied to the opportunistic commerce example to determine whether to adopt an intention to do a new constituent action. In this example, the driver is considering adopting an intention to pick up a passenger to improve the overall utility of the collaborative activity. If the driver picks up the passenger, she will spend 4 additional minutes driving, but the duration of the highway action is expected to be 3.6 minutes less as a result of using the carpool lane. Additionally, the group will acquire additional value for accomplishing this new action. The new constituent action is beneficial for the group if the value of the action is higher than the driver's cost for driving additional 4 minutes in the given context. The driver adopts an intention to pick up the passenger if the expected utility of the updated PRT with an action added (Figure 3.4 part (b)) is higher than the expected utility of the original plan (Figure 3.4 part (a)).

The algorithms for conveying information, as presented in Algorithms 7, 8 and 9, are formalized and presented for sharing observations among participants of a collaborative activity. However, the uses of these algorithms are not limited to sharing observations.

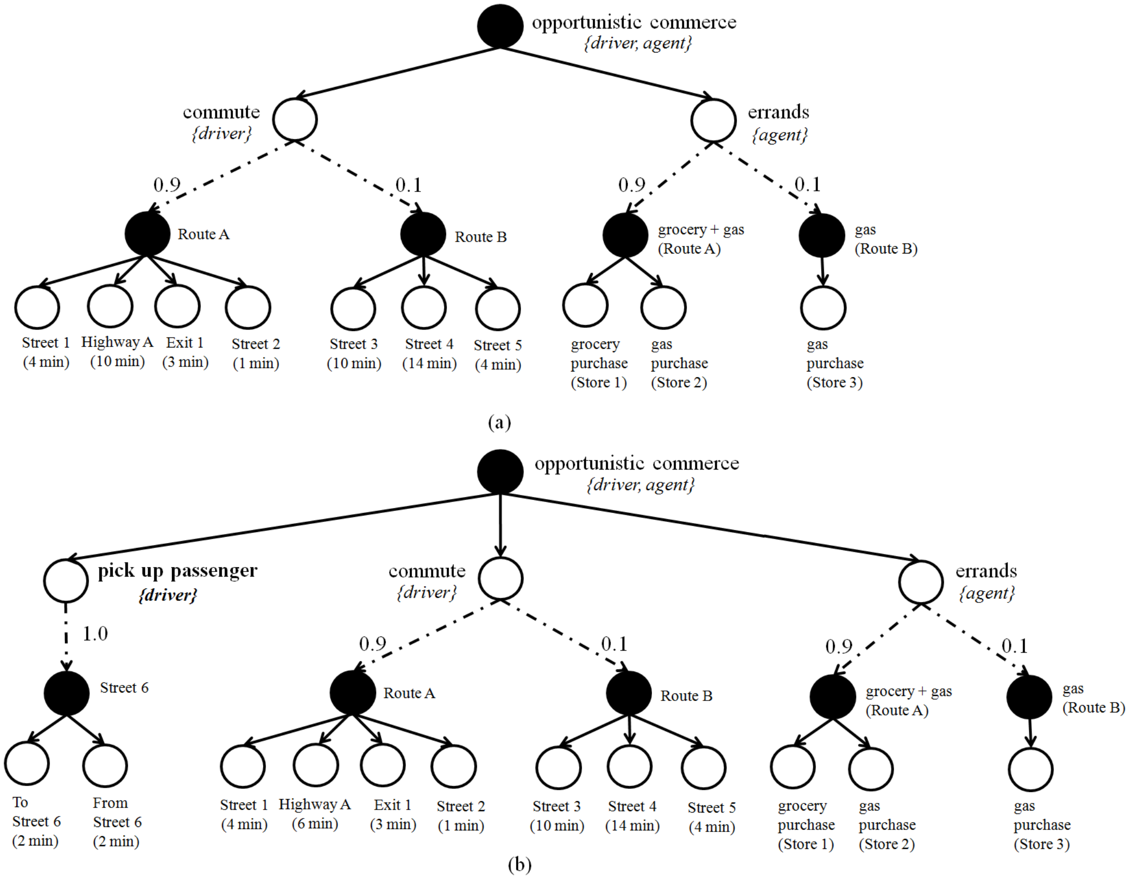


Figure 3.4: (a) A PRT reflecting the driver's beliefs about the original plan. (b) An updated PRT that reflects the driver's beliefs about the way opportunistic commerce action would be done after adopting a commitment for picking up a passenger.

Agents can use the communication algorithms to determine whether to inform their partners about the successes and failures they observe. For example, if Alice observes that she failed in grilling the chicken, but believes that knowing about this failure will not affect the way Bob prepares the appetizer, it is better for her not to inform Bob. On the other hand, if Bob can help Alice to recover from this failure (e.g., by fixing the oven), Alice considers notifying Bob.

The algorithms for managing communication provide an effective way for an agent to mesh parts of a collaborative plan to improve the plan's success. Instead of sharing observations about the world, agents can share their beliefs about the way a collaborative activity or a part of the activity is being (or should be) accomplished. In the opportunistic commerce example, as presented in Figure 3.3, the agent can use Algorithm 7 to determine whether to mesh its part of the plan with the driver's plan. To evaluate the utility of meshing plans, the agent reasons about the original plan that agents are currently following and the way that plan would change after meshing plans. If the agent informs the driver that its plan involves Route B with 100% chance, the driver would update her own part of plan accordingly and choose Route B for the commute plan. Meshing plans this way would improve the expected utility of the opportunistic commerce as the updated plan would exclude the congested highway.

Helpful behavior has been identified by many prominent teamwork formalizations as a crucial requirement for a successful collaborative activity. These models propose different ways that an agent can help a partner and support a collaborative activity: communicating successes and failures (Cohen and Levesque, 1991), meshing plans (Bratman et al., 1988) and performing helpful acts (Grosz and Kraus, 1996). The decision-theoretic algorithms presented in this chapter address all of these different ways that agents can help each other. The chapter provides a complete model for managing helpful behavior in a way that is beneficial for the collaborative activity given different costs associated with helpful actions.

3.6 Empirical Evaluation

This section provides an empirical evaluation of the decision-theoretic helpful behavior rules presented in this chapter. The evaluation used the Colored Trails (CT) system, a publicly available test-bed developed for investigating decision-making in task settings, where the key interactions are among goals, tasks required to accomplish those goals, and resources needed to perform the tasks (Grosz et al., 2004). CT is played on a rectangular board of colored squares. The players are located randomly on the board. They are given chips of the same colors used in the game board. Goal squares are positioned in various locations on the board, and the object of the game is to reach to those goals. At each turn of the game agents can move to an adjacent square on the board by surrendering a chip in the color of the square.

The empirical evaluation used a configuration of CT in which certain squares on the board may turn into traps and prevent players' advancement. Players have full visibility of the board and players' positions on the board, but cannot observe the chips the other player has. One of the players (called the observer) is able to observe the trap locations, whereas the other player (the partner) cannot. The game proceeds for a specified number of turns. At the end of the game, a score is computed for each player that depends on the number of goals the player was able to achieve, the number of chips left in its possession, and the score for the other player. The game is fully collaborative in that the final score of a player is the cumulative scores of the players in the game.

This CT game is an analogue of a task-setting in which players have partial knowledge about the world and about each other. Chips represent agents' capabilities, and a path to a goal square on the board represents agents' plans for achieving their goals. There may be several paths agents can take to reach a goal square, like there are several recipes that agents can use to achieve their goal. Traps represent the possibility that a plan may fail in a world that may change. Players have an incentive to collaborate in this game because their scores depend on each other.

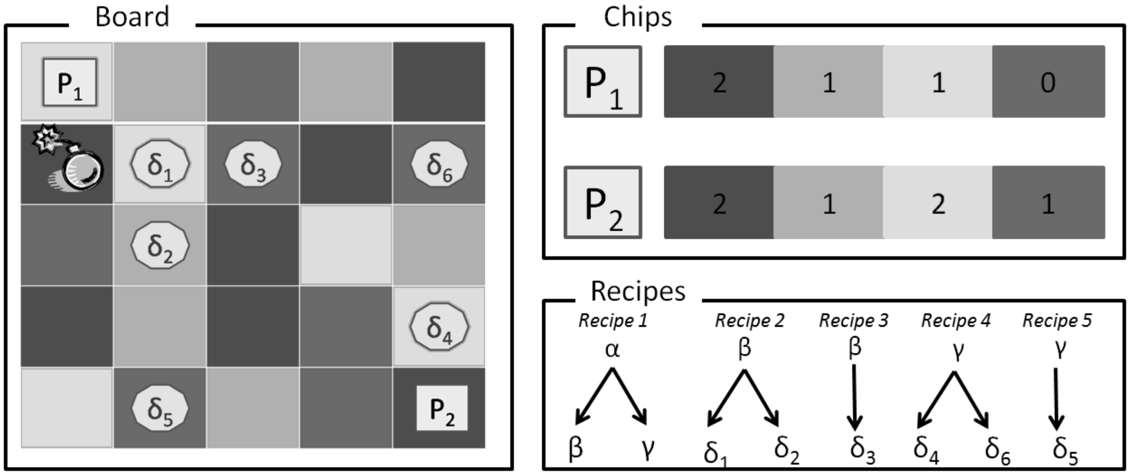


Figure 3.5: Screen-shot of the CT game.

A snapshot of the game is presented in Figure 3.5. The game board is displayed on the left of the figure; P_2 represents the observer player and P_1 represents its partner; δ_1 to δ_6 represent lower-level goals that are positioned on the board itself; α, β and γ represent higher level goals. Players' chips are shown on the top-right of the figure, and the possible recipes for α are presented on the bottom-right section of the figure.

In this game, P_1 is committed to accomplishing goal β because it is closer to the constituent goals of β, and P_2 is committed to γ. P_1 is able to achieve goal β by achieving sub-goals δ_1 and δ_2, but it cannot achieve δ_3 because it lacks a chip. P_1 is unable to observe trap positions, and a trap is located just below its current location.

In this CT game, agents' contexts include both their beliefs about the world (e.g., probability distribution over possible trap positions), and beliefs about their partner's contexts (e.g., probability distribution over the chips the partner possesses). The success probability (p-CBA value) of a path towards a goal is obtained by combining these beliefs. A player successfully moves to an adjacent square if it is not a trap position. The value of the basic action $cba.basic$ corresponds to the probability that the square the agent moves onto is not a trap. To move to an adjacent square on the game board, a player needs to hand in a chip of the same color as the square. The cost of a basic-level action corresponds to the cost of a chip. Players receive 100 points for reaching the goal (completing task α) and 10 points for reaching any of δ_i. Thus the valuation is 100 points for α, 10 for any δ_i, and zero for the rest of the goals. The agreed upon utility function of our formalism corresponds to the joint scoring function of the game. A sample PRT for a collaborative plan for the CT game is shown in Figure 3.6.

In the game, the observer can help its partner by giving away chips so that the partner is able to realize a path to the goal which was formerly inaccessible. In addition, players can communicate information about traps in one of two ways: the observer can inform its partner about trap positions, or its partner can ask about the location of traps directly. There is a cost associated with all of these helpful-behavior actions, and players need to weigh this trade-off when they engage in helpful-behavior decisions.

3.6.1 Experimental Setup and Results

The CT game described above was used as a test-bed for quantitative analysis of our decision-theoretic helpful behavior rules. Both players of the game were computer agents. The helpful behavior rules were evaluated in terms of the average score they generated

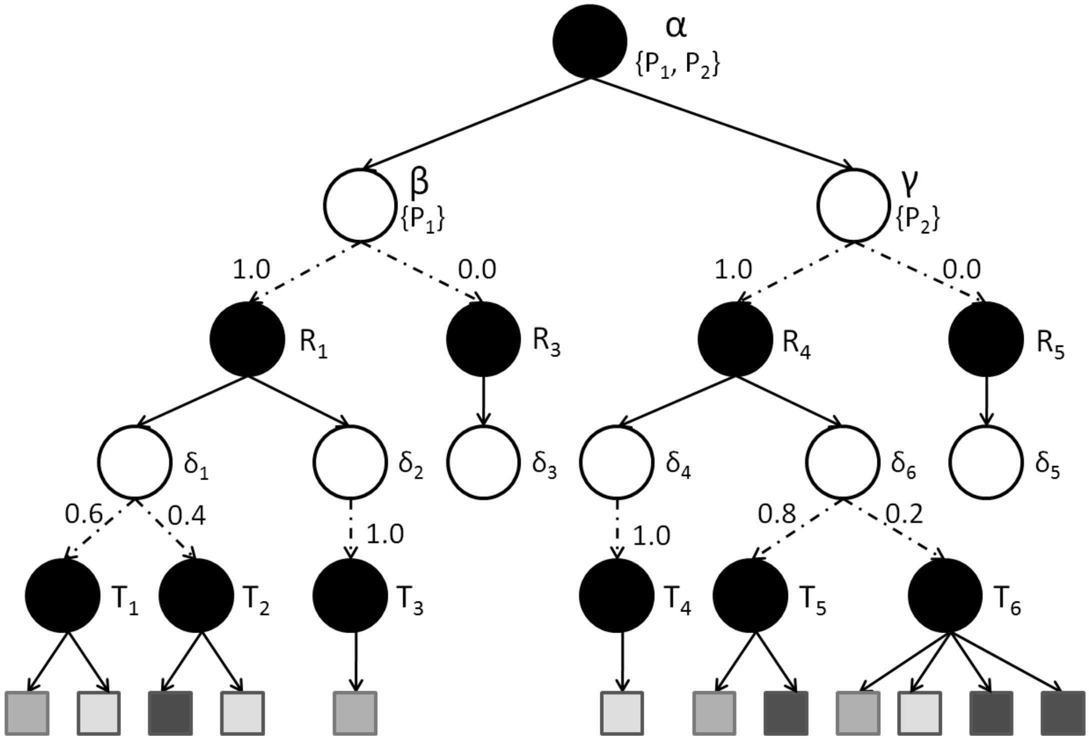


Figure 3.6: A sample PRT for the CT SharedPlan game.

across 500 game plays. The significant differences in average scores of the protocols are tested with t-test for paired two-samples for means and labeled whenever the difference was not in the 95% confidence interval. In all experiments, the chips agents possess in the game and the board layout are drawn from a uniform distribution that is common knowledge between players. The probability that a trap may appear for a given color is known to the observer, but not to its partner.

The first set of experiments compared the following three protocols for deciding whether the observer agent should perform a helpful act: The Helpful Act protocol uses Algorithm 5 to determine whether to perform a helpful act; the Random Help protocol gives away a random colored chip; the No Help protocol never gives away chips. These experiments varied whether the observer has complete or incomplete information about the chips of its partners. Figure 3.7 shows the joint scores that players achieve by using the different protocols.

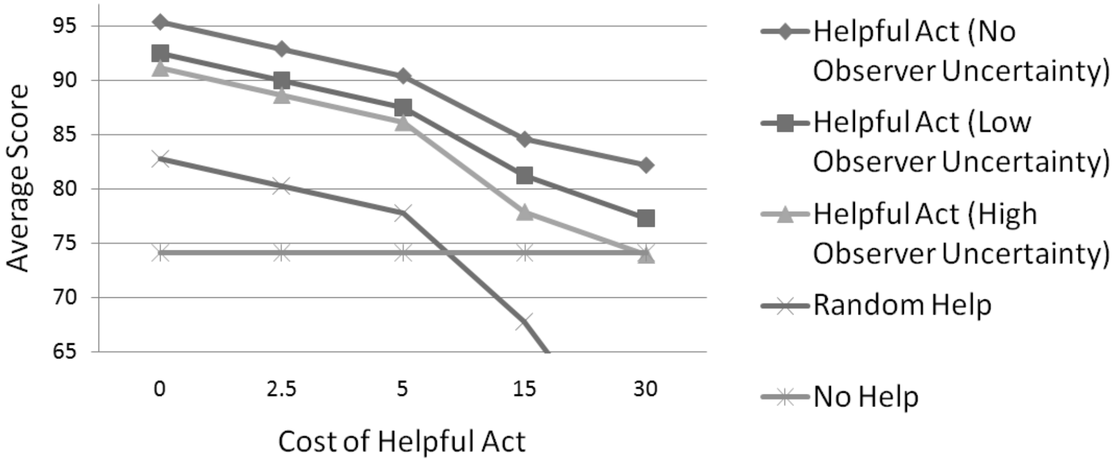


Figure 3.7: Performance of helpful act protocols by helpful-act cost, observer uncertainty.

The results show that the Helpful Act protocol performs significantly better than the Random Help and the No Help protocols. The only exception is the case in which the cost of a helpful act is 30. In that case, the performance of Helpful Act with high observer uncertainty is not significantly different than No Help. In general the performance of the Helpful Act protocol improves significantly as the observer's uncertainty decreases.

The second set of experiments compared four different communication protocols. The Inform protocol uses Algorithm 7 to determine whether the observer should tell its partner about trap positions. The Ask protocol utilizes Algorithm 8 to determine whether the partner should ask the observer about trap positions. In the Always Inform protocol (AI), the observer always informs its partner, regardless of its partner's need. The Never Communicate protocol (NC) does not allow any type of communication. We varied three factors to assess the performance of these protocols: the communication cost, observer uncertainty about the partner's chips, and the probability that traps occur on the board (uncertainty in the world). The players are provided accurate models of the uncertainty in the world. As the probability that traps occur increases, and thus the partner player's uncertainty about the world grows, we expected the benefit of communication to increase because the partner cannot observe trap positions.

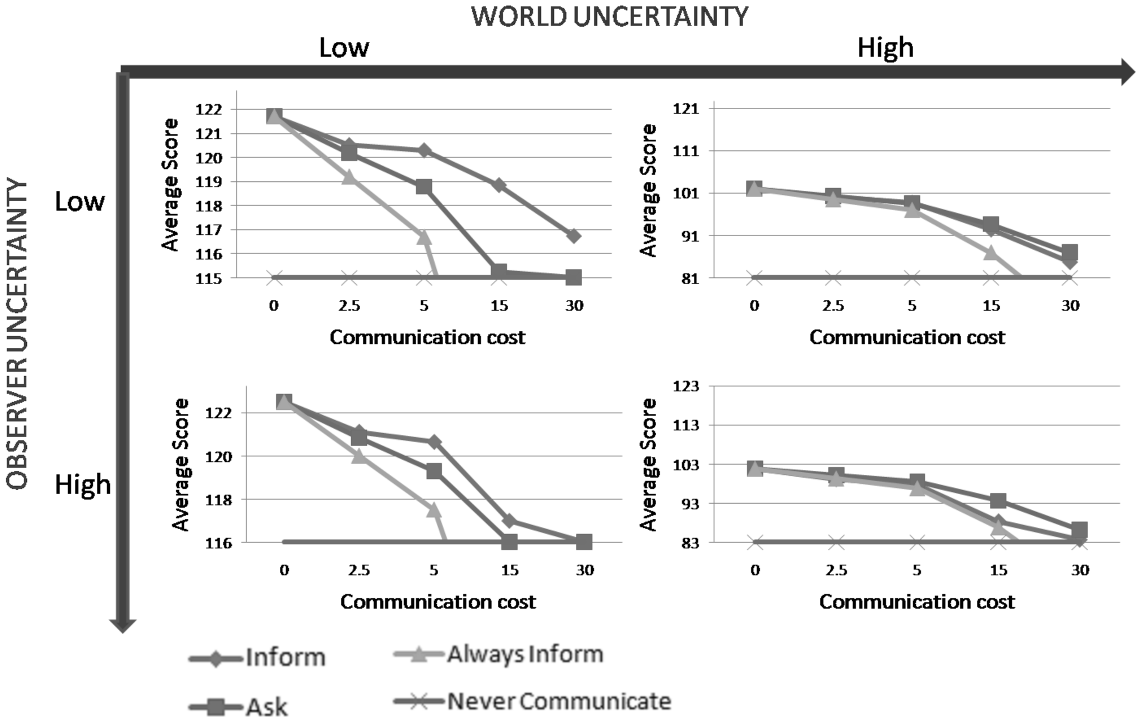


Figure 3.8: Performance of communication protocols by communication cost, observer uncertainty, world uncertainty.

Figure 3.8 shows the average performance of the different communication protocols. On each graph, the vertical axis represents the average score of the game; the horizontal axis varies the cost of communication from low to high. The four graphs cover four possible configurations of world and observer uncertainties (low-low, low-high, high-low, high-high). As shown in the figure, the decision-theoretic protocols (Ask and Inform) outperform or perform as well as the AI and NC protocols for all communication costs and uncertainty levels. When the observer has a good model of its partner (observer uncertainty is low), the Inform protocol performs better than (or equally as good as) the other communication protocols because the observer gets to see the (sometimes unexpected) changes in the world and is good at predicting when its observations are useful for its partner. Interestingly, when world uncertainty is high, the partner expects the world to change frequently and benefits from asking the observer about traps; therefore the Ask protocol performs better or equivalent to other protocols. However, when changes occur in the world and the world uncertainty is low, the Inform protocol is better. Overall, the decision-theoretic protocols outperform axiomatic models (i.e., non-decision theoretic models without probabilistic representation). The performance of the two decision-theoretic models varies with the uncertainty conditions.

So far it is assumed that the relative communication costs of Ask and Inform protocols are identical. However, the cost of the Ask protocol may be higher than the Inform protocol because the Ask protocol includes two steps of communication; from the partner to the observer and from the observer to the partner. Figure 3.9 shows the average performance of the Ask and Inform protocols given that the relative cost of Ask with respect to Inform varies from 1.0 (identical) to 2.0 (double the communication cost).

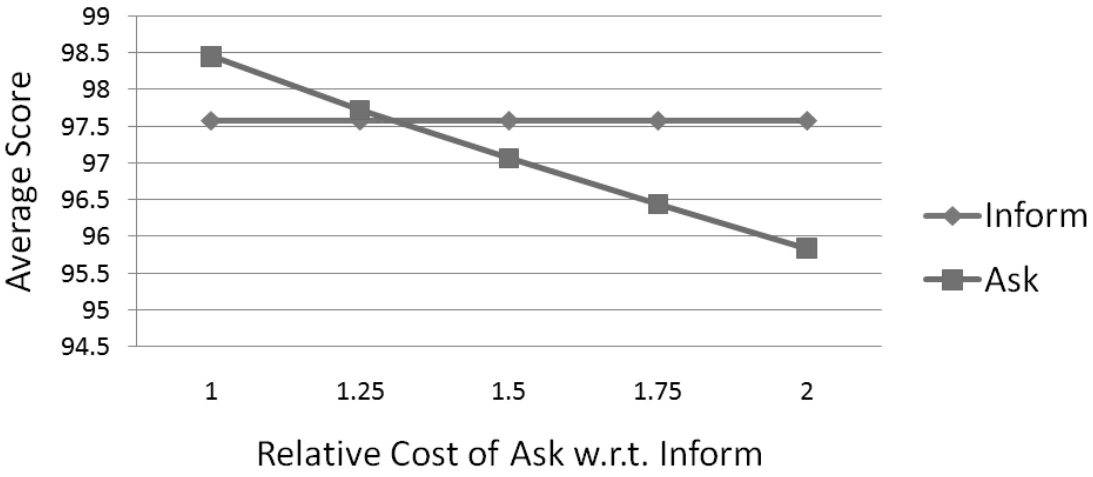


Figure 3.9: Performance of Inform and Ask protocols as relative cost of Ask w.r.t Inform varies (high observer and world uncertainty, communication cost is 5).

Chapter 4

Exploiting Near-Decomposability in Decentralized Decision-Making

Participants of a collaborative activity work together towards satisfying a joint commitment, but they also adopt and care about their individual goals. To be successful in such settings, agents need to reason about accomplishing their individual goals and the effects of their actions on the collaboration, as well as when and how to act together for the sake of the joint commitment.

This chapter identifies special structural characteristics that arise in settings in which computer agents and people collaborate. Consequently, it introduces a new approach for efficient collaborative decision making that reflects these characteristics. The proposed approach is in the spirit of nearly-decomposable models (Simon, 1962) and focuses on collaborative domains in which multiple agents (and people) have individual tasks that they need to achieve, but they interact occasionally to support each other. A new MDP model, Nearly-Decomposable MDP (ND-MDP), is presented to formally define structural characteristics of such collaborative domains. The ND-DECOP algorithm presented in this chapter for solving ND-MDPs exploits the structure of the model for efficient decision-making. The algorithm distinguishes the set of individual and joint actions, thus reduces the search space for joint policies and generates optimal joint plans for multiple agents.

This chapter is organized as follows: Section 4.1 presents Nearly-Decomposable MDPs (ND-MDPs). Section 4.2 introduces the ND-DECOP algorithm for solving ND-MDPs. The chapter concludes with an analysis of the ND-DECOP algorithm.

4.1 Nearly-Decomposable Decision-Making Problems

This section considers a special group of collaborative decision making problems with "nearly-decomposable" structural characteristics. First, it shows that these nearly-decomposable decision-making problems can be modeled using traditional approaches for multi-agent decision-making. Next, a novel approach is introduced for modeling these problems that makes explicit the interdependencies between individual and joint actions of each agent.

4.1.1 Background on Decentralized MDPs

A Decentralized Markov Decision Process (Dec-MDP) is a formalism for multi-agent planning that models collaboration among agents and captures the uncertain and partially observable nature of the real-world. A Dec-MDP includes a set of states with associated transition probabilities, a set of actions and observations for each agent and a joint reward function (Bernstein et al., 2002). A solution of a Dec-MDP is an optimal joint policy as a mapping from joint states to actions. A Decentralized Markov Decision Process (Dec-MDP) defines a joint decision-making problem as a tuple $\langle \mathbf{I}, \mathbf{S}, \mathbf{A}_1, \mathbf{A}_2, \mathbf{T}, \mathbf{R}, \rangle$, where

- $\mathbf{I}$ is a set of agents.
- $\mathbf{S} = \mathbf{S_1} \times \mathbf{S_2}$ is a set of states. S_i indicates the set of states of agent i. A DEC-MDP has a finite horizon H, and each state $s_i^h \in S_i$ is associated with a time step $h \leq H$.
- $\mathbf{A}_1$ and $\mathbf{A}_2$ are sets of individual actions of agents 1 and 2 respectively.
- $\mathbf{T}$ is a transition function. $T\big(s^{h+1} \mid s^h, (a_1, a_2)\big)$ is the probability of transitioning from state $s^h = (s_1^h, s_2^h)$ to $s^{h+1} = (s_1^{h+1}, s_2^{h+1})$ given actions $a_1 \in A_1$, $a_2 \in A_2$.
- $\mathbf{R} : \mathbf{S} \to \mathcal{R}$ is a reward function. $R(s^h)$ is the reward obtained by agents for being in state s^h. [1]
- If one or more agents in I are not able to observe the complete state of the world, the Dec-MDP formalization also includes a finite set of observations Ω and an observation function O that represents a probability distribution over Ω. Agents which cannot observe the state completely keeps a belief state, which is a probability distribution over the state space. The original transition function defined over states is replaced with an updated transition function that combines the observation function with a state estimator function. (See Kaelbling et al. (1998) for details of the POMDP formalization.) For simplicity and ease of representation the rest of the formalism is defined over the states and the regular transition function. The formalism can be adapted for agents that are not able to observe the true state by replacing their

[1] It is also possible to make the reward function depend on joint actions. Both forms of representation are equivalent.

state representation with their belief states and the regular transition function with the updated transition function.

A local policy $\pi_1 : \mathbf{S}_1 \longrightarrow \mathbf{A}_1$ for agent 1 is a mapping from an individual state to an action for agent 1 (and similarly for agent 2). The action subscribed by policy π_1 for state s_1^h is denoted as $\pi_1(s_1^h)$. A joint policy $\pi = (\pi_1, \pi_2)$ is a pair consisting of a local policy for each agent. The expected value $V^\pi(s^h)$ for the joint policy is defined as,

$$\begin{aligned} V^\pi(s^h) = V^{(\pi_1,\pi_2)}(s^h) = R(s^h) + \sum_{s^{h+1} \in \mathbf{S}} & T(s^{h+1} \mid s^h, (\pi_1(s_1^h), \pi_2(s_2^h))) \\ & \cdot V^{(\pi_1,\pi_2)}(s^{h+1}) \end{aligned} \tag{4.1}$$

An optimal joint policy $\pi^*(s^h) = (\pi_1^*(s_1^h), \pi_2^*(s_2^h))$ maximizes Equation 4.1.

An optimal policy for a Dec-MDP determines when it's beneficial for agents to act individually or jointly. It initiates coordination among agents if doing so improves the collaborative utility. However, the complexity of finding optimal solutions to Dec-MDPs is proven to be NEXP-complete by Bernstein et al. (2002), thus Dec-MDPs are often not feasible to model real-world problems.

4.1.2 Nearly-Decomposable MDPs

Due to the infeasibility of solving general multi-agent decision-making problems, this section focuses on a special subset that can represent structural characteristics of many domains in which computer agents and people work together: When participants of a collaborative activity are not interacting, they are performing their individual tasks, and each participant only needs to consider its individual performance. In this case, participants' tasks are essentially independent. They need to reason about their joint value only to determine when and how to coordinate, to communicate with each other and to support each other. Thus, such decision-making problems are nearly-decomposable into individual problems, have some special properties that enable efficient reasoning and can be modeled with a special class of Dec-MDP called Nearly-Decomposable MDP.

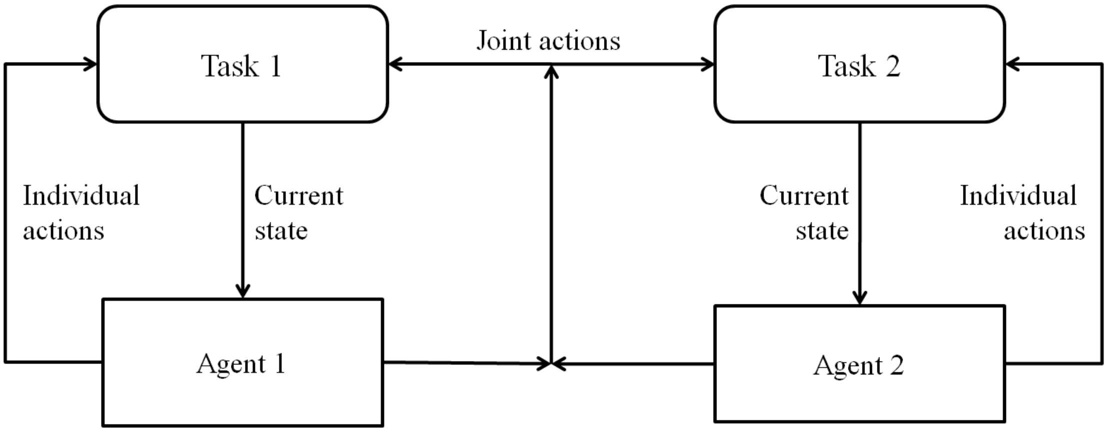


Figure 4.1: An example of nearly-decomposable decision-making.

Figure 4.1 represents the main structural characteristics of nearly-decomposable decision-making problems. Two agents involved in a collaborative activity have individual tasks that they are responsible for but aim at maximizing the joint performance. Agent 1 and agent 2 have two sets of individual actions that they can perform for making progress on Task 1 and Task 2 respectively. An agent chooses which individual action to take individually based on the current state of its task. However, agents' decisions about acting jointly need to be made in coordination by taking both agents' states into account.

An example of nearly-decomposable decision-making problems is managing interruptions among computer agents and people. Consider a collaboration between a computer assistant responsible for meeting scheduling and a user working on a report. The computer agent aims at generating the best possible schedule, and the user is focusing on writing the report as well as she can. Both the assistant and the user benefit from accomplishing both tasks successfully; thus they share a joint utility function. If they are not interacting, the agent's actions are independent of the report the user is working on. Similarly the user does not reason about the meetings scheduled by the agent. However, to determine when and how to interrupt the user to ask for feedback or to acquire user's preferences, the agent needs to consider the cognitive state of the user, whether the user is in the middle of a paragraph or talking on the phone. Both tasks are affected by the timing, content and duration of the interactions.

Nearly-decomposable decision-making problems represent the characteristics of multi-agent planning situations in which agents' decision-making problems are highly decoupled into individual problems, but agents can affect each other's progress by performing a set of joint actions such as communication. These problems are computationally simpler than multi-agent planning problems with highly coupled structure. For example, two Mars rovers with a joint goal of lifting a rock need to reason about the way their joint actions affect the joint state at each time step, and they need to coordinate their actions continuously. Such highly coupled problems are modeled as Dec-MDPs, and solving them takes doubly exponential time in practice. On the other hand, solving problems with completely decoupled structure is computationally simpler than nearly-decomposable problems. For example, if an assistant and a user can act completely independently and are not allowed to communicate, they do not need to reason about each other's task. Solving such completely decoupled problems is as easy as solving a single agent planning problem. A variety of multi-agent systems problems can be viewed as falling between highly-coupled and completely decoupled problems. The techniques of this chapter are aimed at such problems.

A Nearly-Decomposable Markov Decision Process (ND-MDP) is a Dec-MDP in which the state space $\mathbf{S}$ can be factored into individual state spaces $\mathbf{S}_1, \mathbf{S}_2$ for agents 1 and 2 such that it satisfies the following conditions:

1. there exists an individual transition function for agent 1, $T_1\big(s_1^{h+1} \mid s_1^h, (a_1, a_2)\big)$, that represents the probability of reaching an individual state s_1^{h+1} given current state s_1^h and joint action pair (a_1, a_2) (and similarly for agent 2),

2. there exists an individual reward function for agent 1, $R_1(s_1^h)$, mapping state s_1^h and action a_1 to a real number (and similarly for agent 2),

3. the joint transition function T can be represented as the product of agents' individual transition functions,

$$T\big((s_1^{h+1}, s_2^{h+1}) \mid (s_1^h, s_2^h), (a_1, a_2)\big) = T_1\big(s_1^{h+1} \mid s_1^h, (a_1, a_2)\big) \cdot \\ T_2\big(s_2^{h+1} \mid s_2^h, (a_1, a_2)\big) \tag{4.2}$$

4. the joint reward function R can be represented as the aggregate of individual reward functions R_1 and R_2,

$$R(s_1^h, s_2^h) = R_1(s_1^h) + R_2(s_2^h) \tag{4.3}$$

5. action set $\mathbf{A}_1$ for agent 1 can be factored into a set of independent actions $\mathbf{A}_1^I$ and joint actions $\mathbf{A}_1^J$ (and similarly for agent 2). Both agents coordinate to perform the joint actions. When both agents are acting jointly, $a_1 \in \mathbf{A}_1^J$ and $a_2 \in \mathbf{A}_2^J$, the transition function for each agent depends on the other agent's action. However, when both agents are acting independently, $a_1 \in \mathbf{A}_1^I$ and $a_2 \in \mathbf{A}_2^I$. In this case, the transition function for each agent is independent of the other's action.

$$T_1(s_1^{h+1} \mid s_1^h, (a_1, a_2)) = T_1(s_1^{h+1} \mid s_1^h, a_1) \qquad \text{if } a_1 \in \mathbf{A}_1^I, a_2 \in \mathbf{A}_2^I \tag{4.4}$$

(and similarly for agent 2)

6. the set of joint actions for the group, A^J, is the Cartesian product of A_1^J and A_2^J, containing all pairs of joint actions agents can perform in coordination.

7. there exists an individual observation function for agent 1, $O_1(o_1^h \mid s_1^h, (a_1, a_2))$, that represents the probability of observation o_1^h given state s_1^h and joint actions (a_1, a_2) if agent 1 is not able to observe the complete state of the world. When both agents are acting independently, function O_1 is independent of the action selected by action 2.

$$O_1(o_1^h \mid s_1^h, (a_1, a_2)) = O_1(o_1^h \mid s_1^h, a_1) \qquad \text{if } a_1 \in \mathbf{A}_1^I, a_2 \in \mathbf{A}_2^I \tag{4.5}$$

A joint policy for an ND-MDP is a mapping from joint states to joint actions.

Theorem 4.1.1. *Let π be a joint policy of both agents in an ND-MDP, and let π_1 and π_2 be the individual policies for agent 1 and 2, respectively. The following holds:*

- *The value of joint policy π of Equation 4.1 is the aggregate of the values of the individual policies for agents 1 and 2:*

$$V^\pi(s_1^h, s_2^h) = V_1^\pi(s_1^h, s_2^h) + V_2^\pi(s_1^h, s_2^h) \tag{4.6}$$

where the value of agent 1 choosing policy π in state $s^h = (s_1^h, s_2^h)$ is

$$V_1^\pi(s_1^h, s_2^h) = R_1(s_1^h) + \sum_{(s_1^{h+1}, s_2^{h+1})} T\big((s_1^{h+1}, s_2^{h+1}) \mid (s_1^h, s_2^h), \pi(s_1^h, s_2^h)\big) \cdot V_1^\pi(s_1^{h+1}, s_2^{h+1}) \tag{4.7}$$

(and similarly for agent 2).

- *An optimal policy for initial states (s_1^0, s_2^0) maximizes Equation 4.6 as given below,*

$$\pi^*(s_1^0, s_2^0) = \arg\max_{(a_1,a_2)}(R_1(s_1^0) + R_2(s_2^0) + \begin{cases} \sum_{s_1^1,s_2^1} T_1\big(s_1^1 \mid s_1^0, (a_1,a_2)\big) \cdot T_2(s_2^1 \mid s_2^0, (a_1,a_2)) \\ \quad \cdot(V_1^{\pi^*}(s_1^1, s_2^1) + V_2^{\pi^*}(s_1^1, s_2^1))) & \textit{if } (a_1,a_2) \in A^J \\ \\ \sum_{s_1^1,s_2^1} T_1(s_1^1 \mid s_1^0, a_1) \cdot T_2(s_2^1 \mid s_2^0, a_2) \\ \quad \cdot(V_1^{\pi^*}(s_1^1, s_2^1) + V_2^{\pi^*}(s_1^1, s_2^1))) & \textit{if } (a_1,a_2) \in A^I \end{cases} \tag{4.8}$$

The proof of Theorem 4.1.1 is presented in Appendix A.

For simplicity, the value function for an ND-MDP is defined as given in Equation 4.7 specific for situations in which both agent models are fully observable. This function can be revised to model nearly-decomposable decision-making problems in which agents are not able to observe the complete state of the world by incorporating agents' belief states and their observation functions. Thus, the ND-MDP formalism is able to model decision-making problems with agents that cannot fully observe the state of the world.

4.2 Solving Nearly-Decomposable MDPs

This section presents an algorithm that builds on the independence relationships represented in ND-MDPs in order to solve them more efficiently.

4.2.1 ND-DECOP Algorithm

ND-DECOP, decoupling algorithm for solving Nearly-Decomposable MDPs, decouples an ND-MDP into individual models that are connected though a type sequence for coordinating joint actions. There are two building blocks to this algorithm, a *query* model and a *coordination* model. The query model is used to compute the value of a sequence of individual and joint actions to a single agent. The coordination model uses the query model to capture the joint value to both agents of a given sequence of individual and joint actions and to find the sequence of actions that maximizes the joint value.

A is the expanded set of actions including agents' individual actions and their joint action pairs as given below,

$$A = A_1^I \cup A_2^I \cup A^J \tag{4.9}$$

$t(a^i)$ denotes an action *type* for $a^i \in A$. The type of action a^i is constant I if $a^i \in \mathbf{A}_j^I$ (a^i is an individual action) and is constant J_i if $a^i \in \mathbf{A^J}$ (a^i is a joint action).

$$t(a^i) = \begin{cases} I & \text{if } a^i \in \mathbf{A}_j^I \\ J_i & \text{if } a^i \in \mathbf{A}^J \end{cases}$$

t^k denotes the action type agents agree on performing at time step k. A *type sequence* $\mathbf{C}^{h,H-1} = \{t^h, t^{h+1}, \ldots, t^{H-1}\}$ represents a sequence of action types agreed on for time steps h through $H-1$ to reach a final state at time H.

Definition 4.2.1. A query model representing the decision-making problem of agent 1 is an expanded MDP defined by the tuple $\langle \mathbf{S}_q, \mathbf{A}_q, \mathbf{T}_q, \mathbf{R}_q \rangle$, where

- $\mathbf{S}_q$ is a finite set of states that are derived from $\mathbf{S}_1$ by appending to each $s_1^h \in \mathbf{S}_1^h$ a possible type sequence between horizon h and $H-1$. For all $s_1^h \in \mathbf{S}_1$ and $\mathbf{C}^{h,H-1} \in \{I, J_1, \cdots, J_{|A^J|}\}^{H-1-h}$, there exists a state $s_q^h = s_1^h \cup \mathbf{C}^{h,H-1}$. In particular, s_q^h represents a state of the world at time h, such that for any time $h \leqslant n \leqslant H$, the type of action taken by agent 1 agree with value $t^n(a_1) \in \mathbf{C}^{h,H-1}$. In other words, future actions of agent 1 have to agree with the action type sequence. Each state of the original MDP model maps to exponentially many states in the query model.

- a set of actions, A_q, is the combination of the individual actions for agent 1 and the set of joint actions. A_q is defined as $A_q = A_1^I \cup A^J$.
- a transition function, T_q, is defined as $T_q(s_q^{h+1} \mid s_q^h, a_q) = T_1(s_1^{h+1} \mid s_1^h, a_q)$, where $a_q \in A_q$, $s_q^{h+1} = s_1^{h+1} \cup \mathbf{C}^{h+1,H-1}$ and $C^{h+1,H-1} = C^{h,H-1} \backslash \{t^h\}$.
- a reward function, R_q, is defined as $R_q(s_q^h) = R_1(s_1^h)$.

If the model representing agent 1's decision-making problem is a Partially Observable MDP (POMDP), a query model for agent 1 is an expanded POMDP defined by the tuple $\langle \mathbf{S}_q, \mathbf{A}_q, \mathbf{T}_q, \mathbf{R}_q, \mathbf{\Omega}_q, \mathbf{O}_q \rangle$, where

- S_q, A_q, T_q and R_q are defined as above.
- Ω_q is defined as $\Omega_q = \Omega_1$.
- the observation function is defined as $O_q(o_q^h \mid s_q^h, a_q) = O_1(o_1^h \mid s_1^h, a_q)$, where $a_q \in A_q$.

An optimal policy for the query model, π_q^*, maximizes the value function of the query model as given in Equation 4.10. $V^{\pi_q^*}(s_q^h)$ denotes the optimal value of agent 1 for state s_q^h given that policy π_q^* agree with type sequence $\mathbf{C}^{h,H-1} = \{t^h, \ldots, t^{H-1}\}$.

$$\pi_q^*(s_q^h) = \operatorname*{arg\,max}_{a_q : t(a_q) = t^h} \Sigma_{s_q^{h+1}} T_q(s_q^{h+1} \mid s_q^h, a_q) \cdot V^{\pi_q^*}(s_q^{h+1}) \tag{4.10}$$

$$V^{\pi_q^*}(s_q^h) = \max_{a_q : t(a_q) = t^h} [R_q(s_q^h) + \Sigma_{s_q^{h+1}} T_q(s_q^{h+1} \mid s_q^h, a_q) \cdot V^{\pi_q^*}(s_q^{h+1})] \tag{4.11}$$

If agent 1's decision-making problem is modeled as a POMDP, Equation 4.10 is defined over agent 1's belief state, and T_q incorporates O_q and a state estimator function for agent 1.

A complete, optimal policy for the query model specifies the best action to follow for each state and for each possible type sequence. Value functions computed by the query model are used by the coordination model to construct a joint value function.

Definition 4.2.2. A coordination model representing the decision-making problem of agent 2 is an extended MDP defined by the tuple $\langle \mathbf{S}_c, \mathbf{A}_c, \mathbf{T}_c, \mathbf{R}_c \rangle$, where

- S_c is a finite set of states derived from S_2. For each state $s_2^h \in S_2$, there exists a coordination state $s_c^h = s_2^h \cup C^{0,h-1} \cup s_1^o$, which is expanded from s_2^h with a type sequence $C^{0,h-1}$ that keeps the history of actions that led to this state from the initial state s_c^0. The initial state of agent 1 is also included into the state representation as a reference to the query model.

- a set of actions, A_c, is the combination of the individual actions for agent 2 and the set of joint actions. It is defined as $A_c = A_2^I \cup A^J$.

- a transition function, T_c, is defined as $T_c(s_c^{h+1} \mid s_c^h, a_c) = T_2(s_2^{h+1} \mid s_2^h, a_c)$, where $a_c \in A_c$, $s_c^{h+1} = s_2^{h+1} \cup C^{0,h} \cup s_1^0$ and $C^{0,h} = C^{0,h-1} \cup \{t(a_c)\}$.

- a reward function, R_c, is defined as $R_c(s_c^h) = R_2(s_2^h)$, where $s_c^h = s_2^h \cup C^{1,h}$.

If agent 2's decision-making is modeled as a Partially Observable MDP (POMDP), the coordination model for agent 2 is an expanded POMDP defined by the tuple
$\langle \mathbf{S}_c, \mathbf{A}_c, \mathbf{T}_c, \mathbf{R}_c, \mathbf{\Omega}_c, \mathbf{O}_c \rangle$, where

- S_c, A_c, T_c and R_c are defined as above.
- Ω_c is defined as $\Omega_c = \Omega_2$.
- the observation function is defined as $O_c(o_c^h \mid s_c^h, a_c) = O_2(o_2^h \mid s_2^h, a_c)$, where $a_c \in A_q$ and $s_c^h = s_2^h \cup C^{1,h}$.

A value function of the coordination model is denoted as $V^{\pi_c^*}(s_c^h)$. It incorporates the value function of the query model into the coordination model when policy search reaches to the horizon and propagates the joint values of the agents up to the initial state.

Theorem 4.2.3. *The policy π_c^* maximizing the ND-DECOP algorithm value function $V^{\pi_c^*}$ is an optimal policy for a given ND-MDP.*

$$V^{\pi_c^*}(s_c^h) = \begin{cases} \max_{a \in A_c}(R_c(s_c^h) & \\ \quad +\Sigma_{s_c^{h+1}} T_c(s_c^{h+1} | s_c^h, a) \cdot V^{\pi_c^*}(s_c^{h+1})) & \textit{if } h<H \\ R_c(s_c^h) + V^{\pi_q^*}(s_1^o \cup C^{0,H-1}) & \textit{if } s_c^h = s_2^H \cup C^{0,H-1} \cup s_1^o \end{cases} \tag{4.12}$$

$$\pi_c^*(s_c^h) = \arg\max_{a \in A_c}(\Sigma_{s_c^h} T_c(s_c^{h+1} | s_c^h, a) \cdot V^{\pi_c^*}(s_c^{h+1})) \tag{4.13}$$

The proof of Theorem 4.2.3 is presented in Appendix A.

If agent 2's decision-making is modeled as a POMDP, Equation 4.12 is defined over the belief state of agent 2, and T_c incorporates O_c and a state estimator function for agent 2.

C^*, an optimal type sequence computed by the coordination model, specifies when and how (i.e., by taking which joint action) agents should coordinate and when they should act individually. For cases in which agents are acting individually, the policy generated by the coordination model already includes the individual actions that agent 2 should perform.

The policy for agent 1's individual actions is retrieved from the query model with query $\pi_q^*(s_1^o \cup C^*)$.

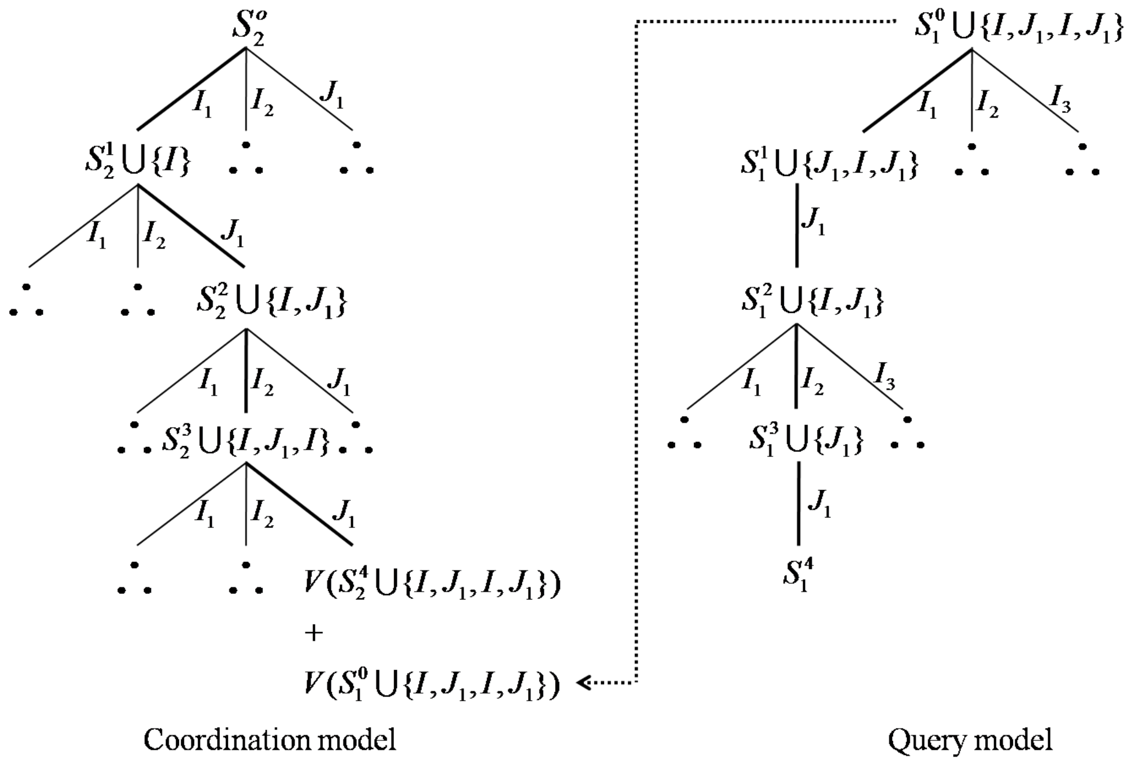

S_2^o
I_1
I_2
J_1
$S_2^1 \cup \{I\}$
$S_2^2 \cup \{I, J_1\}$
$S_2^3 \cup \{I, J_1, I\}$
$V(S_2^4 \cup \{I, J_1, I, J_1\})$
+
$V(S_1^0 \cup \{I, J_1, I, J_1\})$
Coordination model
$S_1^0 \cup \{I, J_1, I, J_1\}$
I_3
$S_1^1 \cup \{J_1, I, J_1\}$
$S_1^2 \cup \{I, J_1\}$
$S_1^3 \cup \{J_1\}$
S_1^4
Query model

Figure 4.2: ND-DECOP Algorithm.

The coordination model unifies value functions of both agents for each possible type sequence so that the type sequence selected by the coordination model maximizes the collaborative value of agents. The ND-DECOP algorithm generates an optimal joint policy for an ND-MDP that maximizes the joint value of agents.

Figure 4.2 illustrates the steps the ND-DECOP Algorithm. The query model computes the value of agent 1 for all sequences of joint and individual actions (i.e. type sequences). For all such sequences, the coordination model incorporates agent 1's value for that particular sequence into agent 2's corresponding value. The coordination model selects the type sequence that maximizes the joint value. In the example given, two and three individual actions are available for agent 2 and 1 respectively, and agents can coordinate on a single joint action.

4.2.2 Complexity Analysis of the Algorithm

The ND-DECOP algorithm is a two step process in which a query model solves one of the individual agent models for each possible type sequence, and a coordination model incorporates the values that are computed by the query model into its own value function to generate an optimal type sequence. Let Ω be the complexity of solving the agent model represented by the query model, Θ be the complexity of solving the agent model represented by the coordination model, $|A^J| = k$ be the number of joint actions, and H be the horizon, the complexity of finding an optimal policy with the ND-DECOP algorithm is $(k+1)^H \times \Omega + \Theta$.

Generating a complete policy for an ND-MDP with the ND-DECOP algorithm requires solving the agent model represented with the query model exponentially many times and solving the agent model represented as the coordination model only once. Thus, computational complexity of the ND-DECOP Algorithm is determined by the complexity of solving the agent model represented with the query model. Representing the agent model with lower computational complexity as the query model and solving it exponentially many times reduces the computational complexity of the ND-DECOP Algorithm.

If the agent model represented with the query model is a fully observable MDP and thus can be solved in polynomial time, the complexity of the ND-DECOP algorithm is exponential in the horizon. In such cases, the ND-DECOP algorithm is able to achieve exponential savings in computation time for solving ND-MDPs in practice as compared to general DEC-MDP approaches. However, if both agent models are partially observable and cannot be solved in polynomial time, running time of the ND-DECOP algorithm can be as high as doubly exponential.

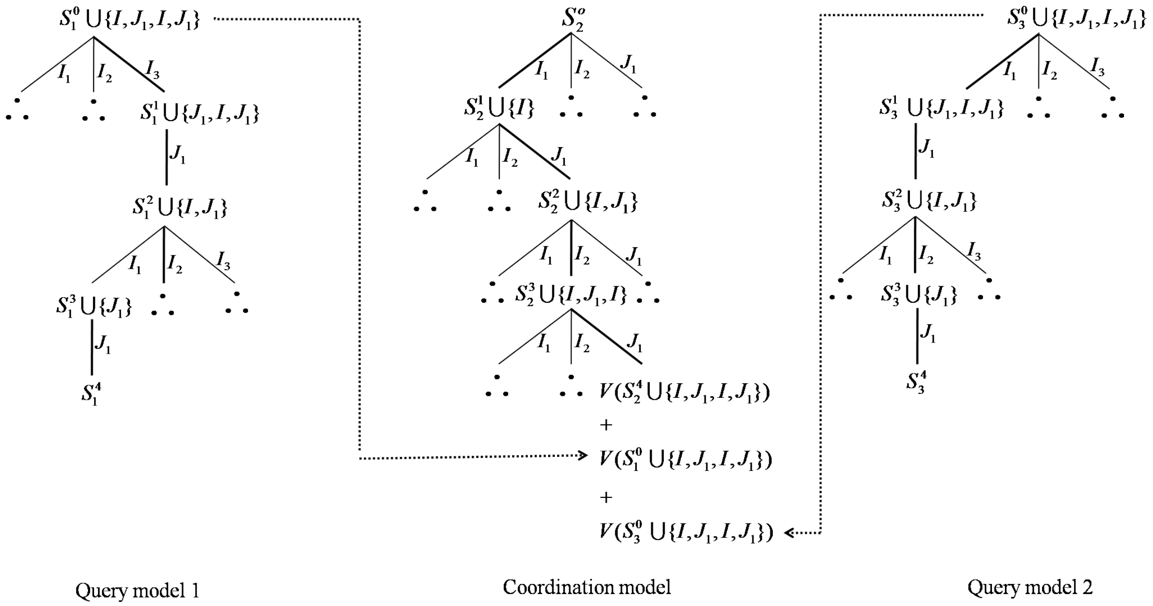


Figure 4.3: Solving ND-MDP for one-to-many interactions.

For simplicity, the ND-MDP model is formalized for managing one-to-one interactions between two agents. The ND-MDP model and its nice computational properties can be easily generalized to model one-to-many interactions of agent groups of size larger than two. These one-to-many interactions may occur between a single user and multiple computer processes that demand the attention of the user, or a computer process that may request information from multiple users that it is connected to. one-to-many interactions can be considered as a resource sharing problem in which a single agent is a resource that multiple agents share by coordinating multiple joint actions available for them (each joint action may represent a communication opportunity with a different agent.) (See Figure 4.3).

In an ND-MDP for modeling one-to-many interactions, the coordination model represents a resource agent. An individual query model is generated for each partner agent that can jointly act with the resource agent. Given that agent i is the resource agent, and agent j is one of the partner agents that can coordinate with agent i, the set of joint actions that the query model for agent j needs to consider is $A^J_{q_j} = A^J_i \times A^J_j$. The set of joint actions for the coordination model is $A^J_c = \bigcup_{j \neq i} A^J_i \times A^J_j$. Thus, the number of joint actions and consequently the branching factor of the coordination model grows linearly with respect to the number of agents the coordination agent can interact with. Figure 4.3 illustrates the way coordination model incorporates multiple query models to determine the optimal type sequence to follow.

The complexity of an NDP-MDP that models one-to-many interactions is bounded by the complexity of solving the most expensive query model and does not grow exponentially in number of agents. When compared with Dec-MDP solution methods that have exponential complexity in number of agents, the ND-DECOP algorithm is significantly more efficient for solving decision-making problems of nearly-decomposable structure with one-to-many interactions.

However, these nice properties of ND-MDPs do not generalize to model many-to-many interactions among agents (i.e. multiple resources are connected to multiple partners). Having multiple coordination models require consistency checks between coordination models to ensure that plans constructed by the models are consistent with each other, which is exponential in the number of coordination models.

Chapter 5

Interruption Management as an Example of Nearly-Decomposable Decision-Making

This chapter presents interruption management as an example of collaborative decision-making capability needed to manage interactions well among computer agents and people. For example, a writer's collaborative assistant that autonomously searches for bibliographical and citation information for a paper that a user is working on needs to know when to ask the user for information and how to time these requests (Babaian et al., 2002). If the assistant continuously asks whether to cite each paper that meets the user's keywords and commands, it will disturb the user's writing process.

This decision-making problem has nearly-decomposable structural characteristics in that (1) when not interacting, both the computer agent and the user have individual tasks that they want to accomplish (the user is writing new sections of the paper and the agent is searching for citation information for previous sections), (2) the individual tasks are independent of each other's performance (sections written by the user are independent of the agent's citation search), (3) when they are acting jointly, joint actions affect the performance of the user and the agent (the agent's search is affected by the additional information obtained from the user, the user's progress in writing the paper is suspended by the interruption).

This chapter presents a game setting that is an abstract analogue of interruption management domains. It introduces models and algorithms for capturing the value of interruption in this setting.

5.1 The Interruption Game

The "interruption game" is designed for investigating the problem of managing interruptions in a setting that does not require sophisticated domain expertise. The game is developed on the Colored Trails (CT) infrastructure which has been widely used as a research test-bed for a variety of decision-making problems (Grosz et al., 2004). It involves two players, referred to as the "person" and the "agent". Each player needs to complete an individual task but the two players' scores depend on each other's performance making this a collaborative endeavor.

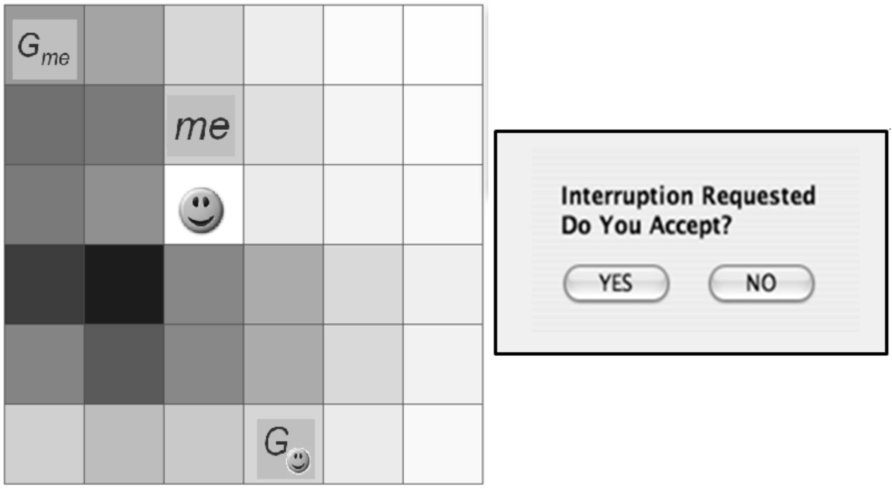
The game is played on a board of 6x6 squares. Each player is allocated a starting position and a goal position on the board. The game comprises a fixed, known number of rounds. At each round, players advance on the board by moving to an adjacent square. The players' goals move stochastically on the board according to a Gaussian probability distribution centered at the current position of the player. [1] The probability of a goal moving to a particular square on the game board is computed as the ratio of the integral of the Gaussian function over the square and the integral of the function over the game board. Players earn 10 points in the game each time they move to the square on which their assigned goal is located, and the goals are reassigned to random positions on the board. Players can see their positions and the goal location of the other player, but they differ in their ability to see their own goal location: The person can see the location of its goal throughout the game, while the agent can see the location of its goal at the onset of the game but not during consecutive rounds.

At any round, the agent can choose to interrupt the person and request the current location of its goal. The person is free to accept or reject an interruption request. If the person rejects the interruption request, the players continue moving. If the interruption is accepted by the person player, the location of the agent's goal in the current round (but not in consecutive rounds) is automatically revealed to the agent. There is a joint cost for revealing this information to the agent in that both participants will not be able to move for one round. The game scenarios used in the empirical evaluation with human subjects are simplified to allow a single interruption request through the game.

The rules of the game also require the agent to provide the person with its belief about the location of its goal. This information may influence the person's decision about whether to accept an interruption. A snapshot of the game from the perspective of the person player is shown in Figure 5.1. *me* icon on the board represents the person player, *smiley* icon represents the agent player, G_{me} represents the person's goal, G_{smiley} represents the agent's goal. The degree to which each square is shaded represents the agent's uncertainty about

[1] The movement of the goal is restricted in that it does not move closer to the position of the player.

its goal. Dark squares imply higher certainty.



Interruption Requested
Do You Accept?

Figure 5.1: Screen-shot of the interruption game.

The rules of the game provide incentives to players for reaching to their goals as quickly as possible, and interruptions initiated by the agent are critical determinants of players' performance. The agent's uncertainty about the location of its own goal increases over time, and its performance depends on successfully querying the person and obtaining the correct position of its goal.

The game is collaborative in that the score for each player depends on a combination of its own performance and the performance of the other player. The players share a joint scoring function that is the cumulative score of both players. An interruption is potentially beneficial for the individual performance of the agent, who can use this information to direct its movement, but it only affects the person's performance negatively. Sharing this information is costly for both players. When the agent deliberates about whether to ask for information, or when the person deliberates about whether to reveal the information to the agent, the players need to weigh the potential benefit to the agent player with the detriments to their individual performance in the game. The success of both players in the game depends on the agent's ability to estimate the collaborative value of interruption at each point in the game and use that estimate to choose when to interrupt the person.

The interruption game is not meant to be a complete model of any specific domain or application. Its purpose is to provide a simple setting in which to study the factors that influence interruption management in collaborative settings. It provides a setting that is analogous to the types of interactions that occur in collaborative settings involving a mixed network of computer agents and people. For example, the person player in the interruption game may represent a user of a collaborative system for writing an academic paper, and the agent may represent a collaborative assistant responsible for obtaining bibliographical data. While both of the participants share a common goal of completing a document, each of them must work independently to complete its individual task such as composing paragraphs or searching for bibliographical information. This aspect is represented in the interruption game by assigning an individual goal for each player. The stochastic movement of these goals on the board corresponds to the dynamic nature of these tasks. For example, the user may not know what to write next, and the system may have uncertainty about search results. The agent's lack of information about its own goal location in the game corresponds to the uncertainty of a system about the preferences and intentions of its user such as which bibliographical information to include in the paper. The ability to query the user for keywords and to choose among different bibliographies provides the system with valuable guidance and direction. It may, however, impede the performance of both participants on their individual tasks, because the system needs to suspend its search for bibliographical data when it queries the user, and the user may be distracted by the query.

This dynamic cost of interruption represents the costs incurred to both users and computer agents due to task switching and task recovery for initiating and responding to interruptions.

5.2 Estimating Value of Interruption with Nearly-Decomposable Models

Managing interruptions in the interruption game is a multi-agent planning problem. The players of the game share a joint reward function. Interruption is a joint action which requires both players to agree on at the same time. The interruption action affects the states both players transition to, and it affects the performance of both players. To determine whether an interruption is beneficial for the collaboration, the players need to reason about the way the interruption affects the progress of the players in the game. The players have different perspectives about the world when they are making decisions; the person player can observe the location of the agent player's goal but the agent player cannot.

Kamar and Grosz (2007) have shown that the interruption game can be modeled as a Decentralized Markov Decision Process (Dec-MDP). To model the interruption game, the state space of the Dec-MDP combines all of the information relating to the tasks of both players, including their positions on the board, the positions of their goals, the current round and the belief of the agent about its own goal position. The solution of the Dec-MDP assigns a policy to the agent that initiates interruption requests when they are expected to result in a benefit to both players according to the joint reward function. A policy generated for the person accepts interruption requests that have actual positive benefit. [2] Due to the

[2] The way "the person player" and "the person" used in this section refers to a computer agent that uses decision-theoretic models to generate policies to be used by a fully-rational and computationally unbounded

computational complexity of general solution algorithms for solving Dec-MDPs and the size of the state space, it is infeasible to compute a complete joint policy for both players in the interruption game using general algorithms.

This section exploits the nearly-decomposable structure of the interruption game to devise strategies for computer agents to interact with people in a way that interruptions are beneficial for the collaborative activity and people find these interactions useful. In real-life systems the interaction between computer agents and people may be far from optimal due to the following reasons: (1) the agent may fail to correctly capture the utility of an interruption because of the dynamic and partially observable nature of the environment, (2) a person's estimate of the utility of an interruption may not match a fully rational computational estimate. This section addresses the first problem and provides two methods for computing players' estimates of the benefit of interruptions in the game under the assumptions that the person player is fully rational and computationally unbounded. The DECOP algorithm presented in Section 5.2.1 follows a myopic approach by assuming that only a single interruption request can occur through a game. Section 5.2.2 models the interruption game as a Nearly-Decomposable MDP and uses the ND-DECOP algorithm to capture the value of an interruption accurately even when multiple interactions are allowed among agents. These algorithms are not meant to predict the way people play the interruption game or respond to interruption requests in general. The benefit of interruption as calculated by the proposed algorithms is used as a theoretical, fully-rational baseline to enable empirical investigations of human behavior in mixed human-computer collaborative settings.

person player in the interruption game. The baseline values computed by these models are used to empirically investigate the way human agents play the game.

5.2.1 A Myopic Approach: DECOP Algorithm

Given the complexity of solving the interruption game with general Dec-MDP solution algorithms optimally, this section focuses on a more constrained problem: Instead of exhaustively computing optimal policies in the interruption game, this section focuses on generating interruptions when they are perceived to be beneficial to the collaboration. Such interruptions are hypothesized to be likely to get accepted by people. The myopic decision-making algorithm presented here, called DECOP, exploits the nearly-decomposable nature of the interruption game in that when players are not making or replying to interruption requests, they are performing their individual tasks, and each player needs to consider only its individual score in the game. In this case, the two tasks are essentially independent, and they can be solved separately. The algorithm also makes the assumption that agents can interact at most once. Under this assumption the expected utility of an interruption can be computed efficiently because an interruption request will render the two tasks independent from the interruption moment until the end of the game. At each turn, the policy for the agent is to interrupt and request information from the person when it is deemed beneficial for both participants.

This next section details the DECOP algorithm that captures the benefit of an interruption by solving the individual tasks for both participants in the game and combining these solutions to devise strategies for interruption management.

Computing a Policy for the Person

The person has complete information about the game so its task can be modeled as a Markov Decision Process (MDP). Let B denote the set of board positions; $|B|$ denotes the size of the game board; $p \in B$, $g \in B$ are the positions of the person and its goal respectively; $m \in A$ is a movement action of the player; $P(g' \mid p, g)$ is the probability of the goal position moving from position g to position g' when the player is in position p. The state space of the MDP includes every possible position of the person and its goal at each round. $S_P^h = \langle p, g \rangle$ represents the state at round h. Transition function T assigns a probability for reaching state S_P^{h+1} from S_P^h given action m. T can be directly derived from P. The reward function R assigns the score in the game for reaching to the goal if an action transitions a player to its goal square and 0 otherwise.

Let Π_P^* denote an optimal policy for the person player in the game. The value of this policy at state S_P^h, $V^{\Pi_P^*}(S_P^h)$, maximizes the reward at state S_P^h and at future states given the transition probability function,

$$V^{\Pi_P^*}(S_P^h) = max_m[R(S_P^h) + \sum_{S_P^{h+1}} T(S_P^{h+1} \mid S_P^h, m) \cdot V^{\Pi_P^*}(S_P^{h+1})] \tag{5.1}$$

An optimal policy and its value are computed using ExpectiMax search. In this process a tree is grown with two types of nodes, decision nodes and chance nodes. There is a decision node for each state in the MDP, and each child of a decision node is labeled with a movement action for the person. Chance nodes represent moves of nature, and each child of a chance node is labeled with a possible movement of the goal of the person and is assigned a probability according to the transition function. When traversing the tree, the value of each chance node is recursively computed as weighted average of the value of each of its children according to its probability. The value of each decision node is computed by choosing the child with the maximal value. With memoization, the number of nodes generated by the search is bounded by $|B|^2 \cdot |H|$, which is polynomial in the number of rounds in the game.

Computing a Policy for the Agent

The agent cannot observe the position of its goal on the board, and without interrupting the person it receives no information relating to this position. The agent's task is modeled as a No Observation Markov Decision Process (NOMDP), which is a special case of an MDP with no observations. The state space for this model includes the position l of the agent on the board, its belief $b \in \Delta B$ over its goal position and current turn h. The state for the agent is denoted as $S_A^h =< l, b >$. As what is modeled is the agent's individual task, rather than its interaction with the person, the interruption action is left out. The set of actions A and reward function R for the agent are identical to the ones described for the person player. The agent updates its belief b to $b^{'}$ after each turn according to the goal movement distribution P as follows:

$$\forall c' \in B, b'(c') = \sum_{c \in B} b(c) \cdot P(c' \mid l, c) \tag{5.2}$$

The value of Π_A^*, an optimal policy for the agent at state $S_A^h = \langle l, b \rangle$, can be computed using Equation 5.1, substituting Π_A^* for Π_P^* and S_A for S_P. Because the belief of the agent about its goal position is incorporated into the state space, there are an infinite number of states to consider, and using ExpecitMax in a straightforward fashion is not possible. However, applying the belief update function after each turn only a small number of states turn out to be reachable. The deterministic belief update function maps each combination of states with full information (i.e., states in which the agent knows the correct position of its goal) and the number of turns since full information to a single belief state, thus to a single state. As a result, the search tree can be grown "on the fly" and only by only expanding those states that are reachable after each turn. Memoization is not possible in this technique, and thus the complexity of the complete search is exponential in the length of the horizon.

Computing the Benefit of Interruption

To compute the benefit of an interruption, its effect on both the agent's and the person's individual performances must be taken into account. It is the aggregate of these two effects that determines the utility of interruption. An interruption is initiated by the agent, but it is only established if the person accepts it. The effect of an interruption on the individual game play of a player is the difference of the values of two states; one in which an interruption is established and other in which it is not. Given the person and its goal are located on squares p and g respectively in game round h, $EU_P^{NI}(S_P^h = \langle p, g \rangle)$ denotes the expected utility of the person when it is not interrupted and pursues its individual task. This is equal to the value for the person of carrying out its individual task as shown in Equation 5.1.

$$EU_P^{NI}(S_P^h) = V^{\Pi_P^*}(S_P^h) \tag{5.3}$$

For the agent that does not observe its own goal position, $EU_A^{NI}(S_A^h = \langle l, b \rangle)$ denotes the expected utility of the agent when it is not interrupted and pursues its individual task. This is the value to the agent of carrying out its individual task:

$$EU_A^{NI}(S_A^h) = V^{\Pi_A^*}(S_A^h) \tag{5.4}$$

$EU_P^I(S_P^h = \langle p, g \rangle)$ denotes the expected utility of the person when it accepts an interruption. If the person player is interrupted, it cannot move for one round but its goal may move stochastically according to the probability distribution P. The new goal position is denoted as g^{h+1}. Given the game constraint that there can only be one interruption made in the game, the benefit of interruption for the person is the expected value of its individual task in future rounds for any possible position of its goal. Formally, the utility of interruption for state S_P^h, denoted $EU_P^I(S_P^h)$, is computed as follows:

$$EU_P^I(S_P^h) = \sum_{g^{h+1} \in B} P(g^{h+1} \mid p, g) \cdot V^{\Pi_P^*}(S_P^{h+1} = \langle p, g^{h+1} \rangle) \tag{5.5}$$

If the agent successfully interrupts the person, the person will reveal the position of the agent's goal. The agent will update its belief about its goal position in the following round and use this belief to perform its individual task in future rounds. However, when it deliberates about whether to interrupt in the current round, it needs to sum over every possible position of its unobserved goal according to its belief about the goal location. $S_A^h = \langle l, b \rangle$ denotes the current state of the agent including its position on the board and belief about its goal position. g denotes the current position of its goal. The expected value of interruption for the agent is denoted EU_A^I and is computed as follows:

$$EU_A^I(S_A^h) = \sum_{g \in B} b(g) \cdot V^{\Pi_A^*}(S_A^{h+1} = \langle l, b' \rangle) \tag{5.6}$$

Here, b' refers to the belief state of the agent in which probability 1 is given to g, the true position of its goal as revealed by the person, and updated once to reflect the stochastic movement of the goal in turn h.

Deciding Whether to Interrupt

By combining the expected values of the person and agent players with and without interruption, it is now possible to compute the agent's estimate of the benefit of an interruption. $EBI_P(S_P^h)$ represents the expected benefit of interruption for the person, which is simply the difference in utility of the person between accepting and interruption and carrying out its individual task.

$$EBI_P(S_P^h) = EU_P^I(S_P^h) - EU_P^{NI}(S_P^h) \tag{5.7}$$

The expected benefit of interruption for the agent is denoted as $EBI_A(S_A^h)$ and is computed similarly:

$$EBI_A(S_A^h) = EU_A^I(S_A^h) - EU_A^{NI}(S_A^h) \tag{5.8}$$

The interruption game is collaborative in that the combined performance of both participants determines their individual scores. The agent can observe the state S_P^h of the person, and for any combined state $S^h = (S_P^h, S_A^h)$ the agent will consider the joint expected benefit of interruption to both participants, EBI, and choose to interrupt if this joint benefit is positive.

$$EBI(S^h) = EBI_P(S_P^h) + EBI_A(S_A^h) \tag{5.9}$$

The agent cannot observe the correct position of its goal and estimates the benefit of interrupting under this uncertainty. Thus, not every interruption initiated by the agent is truly beneficial for the team. In contrast, the person can observe the position of agent's goal and can capture the actual benefit of the interruption, denoted as ABI, with certainty. Any interruption with positive ABI offers a positive expected benefit to the team. ABI is the sum of the individual benefits of interruption to both the person and the agent computed from the person's perspective.

Let g_a be the agent's goal position, the actual benefit of interruption for both participants given states S_P^h and S_A^h is computed as:

$$ABI(S^h) = EBI_P(S_P^h) + EBI_{P,A}(S_A^h) \tag{5.10}$$

Here, the term $EBI_{P,A}(S_A^h)$ refers to the person's perception of the agent's benefit from revealing the goal position g_a, where l is the current position of the agent, b' refers to the belief state of the agent in which probability 1 is given to g_a and updated once.

$$EBI_{P,A}(S_A^h) = V^{\Pi_A^*}(S_A^{h+1} = \langle l, b' \rangle) - EU_A^{NI}(S_A^h) \tag{5.11}$$

The advantage of the DECOP algorithm is that it reduces the complexity of multi-agent decision making to that of two separate single agent decision making processes. Because the agent is allowed to interrupt only once during a game-play, the decoupling method is able to accurately capture the benefit of an interruption initiated by the agent.

5.2.2 Nearly-Decomposable MDP Approach: ND-DECOP Algorithm

The interruption game is designed to reflect the general characteristics of the collaborative domains in which computer agents and people interact, which include nearly-decomposable structure (See Figure 5.2): (1) the players of the interruption game are fully collaborative, they share a common utility function that is the cumulative reward of both players. (2) the individual rewards of the players do not depend on each other's state. (3) the state that a player transitions to does not depend on each other's state, but it may depend on the joint actions they perform together. A joint policy for the interruption game is a combination of decisions about when to interrupt (i.e., performing joint action $< Interrupt, Accept >$) and decisions about when and how to act individually towards players' goals.

The interruption game can be modeled as a ND-MDP as given below:

- The state space S is factored into two components $\mathbf{S_P} \times \mathbf{S_A}$.

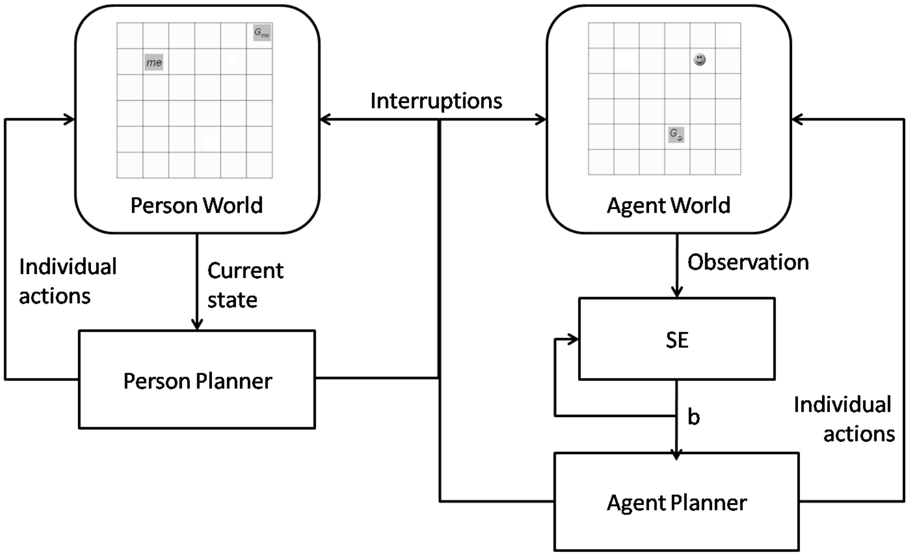


Figure 5.2: Nearly-decomposable structure of the interruption game.

- $\mathbf{S}_P$ is the set of states for the person. A state in S_P is denoted as $\langle p_P^h, g_P^h \rangle$, where $p_P^h, g_P^h \in B$ are the positions of the person and its goal at round h respectively, and B denotes the set of board positions.

- $\mathbf{S}_A$ is the set of states for the agent. A state in S_A is denoted as $\langle p_A^h, b^h \rangle$, where $p_A^h \in B$ is the position of the agent and $b^h \in \Delta B$ is agent's belief over its goal positions at round h. [3] $P(g_A^{h+1} \mid p_A^{h+1}, g_A^h)$ represents the probability of the goal position moving from position g_A^h to g_A^{h+1} when the player is in position p_A^{h+1}. The agent's belief state is updated at each time step according to P to reflect the agent's changing beliefs about the position of its goal. As the number of rounds progress, the uncertainty of the agent about its goal location increases if it does not receive information from the person. The agent updates its belief b^h to b^{h+1} after each turn

[3] If the principle is reasoning about the agent, S_A includes g_A^h, the position of agent's goal at round h.

using the State Estimator (SE) function given below,

$$\forall g_A^{h+1} \in B, b^{h+1}(g_A^{h+1}) = \sum_{g_A^h \in B} b^h(g_A^h) \cdot P(g_A^{h+1} \mid p_A^{h+1}, g_A^h) \tag{5.12}$$

- The set of actions $\mathbf{A}_P$ for the person includes the possible movements on the board and decisions about accepting an interruption request from the agent. The sets of independent actions $\mathbf{A}_P^I, \mathbf{A}_A^I$ for the person and agent are composed of their possible movements on the board. The joint actions for the person $\mathbf{A}_P^J$ are accepting or rejecting an interruption request by the agent, and the joint action for the agent $\mathbf{A}_A^J$ is generating an interruption request. It is not necessary to consider an interruption request from the agent that is rejected by the person as a joint action as it is suboptimal. Thus, A^J includes only a single joint action pair, $< interrupt, accept >$.

- The transition function for the person is denoted as $T_P\big(s_P^{h+1} \mid s_P^h, (a_P^h, a_A^h)\big)$, where $s_P^h, s_P^{h+1} \in \mathbf{S}_P$ represent states at rounds h and $h+1$ for the person; a_P^h represents an action for the person; a_A^h denotes an action for the agent. The transition function is computed separately for acting jointly and independently.

  If the joint action (a_P^h, a_A^h) represents movement actions of the person and agent on the board, then both agents are acting individually. In this case, $a_P^h \in \mathbf{A}_P^I, a_A^h \in \mathbf{A}_A^I$ and by Equation 4.4 the transition function for each participant depends on its own actions, not the actions of the other. For this case:

  - The transition function for the person is

    $$T_P\big(s_P^{h+1} \mid s_P^h, (a_P^h, a_A^h)\big) = P(g_P^{h+1} \mid p_P^{h+1}, g_P^h) \tag{5.13}$$

    where $p_P^{h+1} = p_P^h + a_P^h$, that is, the location of the person and its goal advance from their position in turn h to their position in turn $h+1$.

- The transition function for the agent is

$$T_A\big(s_A^{h+1} \mid s_A^h, (a_P^h, a_A^h)\big) = \begin{cases} 1 & \text{if } p_A^{h+1} = p_A^h + a_A^h,\, b^{h+1} = SE(b^h) \\ 0 & \text{otherwise} \end{cases} \tag{5.14}$$

  where b^{h+1} is correctly updated from b^h using Equation 5.12. [4]

If actions a_P^h,a_A^h are joint actions of the person and agent, then agents are acting jointly. In this case, $a_P \in \mathbf{A}_P^J$ and $a_A \in \mathbf{A}_A^J$. If a_P^h, a_A^h represents an interruption request by the agent that is accepted by the person, then

- the transition function for the person is

$$T_P\big(s_P^{h+1} \mid s_P^h, (a_P^h, a_A^h)\big) = P(g_P^{h+1} \mid p_P^{h+1}, g_P^h) \tag{5.15}$$

  where $p_P^{h+1} = p_P^h$, that is, the person does not move at turn h.

- The transition function for the agent is

$$T_A\big(s_A^{h+1} \mid s_A^h, (a_P^h, a_A^h)\big) == \begin{cases} 1 & \text{if } p_A^{h+1} = p_A^h,\, b^h(g_A^h) = 1,\, b^{h+1} = SE(b^h) \\ 0 & \text{otherwise} \end{cases} \tag{5.16}$$

  that is, the agent does not move at turn h and it has knowledge about its true goal location, and b^{h+1} is correctly updated from b^h using Equation 5.12.

- The reward function is defined as,

$$R(s^h) = R_P(s_P^h) + R_A(s_A^h) \tag{5.17}$$

[4] If s_A^h includes g_A^h, T_A also includes the goal movement function P.

where R_P gets value 10 when the person reaches its goal as shown below,

$$R_P(s_P^h) = \begin{cases} 10 & \text{if } p_P^h = g_P^h \\ 0 & \text{otherwise} \end{cases} \tag{5.18}$$

The reward function $R_A(s_A^h)$ for the agent is similar.

Solving the ND-MDP model for the interruption game yields an optimal joint policy $\pi^* = (\pi_P^*, \pi_A^*)$ which is a tuple composed of local policies for each agent. For the person this policy is a mapping from $\mathbf{S}_P$ to $\mathbf{A}_P$ because its observations fully describe its state. For the agent with incomplete information, the agent's observations about its goal location are incorporated into its belief b using Equation 5.12 and b is a component of the agent's state. Therefore, a policy of the agent is a mapping from $\mathbf{S}_A$ to $\mathbf{A}_\mathbf{A}$. A joint policy for the person and the agent specifies the movement actions on the board and the times in which it is optimal for the agent to interrupt the person.

Solving the Interruption Game with the ND-DECOP Algorithm

As shown in Section 5.2.2, the interruption management problem can be formalized as an ND-MDP, as it has the main characteristics of nearly-decomposable domains. This section demonstrates the way the ND-DECOP algorithm can be used to solve the interruption problem.

Generating an optimal policy with the ND-DECOP algorithm requires solving an MDP corresponding to the query model exponentially many times and solving an MDP corresponding to the coordination model only once. As the query model is the bottleneck of the algorithm, representing the MDP with the lower complexity as the query model helps to reduce the complexity of generating an optimal policy for the interruption problem.

When modeling the interruption game, the person MDP is modeled as the query model and solved exponentially many times, and the more expensive partially observable agent model is modeled as the coordination model and solved only once.

The extended action set for players of the interruption game is composed of four movement actions (i.e., for moving up, down, left and right on the game board) and one joint action pair ($< interrupt, accept >$). The type function $t(a^i)$ maps movement actions to the individual label I and maps $< interrupt, accept >$ to the joint action label J_1. A type sequence for the interruption game consists of I and J_1 labels, as one per time step.

Figure 5.3 shows an example of the ND-DECOP algorithm computing the optimal policy for a given state of the interruption game. (s_A^{H-2}, s_P^{H-2}), for time step $H-2$ (two rounds until the end of the game). The example uses ExpectiMax search to compute policies for the coordination and query models. The initial state of the coordination model, (s_c^{H-2}), is composed of the initial states for the two individual player models and an empty type sequence. Each state branches to five possible next states, and branches are labeled with the action that is leading to the next state. The type of the action leading to the next state is included into the type sequence, to keep track of the history of actions leading to the state. When the coordination model reaches to the leaf nodes of the search tree, it makes a query to the query model with a complete type sequence that leads to a particular leaf. This complete type sequence and the initial state of the person model compose the initial state of the query model (e.g., in the example, the action history of the labeled leaf is $\{left, interrupt\}$, thus the type sequence provided to the query model from this leaf is $C^{H-2,H-1} = \{I, J_1\}$). In the ExpectiMax search tree of the query model, each state only has branches (i.e., actions) that agree with the value of the type sequence for that time step (e.g., for the given example, state s_q^{H-2} branches only with movement actions, since $t^{H-2} = I$). Once the query model builds a complete tree, rewards from all states are propagated from the bottom of the tree to the top. The expected value of the query model for the initial state is provided to the coordination model to be incorporated to the value of the corresponding leaf of the coordination model. After the values collected from the queries are incorporated into the coordination model, actions optimizing the value of the coordination model are chosen as the optimal policy.

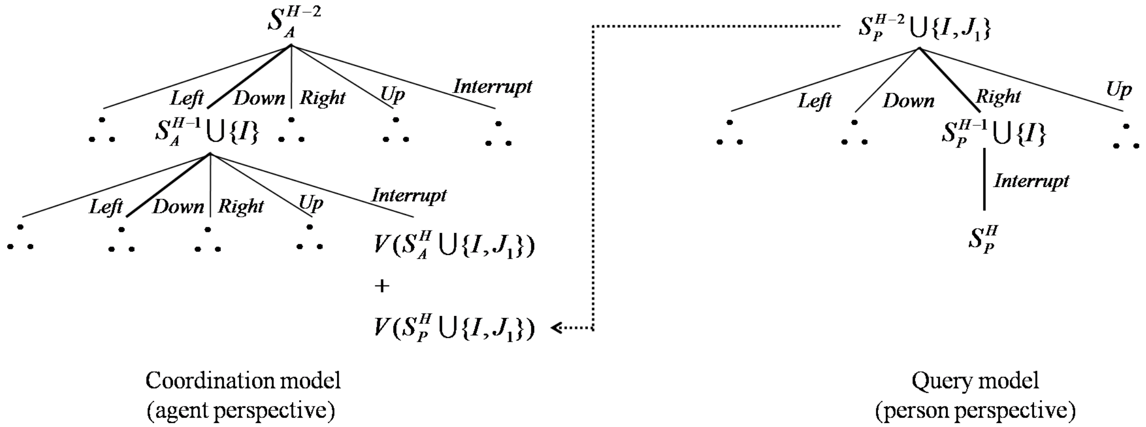


Figure 5.3: Simulating the ND-DECOP algorithm on the interruption problem. Branches representing chosen actions are shown in bold.

Solving the interruption problem with the ND-DECOP algorithm results in exponential savings in computation time. Given the complexity of solving the person MDP is $O(poly(|B|, H))$ where $|B|$ is the size of the board and H is the game horizon, and the complexity of solving the agent POMDP is $O(exp(H))$ in practice, the complexity of solving the interruption problem with the ND-DECOP algorithm is $(|A^J|+1)^H \times O(poly(|B|, H)) + O(exp(H)) \Rightarrow O(exp(H))$. The ND-DECOP algorithm is able to generate an optimal joint policy for the interruption game without adding a significant computational overhead to the complexity of solving the partially-observable single-agent model.

Estimating Value of Interruption

The value of an interruption is the difference between the expected utility (EU) of the current state of the game when the interruption is established and the EU when no interruption is established. Having two players leads to two different perspectives on the value of an interruption. Since the agent cannot observe the position of its goal but keeps a belief about the position; the agent makes interruption decisions based on its estimate of the value of an interruption which is called the "Expected Benefit of Interruption" (EBI). On the other hand, the person knows the actual position of the agent's goal. The person's decision of whether accepting an interruption is based on the "Actual Benefit of Interruption" (ABI). It is beneficial for the agent to initiate an interruption when $EBI > 0$, whereas it is beneficial for the person to accept it when $ABI > 0$.

Let (s^h_P, s^h_A) denote the joint state of the person and agent players as formalized in Section 5.2.2, and $V^{\pi^*}(s^h_P, s^h_A)$ denote the value of the joint state as computed by the ND-DECOP algorithm, $EBI(s^h_P, s^h_A)$ is captured as the difference in the joint utility of the game state between interrupting and carrying out individual tasks as given below:

$$EBI(s^h_P, s^h_A) = EU^I(s^h_P, s^h_A) - EU^{NI}(s^h_P, s^h_A) \tag{5.19}$$

EU^I denotes the expected joint utility when an interruption is established. In case of an interruption, both players cannot move, but their goals may move according to the probability distribution P. When the agent successfully interrupts the person, the person reveals the position of the agent's goal. The agent updates its belief about its goal position in the following round and uses this belief to perform its individual task in future rounds. When it deliberates about whether to interrupt in the current round, it needs to sum over every possible position of its unobserved goal, according to its belief about the goal location.

$$EU^I(s_P^h, s_A^h) = \Sigma_{g_A^h} b^h(g_A^h) \cdot \Sigma_{g_P^{h+1}} P(g_P^{h+1} \mid p_P^{h+1}, g_P^h) \cdot V^{\pi^*}(s_P^{h+1}, s_A^{h+1}) \tag{5.20}$$

Here, $s_P^{h+1} =< p_P^{h+1} = p_P^h, g_P^{h+1} >$, $s_A^{h+1} =< p_A^{h+1} = p_A^h, b^{h+1} >$ and, b^{h+1} denotes the belief state of the agent in which probability 1.0 is given to position g_A^h, a possible true position of its goal as it is revealed by the person, and updated once to reflect the stochastic movement of the goal in turn h.

EU^{NI} denotes the expected utility when an interruption is not established and both players pursue their individual tasks by selecting the actions that maximize their joint expected utility.

$$EU^{NI}(s_P^h, s_A^h) = max_{a_P^h \in A_P^I, a_A^h \in A_A^I}(\Sigma_{g_P^{h+1}} P(g_P^{h+1}, \mid p_P^{h+1}, g_P^h) \cdot V^{\pi^*}(s_P^{h+1}, s_A^{h+1})) \tag{5.21}$$

Here, $s_P^{h+1} =< p_P^{h+1} = p_P^h + a_P^h, g_P^{h+1} >$, $s_A^{h+1} =< p_A^{h+1} = p_A^h + a_A^h, b^{h+1} >$, and b^{h+1} denotes the updated belief of the agent.

The agent cannot observe the correct location of its goal and estimates the benefit of interrupting under this uncertainty using the EBI calculations given above. Thus, not every interruption initiated by the agent is truly beneficial for the team. In contrast, the person can observe the position of agent's goal and can capture the actual benefit of the interruption, denoted as ABI with certainty. Any interruption with positive ABI offers a positive expected benefit to the team.

$$ABI(s_P^h, s_A^h) = EU_P^I(s_P^h, s_A^h) - EU^{NI}(s_P^h, s_A^h) \tag{5.22}$$

Here, the term EU_P^I refers to the person's estimate of the utility of a game state when an interruption is established.

$$EU_P^I(s_P^h, s_A^h) = \Sigma_{g_P^{h+1}, g_A^{h+1}} P(g_P^{h+1} \mid p_P^{h+1}, g_P^h) \cdot P(g_A^{h+1} \mid p_A^{h+1}, g_A^h) \cdot V^{\pi^*}(s_P^{h+1}, s_A^{h+1}) \tag{5.23}$$

where, $s_P^{h+1} =< p_P^{h+1} = p_P^h, g_P^{h+1} >$, $s_A^{h+1} =< p_A^{h+1} = p_A^h, g_A^{h+1}, b^{h+1} >$ [5] and b^{h+1} refers to the belief state of the agent in which probability 1 is given to g_A^h, the true position of its goal as it is revealed by the person, and updated once to reflect the stochastic movement of the goal in turn h.

The proposed algorithms and the benefit of interruption calculations presented in Sections 5.2.1 and 5.2.2 are not meant to predict how people play the interruption game or respond to interruption requests in general. Rather, they provide a way to compute a theoretical baseline, which is a fully rational computational estimate for the value of interruption in the game. They assume that person players are fully rational and computationally unbounded. In the case of such players, any interruption with positive ABI is expected to be accepted, and any interruption with negative ABI is expected to be rejected. However, people may not be fully rational or computationally unbounded, people's perception of the benefit of interruptions may differ from baseline values calculated by the DECOP and ND-DECOP algorithms. In the empirical investigations described in the next chapter, these baseline values are compared with the subject responses to detect the possible mismatch between a computer's estimate of the benefit of an interruption and a person's perception of it and to identify a subset of factors that affect the way that humans perceive the effectiveness of interruptions. This analysis allows the study of the efficacy of these models when they are used by computers to interact with people under various experimental conditions.

[5] The agent's state is defined from the person's perspective.

Chapter 6

Modeling Human Perception of Interactions

When people participate in collaborative activities with computer agents, it is necessary for the agents to reason about the ways in which people perceive the utility of the collaboration and its constituent actions. This chapter empirically investigates the mismatch between the actual utility of an action in a collaborative context and people's perception of it, explores the different factors that may influence people's perception of this utility. The failure to consider this mismatch may cause a person to reject a valuable interaction opportunity, thereby turning what could have been a beneficial interaction for the collaboration into a performance degrading disturbance. The data collected from the empirical investigation is used to learn about this mismatch and to improve upon the decision-theoretic baseline models.

6.1 Empirical Methodology

This section uses strategies derived from the fully rational decision-making models for playing the interruption game to explore the way people make decisions in an empirical setting. [1] A total of 26 subjects participated in the study. The subjects were between ages of 19 and 46 and were given a 20 minute tutorial of the game. Subjects played 25 to 35 games each and were compensated in a manner that was proportional to their total performance.

During the empirical evaluation, all subjects were allocated to play the roles of person players, while the role of the agent player was assigned to a computer that used the methodology described in Chapter 5 to play the interruption game. Each game proceeded in the manner described in Section 5.1. In particular, the agent could not observe its own goal location but is allowed to initiate an interruption once to acquire the correct location of its goal from the person.

Interruptions were generated by the computer agent at different points in the game with varying actual benefits, game rounds and perceived partner types. People's responses to these requests are measured based on the game conditions at the time of interruptions, which included the number of turns left to play, the positions of both players on the board, and the agent's belief about the location of its goal.

A person player that uses the decision-theoretic models ND-DECOP and DECOP to determine whether to accept an interruption request is perfectly rational in that it uses the

[1] In these experiments, DECOP algorithm is used to calculate EBI and ABI values. Given that the interruption game is designed to have at most one interruption request per game, the DECOP algorithm is able to accurately capture the benefit of an interruption generated by the agent. In single-shot interruption scenario, when an interruption is initiated, the actual benefit of interruption calculated from the person perspective by the DECOP algorithm is identical to the value computed by the ND-DECOP algorithm for the same scenario.

collaborative benefit of interruption (ABI) as the sole factor for this decision. However, people are expected to differ from this rational model. The purpose of the empirical work presented in this chapter is to measure the extent to which different factors in the game, such as the collaborative benefit of interruption (ABI), the timing of interruptions and the perceived partner type, influence people's perception of interruptions. To investigate the way subject responses change with respect to benefit of interruptions, the game scenarios were varied to have different ABI values. To investigate the effect of the timing of an interruption on the subjects' likelihood of acceptance, interruptions are initiated at various rounds of the game. Lastly, the type of agent player (whether a computer or human) is expected to affect the way people respond to interruption requests. For this reason subjects were told they would be interacting with a human for some games, however, they were always paired with an agent. [2]

Subjects were given randomly generated game scenarios that vary the actual benefit of interruption to both participants (ABI) to cover four types of values: -1.5 (small loss), 1.0 (small gain), 3.5 (medium gain), 6.0 (large gain). These values represent the smallest and largest benefit values that can be generated from interruptions with positive expected benefit (EBI), which is a necessary condition to initiate interruption requests by the agent player. The rounds in the game in which interruptions occurred varied to cover the beginning of a game (round 3), the middle of a game (round 5) and the end of a game (round 7). There were 540 game instances played when the perceived agent was a computer (PP:Computer) and 228 data points when the perceived agent was a person (PP:Person).

[2] Approval was obtained for the use of human subjects in research for this misinformation.

6.2 Results

The following results analyze a total of 768 game instances collected in the empirical study. Table 6.1 shows interruption acceptance rates for different rounds and ABI values for the same game instances when perceived partner type (PP) is a person or agent. The optimal policy for the person player is to accept an interruption if its associated benefit (ABI) is positive and to reject otherwise.

To test the significance of the results presented in Sections 6.2 and 6.3, the pairwise sign test is applied to the data collected from the interruption experiments. The pairwise sign test is a nonparametric statistical method that does not make assumptions about the distribution of the collected data and is suitable for the collected data which is best represented as paired binary data points (Conover, 1999). To analyze the effect of one of these factors individually, all matched pairs of data points differing only in that factor are examined (e.g., (AOI=1.0, round=3, PP=person) vs (AOI=3.5, round=3, PP=person)).

As the results of Table 6.1 show, the benefit of an interruption is the major factor influencing the probability that the interruption will be accepted by a person. The interruption acceptance rate increases significantly as the benefit of interruption rises from -1.5 to 1.0 ($p < e^{-20}$, α=0.001) and from 1.0 to 3.5 (p=0.0013, α=0.01). However the increase from 3.5 to 6.0 does not further improve the acceptance rate. These results confirm that people are successful at perceiving interruption benefits above a certain threshold. Similarly, when an interruption is costly for the collaboration, people are significantly more likely to reject the interruption. However, subjects vary in their responses to interruptions offering slightly positive gains, indicating the difficulty to estimate the benefit of interruption when its usefulness is ambiguous.

Table 6.1: Acceptance Rate of Interruptions.

PP:Computer	Round 3	Round 5	Round 7
ABI -1.5	0.16	0.16	0.41
ABI 1.0	0.27	0.7	0.81
ABI 3.5	0.91	0.97	0.79
ABI 6.0	0.91	0.95	0.95
PP:Person	Round 3	Round 5	Round 7
ABI -1.5	0	0.11	0.44
ABI 1.0	0.72	0.94	1
ABI 3.5	0.91	0.94	0.88
ABI 6.0	1	0.88	1

Figure 6.1 summarizes the acceptance rates of interruptions as a function of the actual benefit of interruption and perceived partner type (person vs. computer). The figure is divided into three regions of interruption benefits: small losses (Region 1), small gains (Region 2) and large gains (Region 3). The analysis shows that for large losses (Region 1) and for small gains (Region 3), changing the perceived partner type does not affect the likelihood that the interruption will be accepted. In contrast, for interruptions offering small gains (Region 2), the acceptance rate is significantly larger if the perceived partner type is a person ($p = 3 \times 10^{-5}$, $\alpha = 0.001$). This result implies that when the benefit of interruption is straightforward, people do not care whether the initiator of the interruption is a person or a computer. However, for those cases in which the benefit of interruption is ambiguous,

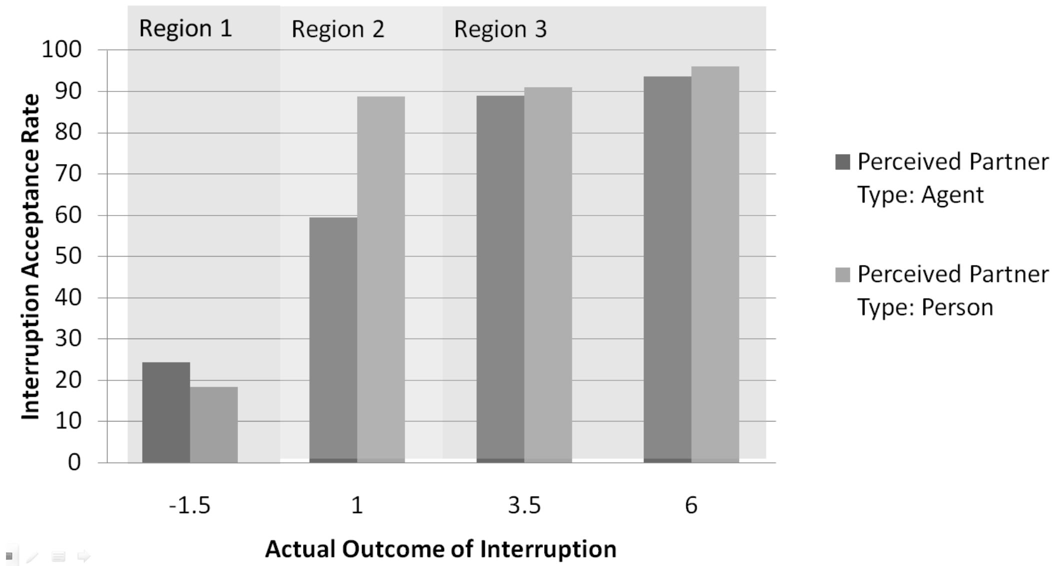


Figure 6.1: Effect of interruption benefit and perceived partner type on interruption acceptance rate.

people are more likely to accept interruptions that originate from other people. This result suggests that agent designers need to consider the way they present interruptions to their users in cases where the perceived benefit is ambiguous. It also aligns with recent studies showing that mutual cooperation is more difficult to achieve in human-computer settings as compared to settings involving people exclusively (Rilling et al., 2004).

Figure 6.2 shows the effect of interruption timing (the round of the game) on people's acceptance rates for interruptions of small losses and small gains. (The interruption timing does not affect the acceptance rate for interruptions of large gains.) It was hypothesized that interruptions occurring late in the game (i.e., with fewer number of turns left in the game) are more likely to be accepted when they incur positive benefit and rejected when incurring a loss. However, as shown in Figure 6.2, as the game round increases, so does the acceptance rate for interruptions of both small losses (ABI -1) and small gains (ABI

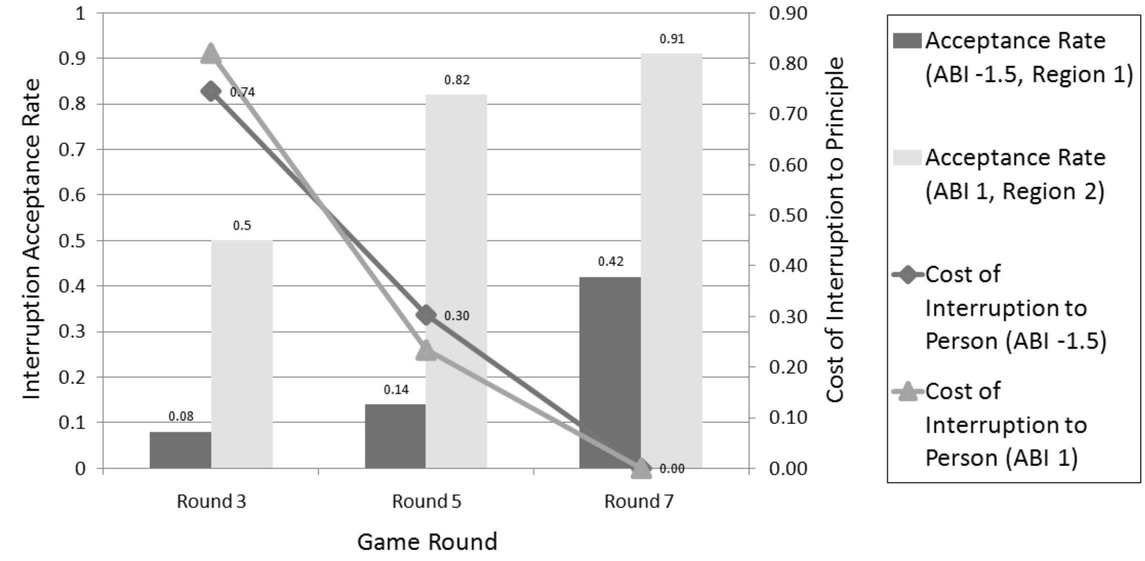


Figure 6.2: Correlation of interruption acceptance rate with the cost of interruption to subjects (-ABI_P) for small gains and losses.

1). There is a significant increase in the acceptance rate when game round increases from 3 to 5 (p=0.002, α=0.01) and from 5 to 7 (p< 10^{-6}, α=0.001).

One factor that may explain the correlation between the acceptance rate and the game round for interruptions of small gains and losses is the cost of interruption to the person. As shown in Figure 6.2, the cost of interruption to the person (ABI_P) decreases as game round increases. Thus, for interruptions of small gains and losses, the acceptance rate is negatively correlated with the cost of interruption to the person. In addition, it is revealed that the benefit of the interruption to the person (ABI_P) is a better predictor of the acceptance rate than ABI_A, the benefit of interruption to the agent (logistic regression $SE = 0.05, R^2 = 0.19, p < 0.001$). Thus, human subjects tend to overestimate their own benefit from interruptions as compared to the benefit for the agent. Consequently, the benefit of interruption to person may be weighed more in people's decision making models than the benefit of the interruption, and people may be more likely to accept an interruption with low ABI_P among interruptions with identical benefit. The studies presented in Section 6.3 further investigate these conjectures by applying learning techniques on the collected data to better understand the correlation of acceptance rate with the cost of interruption to the person. This conjecture is supported by some subject responses to survey questions regarding their strategies for accepting interruptions. Answers include:

"If the agent was in the totally wrong direction and I had several moves left, I would allow the interruption. I always wanted the sure thing for myself."

"If the collaborator was way off in knowing and had enough moves to likely catch it after I told the location, I accepted. If it compromised my ability to get my goal, I declined."

Lastly, it is important to emphasize that these results are a first step in understanding the human perception of interruptions in collaborative settings. The goal of this thesis is not to design computational strategies directly applicable for interruption management in real world domains, but rather to show that effective interruption management needs to consider the collaborative benefit of interruption to both user and system and to point system designers to the types of factors that people consider when they reason about interruptions.

6.3 Learning Human Perception

The ND-DECOP and DECOP algorithms that are presented in the previous chapters assume that people can accurately compute the expected benefit of interacting with computer agents and that they act rationally. In the interruption game, this means that people only accept interruptions at a given round if doing so improves the benefit to the collaborative activity. As discovered in the last section, when the collaborative benefit of interaction is not clear cut, some additional factors may be affecting human decision-making. The purpose of this section is applying learning techniques to investigate these additional factors in collaborative domains in which computer agents and people interact. Several classical learning techniques are employed in the interruption setting to improve fully-rational computational models by learning from empirical data and adjusting them to the characteristics of human decision making.

The goal of this section is not improving the state of the art in learning literature or claiming that the methods implemented are the best to learn human behavior, but empirically showing that learning from user data even with simple learning methods help to improve the way computer agents interact with people. These investigations also highlight important challenges of learning a mathematical model from our user data. These challenges include:

- **Noisy data:** As the data is collected from human studies, it is not an output of a computational model but is a noisy outcome of a collection of computational models and simple-to-complex heuristics that people may be using to make decisions. There may not be a true concept that can be learnt or there may not be a mathematical model that fits the data perfectly. It is highly possible that our subjects did mistakes in evaluating the benefit of interruption requests. These mistakes may be a combination of random errors in judgment and mistakes due to characteristics of human decision-making. It is important to distinguish the mistakes that are characteristics of human decision-making and learn important characteristics of human decision-making in a way that is generalizable to unseen instances.

- **Many features:** Our human subjects may have used many features of the game for making interruption decisions in the CT game. This situation is common for real-world settings in which people's decisions may get affected by many controlled and uncontrolled factors. Our game features include but not limited to game round, player and goal positions, agent's belief distribution about its goal's location, partner type. It is known that considering a large feature set in learning may cause overfitting. Another challenge is selecting a minimal feature set that includes relevant features.

- **Variety in human responses:** Subjects may differ in their decision making. Although the goal is learning general characteristics of human decision-making, personalizing the interaction model may help to improve the model even further.

- **Sparse data:** Given the human model is trained with the data collected from human experiments, it is important to choose a learning algorithm that can work well with a sparse dataset.

The learning problem is defined as follows: Given an interruption instance I that is described by a set features $X = \{x_1, x_2, ..., x_n\}$ extracted from the game state, and a class variable $C = \{Accept, Reject\}$, the task is to learn a classifier function $f : X \rightarrow C$ that can correctly determine whether instance I with feature set X is acceptable by a given user. This problem is modeled as a supervised learning problem and the data collected from our human studies is used to learn classifiers.

The set of features, X, consist of domain-dependent and decision-theoretic features. The domain-dependent (domain) features can be easily extracted from the game state without using a complex computational model. These features include; *partner type* which is either person or agent, *game round* in which interruption is established, *belief distance* which is the aggregate of the term-by-term subtraction of matrices representing agent's belief and correct belief about the position of agent's goal, *agent goal accessible* which is -1 if agent's goal is not accessible within the remaining number of game rounds, 0 if it is barely accessible and 1 otherwise, *person goal accessible* which is defined accordingly for the person player. The decision-theoretic features are computed by the ND-DECOP and DECOP models as presented in Chapter 5. These features include ABI, the collaborative benefit of interruption; ABI_P, person player's benefit of interruption; ABI_A, agent player's benefit of interruption. The set of features referred as *full features* consist of both domain-dependent and decision-theoretic features.

Due to the challenges given above, finding an appropriate learning approach is not a trivial problem. Parametric learning methods make strong assumptions about the shape of the true concept and fail badly when training data does not agree with the assumptions. As stated by the "Lack of true concept" property, it is difficult to make assumptions about the shape of the classifier that best describes the empirical data collected from user studies. On the other hand, non-parametric methods do not make assumptions about the shape of the classifier, but suffer from the curse of dimensionality and become infeasible when the feature set is large.

This section presents investigations with two simple but popular learning methods that do not suffer from the curse of dimensionality. The methods are empirically evaluated for building general models that are trained with the data collected from all subjects, and personal models that are trained individually for each subject. The section concludes with a mixture model that combines the strengths of general and personal models.

6.3.1 Naive Bayesian Learning

Naive Bayesian is a learning method that does not focus on learning a classifier of a certain shape but uses the instances of training data to shape a classifier. It makes the assumption that all of the attributes are conditionally independent of each other given the class, which is true for many subset of the attributes for the interruption domain. Many features extracted from the domain represent individual properties of the game state, thus independent. For example, game round, partner type; goal accessibility features for agent and person; ABI_A and ABI_P are feature pairs that are independent of each other.

Suppose an interruption instance of the game is represented by a set of features $\{x_1$, ..., $x_n\}$, which is a combination of Boolean, discrete and continuous features. The continuous features are discretized into a finite number of intervals. Each interruption instance is labeled with a class $c \in C = \{Accept, Reject\}$ according to the responses collected during the CT experiments. Maximum likelihood approach is used to estimate the parameters of the Naive Bayesian model. For each class c, the density function $P(c)$ is estimated from the training data that specifies the likelihood of class c. For each feature x_i and class c, the density function $P(x_i|c)$ represents the ratio of instances with feature x_i taking a particular value among the instances of class c. Naive Bayesian classifier chooses the class $c \in C$ that maximizes $P(c|x_1, ...x_n)$, as given below:

$$P(c|x_1, ..., x_n) = \frac{P(c) \times \prod_{i=1}^{n} P(x_i|c)}{P(x_1, ..., x_n)} \tag{6.1}$$

Table 6.2 compares the performance of various Naive Bayesian models to a dummy classifier that classifies every interruption instance to $Accept$ and to the decision-theoretic baseline that initiates interruptions when they are estimated to be beneficial by the algorithm given in Section 5.2.2 on the dataset collected from the interruption experiments. For

Table 6.2: Comparison of various Naive Bayesian models with baseline models.

Model	Precision	Recall	Accuracy	F	F_2	$F_{0.5}$
Dummy model	0.71	1	0.71	0.83	0.92	0.75
Decision-theoretic baseline	0.85	0.91	0.82	0.88	0.90	0.86
General Naive Bayesian (domain features)	0.82	0.92	0.80	0.87	0.90	0.84
General Naive Bayesian (full features)	0.87	0.92	0.85	0.90	0.91	0.89
Personal Naive Bayesian (domain features)	0.84	0.80	0.75	0.82	0.81	0.83
Personal Naive Bayesian (full features)	0.90	0.76	0.77	0.82	0.78	0.87

each classifier, the table reports the precision value, the likelihood that a notification created by the classifier is relevant; the recall value, the likelihood that a relevant notification is delivered; the accuracy value, the likelihood of a correct classification. As a measure that weighs precision and recall equally, the table reports the F-measure which is the harmonic mean of precision and recall. F_2-measure is used to compare the performance of the classifiers when the cost of missing a relevant notification is twice as much as the cost of receiving an irrelevant notification. $F_{0.5}$-measure is used for the opposite case where receiving an irrelevant notification is twice as costly as missing a relevant notification.

For comparison multiple Naive Bayesian classifiers are generated by varying the training sets and the features to be considered. Two classifiers are trained and tested with game instances that only consist of domain features (domain-dependent features) and game instances represented with full features. To identify the effect of personalization, general and personal classifiers are generated and evaluated empirically. General classifiers are trained and tested on the complete anonymous training set with cross-validation. Personal classifiers are trained and tested individually for each subject with leave-one-out cross validation. Observing the effect of personalization is important because human decision-making may differ from person to person, and learning personal models of human perception of interruptions may further improve the accuracy of the classification.

As shown in Table 6.2, the Naive Bayesian classifiers trained with the simple feature set perform significantly better than the dummy classifier ($p < 2 \times e^{-9}$, α=0.01) but fail to perform as well as the decision-theoretic baseline. On the other hand, the classifier generated by combining decision-theoretic reasoning with learning (General Naive Bayesian model on full feature set) performs significantly better than the decision-theoretic baseline (p=0.012, α=0.05).

The accuracy of Naive Bayesian models on both full and domain feature sets decreases significantly when they are trained and tested personally. A possible explanation for the decrease in performance is the scarcity of the personal data for a subject. 28 data points per person in average may not be sufficient to learn a good classifier given that the classifier will be a function of many attributes.

These results are important to show that general models can be learnt from user data to better predict the way people perceive interruptions. They also show that the outputs of the decision-theoretic algorithms are useful attributes that contribute to learning more accurate models, in particular when training data is sparse.

6.3.2 Perceptron Learning

In this section, perceptron learning methods are used to learn different weights people may be putting to different components of the collaborative utility.

A perceptron is a simple and efficient classifier that assigns weights to inputs, and uses the weighted summation of the inputs to draw a hyperplane as a decision boundary. It uses a threshold activation function to classify the weighted sum to a binary value. Given that an interruption instance I is defined by a feature set $X = \{x_1, x_2, ..., x_n\}$, w_i represents the weight of feature x_i and T is the threshold of a perceptron, the perceptron activation function f is defined as:

$$f(X) = \begin{cases} Accept & \text{if } \Sigma(x_i \times w_i) > T \\ Reject & \text{otherwise} \end{cases}$$

More specifically, a perceptron that classifies an interruption instance only with respect to the way people weigh individual benefits of players (ABI_A, ABI_P) is defined as:

$$f(ABI_A, ABI_P) = \begin{cases} Accept & \text{if } w_A \times ABI_A + w_P \times ABI_P > T \\ Reject & \text{otherwise} \end{cases}$$

A perceptron takes a set of continuous inputs and maps them to a Binary decision. Due to the size of the hypothesis space of a perceptron, it is easy to learn a perceptron even with a small data set. The Perceptron Algorithm takes a set of labeled training instances of continuous features (Rosenblatt, 1958). Starting from an all-zero weight vector, the algorithm classifies training instances. For each wrongly classified training instance, the algorithm updates the weight vector and the threshold value by adding a factor that is proportional to a learning rate. The Basic Perceptron Algorithm is guaranteed to converge if the data is linearly separable.

The Basic Perceptron Algorithm fails to learn an accurate linear separator from the noisy empirical data when tested on the dataset collected from CT experiments. As a solution, a noise tolerant variant of the Perceptron Algorithm proposed by Khardon and Wachman (2007) is applied to the data. This algorithm is more robust to noise and performs significantly better on the data set. This variant of the algorithm adds a margin term for robustness and accuracy, and chooses the best hypothesis that explains the data by majority voting. The margin term mimics the soft-margin property of Support Vector Machines by updating weights not only when a mistake is done, but also when the weighted sum falls within a margin of the threshold value (Krauth and Mezard, 1987; Li and Long, 2002). Because the last hypothesis generated by the Basic Perceptron Algorithm is not guaranteed to be the best separator when data is noisy, the majority voting variant chooses the classifier that has the highest accuracy on the training set (Freund and Schapire, 1999).

Table 6.3 compares the performance of various Perceptron models with the baseline models. Three feature sets are hand-picked for empirical testing: the sets of domain and full features and a small feature set comprised of only individual utilities (i.e., AOI_P and AOI_A). Multiple perceptron classifiers are generated by varying the training sets and the features to be considered.

The empirical results presented in Table 6.3 show that it is possible to generate a perceptron classifier (general perceptron on full features) that is more accurate than the decision theoretic baseline (p=0.03, α=0.05) despite perceptron being one of the simplest learning methods. However, learning the weights of AOI_A and AOI_P with a general perceptron on

Table 6.3: Comparison of various Perceptron models with baseline models.

Model	Precision	Recall	Accuracy	F	F_2	$F_{0.5}$
Dummy model	0.71	1	0.71	0.83	0.92	0.75
Decision-theoretic baseline	0.85	0.91	0.82	0.88	0.90	0.86
General Perceptron (individual utilities)	0.86	0.91	0.82	0.88	0.90	0.86
General Perceptron (domain features)	0.81	0.96	0.81	0.88	0.93	0.84
General Perceptron (full features)	0.87	0.91	0.84	0.89	0.90	0.88
Personal Perceptron (individual utilities)	0.88	0.89	0.84	0.89	0.89	0.89
Personal Perceptron (domain features)	0.84	0.80	0.75	0.82	0.81	0.83
Personal Perceptron (full features)	0.90	0.76	0.77	0.82	0.78	0.87

the anonymous dataset fails to improve over the decision-theoretic baseline. Further analysis shows that this result is caused by the high variation in the way different subjects weigh these individual benefits. [3] Consequently, the next set of experiments focus on learning

[3] Mean($|w_A|$)=0.73, mean($|w_P|$)=0.39, variance(w_A)=0.03, variance(w_P)=0.2, for all w_P, w_A; $w_A, w_P \in [-1.0, 1.0]$. 65% of the users weigh AOI_P more than AOI_A.

personalized perceptrons that can learn the ways different individual subjects weigh the benefits to the agent and themselves.

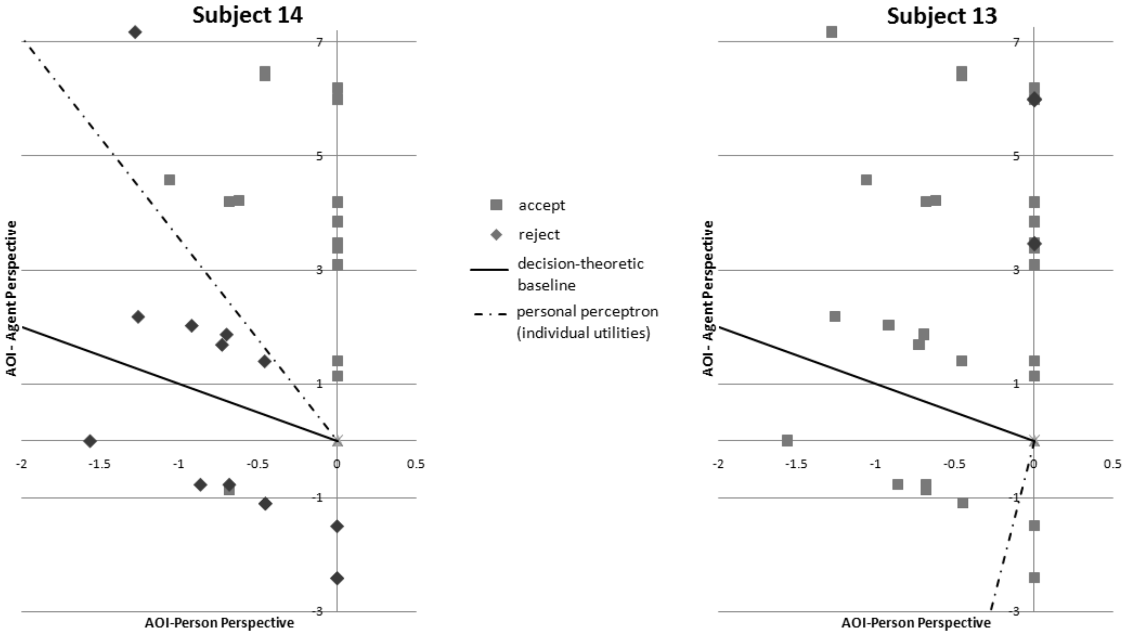


Figure 6.3: Linear classifiers generated by the personal perceptron model for Subjects 14 and 13.

The accuracies of the perceptron models trained with full and simple feature sets decrease significantly when trained individually for each subject on the personal data (p=0.003, α=0.01; p=0.004, α=0.01 respectively) possibly because personal data is not rich enough to learn a classifier of many features. On the other hand, when a perceptron is trained personally with the individual utilities feature set (personal perceptron on individual utilities), it performs significantly better than the decision-theoretic baseline ($p < 9 \times e^{-9}$, α=0.01).

Figure 6.3 illustrates the improvements generated with the personal perceptron on individual utilities feature set for two selected subjects. Classifiers accept interruption instances to the right of the classifier. Square and diamond dots represent interruptions accepted and rejected by the subjects respectively. The figure on the left shows that the learned classifier for Subject 14 correctly classifies 5 false positives created by the baseline model. The learned classifier for Subject 13 shown in the right correctly classifies 5 instances that are incorrectly classified as negatives by the baseline model. However, the improvements are not consistent through the subjects; the personal percepton model improved the prediction accuracy for 27% of the subjects, but negatively affected the prediction accuracy for 31% of the subjects (compared to the decision-theoretic baseline).

These experiments on perceptron learning are important to show that successful personal models can be generated when the chosen learning method is simple enough and feature set is constrained. These models can improve upon the baseline models by learning the differences in the way subjects perceive interruptions.

6.3.3 Mixture Model

Sections 6.3.1 and 6.3.2 have presented a general model (Naive Bayesian on full feature set) and a personal model (perceptron on individual utilities feature set) that significantly improve over the decision-theoretic baseline. However, the performances of these general and personal classifiers are not consistently good on the dataset; the general classifier improves classification accuracy for 42% of the subjects, whereas the personal classifier performs better for only 27% of the subjects (in comparison to the baseline model). A quick analysis shows that if the better performing model (general vs. personal) can be identified and chosen for each subject, such a mixture model has the potential to improve upon the baseline model for 62% of the subjects and to perform at least as well as the baseline for 96% of the subjects. This analysis shows that the general and personal models can learn

Table 6.4: Comparison of the Mixture model with the best performing models.

Model	Precision	Recall	Accuracy	F	F_2	$F_{0.5}$
Decision-theoretic baseline	0.85	0.91	0.82	0.88	0.90	0.86
General Naive Bayesian (full features)	0.87	0.92	0.85	0.90	0.91	0.89
Personal Perceptron (individual utilities)	0.88	0.89	0.84	0.89	0.89	0.89
Mixture	0.90	0.93	0.88	0.91	0.92	0.90

fairly disjoint characteristics about the subjects, but a mixture model has the potential to achieve a better performance.

The hypothesis that is studied in this section is that a combination of the general and personal models may further improve the accuracy of classification. To study the hypothesis, a simple approach is followed which chooses one model to apply over the general and the personal models by using a separate validation set. After each classifier is trained over the training set, the classifier that performs better on the validation set is chosen for testing.

As shown in Table 6.4, the Mixture model performs significantly better than the decision-theoretic baseline ($p < 3 \times e^5$, $\alpha = 0.01$), the general model ($p < 3 \times e^6$, $\alpha = 0.01$) and the personal model ($p = 0.03$, $\alpha = 0.05$). In future work, the performance of the mixture model can be further improved with a more sophisticated model that uses a probabilistic combination of multiple models (Roy and Kaelbling, 2007).

The experiments on the mixture model show that hierarchical models can further improve upon the baseline models by unifying the strengths of learning with personal and general models.

Chapter 7

Collaboration of Self-Interested Agents in Open World

Self-interested agents with conflicting or aligned goals can collaborate on a collective activity if doing so improves their individual utilities. In contrast to general teamwork models that assume the participants of a collaborative activity share a joint utility function, modeling collaboration among self-interested agents requires reasoning about incentives to motivate agents to form and support a successful collaborative plan. This chapter investigates challenges with the generation of efficient collaborative plans for self-interested people based on their preferences and with providing fair incentives to promote collaboration. It explores different mechanism design ideas under real-world considerations by addressing the computational limitations of dynamic mechanisms.

This chapter frames and motivates the development of methods with the real-world challenge of generating shared transportation plans, commonly referred to as *ridesharing*. Rideshare plan generation is an interesting and representative open-world collaboration problem because of the varying goals, diverse preferences, and changing locations and availabilities of actors. Beyond the intriguing technical challenges, rideshare plan generation is an important real-world domain to demonstrate the value of collaboration for real-world applications. Implementing wide-scale collaborative planning for ridesharing can provide value for the environment and the economy.

The rest of the chapter is organized as follows: Section 7.1 reviews the architecture of a prototype system for ridesharing called the Agent-based Carpool (ABC) system. Section 7.2 addresses the problem of generating collaborative plans for ridesharing. Section 7.3 introduces mechanisms for providing incentives to collaborate. Section 7.4 describes a dataset that consists of real-life GPS traces, and Section 7.5 presents an empirical evaluation of the ABC system on the dataset. The chapter concludes with a discussion of various real-world considerations.

7.1 Methodology and Architecture

Computing ideal rideshare plans is a challenging problem as the solution must consider the varied and dynamically changing preferences of self-interested agents, must provide compelling and fair incentives, and must be easy to use. The ABC prototype addresses these challenges by creating personalized rideshare plans while minimizing the cumulative cost of transportation. The system has three main components that embody separate but interrelated reasoning methodologies: a user-modeling component that accesses and represents the preferences of agents, an optimization component that generates collaborative rideshare plans and a payment component that provides incentives to agents to collaborate.

The user-modeling component is responsible for identifying the preferences of agents about their desired commutes and for passing the preferences into the optimization and payment components. It gathers information about agents' individual commute plans, including their origins, destinations, trip timing, and preferences about a return trip. A destination analyzer accesses or infers the intended destination of a mobile user under uncertainty (Krumm and Horvitz, 2006). To perform cost-benefit analysis of a rideshare plan, the user-modeling component models agent-specific costs for driving, delaying a trip, diverting an ideal route to pick up or drop off passengers and changing stop points. Capturing these costs in a dynamic manner is crucial for the success of the rideshare system, as the system needs to adapt to different and changing preferences of agents. For example, an agent may be willing to wait and pick up other agents on the way when the cost of time is low, but not on a rainy day when the cost of time is high.

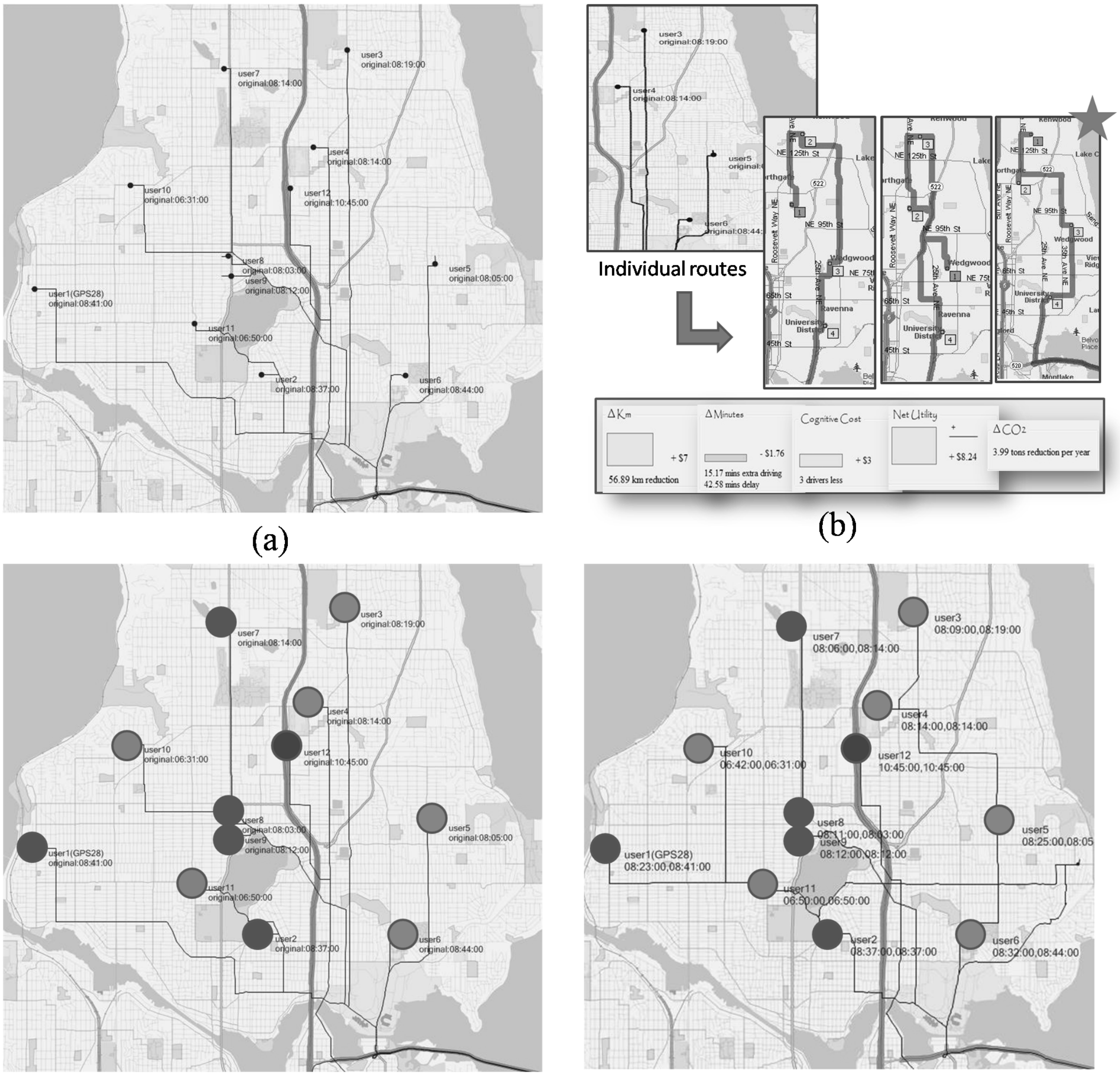
Time is an important resource and is one of the major factors influencing the cost of a commute plan. The user-modeling component employs a probabilistic time-cost model. The model considers as input the time of day, day of week and sets of attributes about agents' commitments drawn from an online appointment book. Probabilistic models for the cost of time and for the commitment to attend events are learned from user-annotated training data via a machine-learning procedure based on a Bayesian structure search. Similar predictive models of the cost of time and meeting commitments have been used in other applications, including mobile opportunistic planning (Horvitz et al., 2007; Kamar et al., 2008), meeting coordination (Horvitz et al., 2002) and the triaging and routing of communications (Horvitz et al., 2005). Horvitz et al. (2005) present the machine learning and reasoning models used for predicting the cost of time in different settings and the empirical evaluation of these predictive models.

For each agent, the user modeling component constructs a time-cost function T to estimate the cost of time spent travelling between start time (t_s) and end time (t_e), and the additional cost for delaying the start time of a rideshare trip from the initial start time t_s^o to t_s. T is captured with respect to the nearest deadlines drawn from the agent's calendar. Given that the set of calendar items falling between $[t_s, t_e]$ is M, $m \in M$ is a calendar item, the start time of m is t_s^m, the end time of m is t_e^m, c_n is the per minute time cost for travelling, c_m is the additional cost for missing a minute of m, c_d is the per minute cost for delay; T is defined as given below:

$$\begin{aligned} T(t_s, t_e) = \quad & ((t_e - t_s) \times c_n) + (|t_s - t_s^o| \times c_d) \\ & +(\sum_{m \in M} (\min(t_e^m, t_e) - \max(t_s^m, t_s)) \times c_m) \end{aligned} \tag{7.1}$$

7.2 Generating Collaborative Plans for Ridesharing

The optimization component groups agents together and generates a collection of rideshare plans that maximizes the efficiency of transportation. The component acquires private user preferences from the user modeling component and combines them with global contexts to capture the collaborative value of a rideshare plan. The optimization component has the following properties that make it difficult for agents to find out about other agents in the system and thus collude in the mechanism; the component combines multiple user preferences and contextual factors to determine the best possible plan, and agents do not know the preferences or rideshare plans of other agents that they are not participating in a rideshare plan with. The optimization component takes in agents' individual commute plans as input and solves two difficult optimization problems to generate a collection of collaborative rideshare plans. The two optimizations are: (1) generating rideshare plans for groups of agents and (2) clustering agents into rideshare groups.



(a)

(b)

(c)

(d)

Figure 7.1: Steps of the ABC rideshare optimization.

Figure 7.1 shows the steps of the ABC rideshare optimization. Part (a) of the figure illustrates the input of the optimization component as a set of individual commute plans. The initial segments of the individual plans are drawn on the map originating from the start positions of agents which are indicated by black dots. Each start position is labeled by the username and original start time. Part (b) of the figure illustrates the rideshare plan optimization step. Individual plans acquired from four agents are shown on the top left of the figure. Middle right images illustrates three different rideshare plans generated for agents. The economical analysis of a rideshare plan in terms of fuel, time, cognitive costs and CO_2 emissions is illustrated on the bottom. The plan with the highest expected value is selected for the group. Part (c) of the figure illustrates the rideshare group optimization step. Agents assigned to the same rideshare groups are labeled with circles of the same shade. Part (d) of the figure illustrates the set of final rideshare plans generated by the optimization component. Collaborative rideshare plans are drawn on the maps. Each circle represents the rideshare group that an agent is assigned, the circle is labeled with the username of the agent and with the updated and original start times.

7.2.1 Rideshare Plans

Choosing the best possible rideshare plan with respect to the agents' preferences is a large search problem where the system explores possible combinations of commute start times, stop orderings, stop locations, commute durations and possible routes among stop points to generate a plan with the highest possible cumulative value. Let P be the set of all agents in the rideshare system, $S \subseteq P$ a rideshare group, and $\mathcal{C}(\mathcal{S})$ the universe of all possible rideshare plans for S. A rideshare plan $C_i \in \mathcal{C}(\mathcal{S})$ is specified by the following attributes:

- $S = \{p_h, \ldots, p_q\}$, the set of agents participating in the rideshare plan; $p_d \in S$, the assigned driver for the rideshare plan; $S_{-d} = S \setminus \{p_d\}$, the set of agents of the rideshare group excluding the driver.

- $\mathcal{L}_{-d} = \{\ell_{h,s}, \ell_{h,e}, \ldots, \ell_{q,s}, \ell_{q,e}\}$, the set of start\end (stop) locations of agents in S_{-d}, where agent p_i's start location is $\ell_{i,s}$, end location is $\ell_{i,e}$. For all $p_i \in S_{-d}$, $\ell_{i,s}$ and $\ell_{i,e}$ are located in a predetermined radii [1] of $\ell^o_{i,s}$ and $\ell^o_{i,e}$ – the initial start\end locations for p_i's individual commute plan. $\mathcal{L}$, the complete set of start\end locations, is the combination of $\mathcal{L}_{-d}$ with the start\end locations of the driver agent p_d: $\mathcal{L} = \mathcal{L}_{-d} \bigcup \{\ell_{d,s}, \ell_{d,e}\}$, where $\ell_{d,s} = \ell^o_{d,s}$, $\ell_{d,e} = \ell^o_{d,e}$.

- Θ_{-d}, a commute chain excluding p_d, is any ordering of $\mathcal{L}_{-d}$ such that for all $p_i \in S_{-d}$, $index(\ell_{i,s}) < index(\ell_{i,e})$ (i.e., any agent's start location precedes the end location in Θ_{-d}). $\Theta = \ell_{d,s} \circ Q_{-d} \circ \ell_{d,e}$ is the commute chain for S, including the driver's start location as the departure location and the driver's end location as the arrival location.

- t_s, the start time of the rideshare plan. $t(l)$, the scheduled time of stop location l, is defined as below, where $\Delta t(\ell_j, \ell_{j+1})$ is the estimated travel duration between two consecutive stop locations ℓ_j, $\ell_{j+1} \in \Theta$:

$$t(\ell) = \begin{cases} t_s & \ell = \ell_{d,s} \\ t_s + \displaystyle\sum_{j<index(\ell)} \Delta t(\ell_j, \ell_{j+1}) & otherwise \end{cases} \tag{7.2}$$

7.2.2 Value of Rideshare Plans

Although reducing fuel costs and CO_2 emissions from vehicles, as personal or organizational (e.g., per the goals of an employer) goals, are the primary motivations for bringing self-interested agents to collaborate in rideshare plans, the additional time and travel required for adding new stops to a commute or having fewer agents driving in heavy traffic

[1] This radius is determined by the agent based on its preferences for divergence from its original start\end locations.

can play an important role in the willingness of agents to participate in the rideshare system. This section defines a personal inconvenience cost model that captures several agent-specific cost factors. A set of personal inconvenience factors are combined to compute the cumulative value of a rideshare plan.

The model for the cost of personal inconvenience combines the cost of lengthening the duration of a commute and of shifts in departure and arrival times with gains in the fuel savings and a reduction in the cognitive costs of driving a vehicle, to yield an estimate of the net value of an agent becoming associated with the commute. The user modeling component provides a probabilistic time-cost function, $T_i(t_s, t_e)$. The fuel cost in dollars for one mile is denoted as c_g. The inconvenience model combines the input from the user modeling component with traffic prediction services and public contexts (e.g., daily events that may affect the traffic) to construct a cognitive cost model for an agent. $CC_i(\ell_s, \ell_e)$ denotes the predicted cognitive cost of p_i for driving between the given stops. The optimization engine makes calls to Microsoft Mappoint services to estimate the travel duration. $\Delta t(\ell_i, \ell_j)$ denotes the duration of travel between stops ℓ_i and ℓ_j, and $\Delta d(\ell_i, \ell_j)$ denotes the distance to be travelled between these stops.

The initial inconvenience cost of agent p_i, $PC^o(p_i)$, represents the cost of following the individual commute that would be created between the initial start\end locations of p_i in the absence of ridesharing, where the start time of the individual commute is $t^o_{i,s}$.

$$PC^o(p_i) = T_i(t^o_{i,s}, t^o_{i,e}) + \Delta d(\ell^o_{i,s}, \ell^o_{i,e}) \times c_g + CC_i(\ell^o_{i,s}, \ell^o_{i,e}) \tag{7.3}$$

$$t^o_{i,e} = t^o_{i,s} + \Delta t(\ell^o_{i,s}, \ell^o_{i,e}) \tag{7.4}$$

An agent incurs costs for driving (e.g., fuel and cognitive costs), if assigned as the driver in a given commute. Let $\ell_j, \ell_{j+1} \in \mathcal{L}$ be consecutive stop locations in commute chain Θ,

$PC(p_d, C)$ is the inconvenience cost of driver agent p_d for rideshare plan C.

$$PC(p_d, C) = T_d(t(\ell_{d,s}), t(\ell_{d,e})) + \sum_{\ell_j, \ell_{j+1}} (\Delta d(\ell_j, \ell_{j+1}) \times c_g + CC_d(\ell_j, \ell_{j+1})) \tag{7.5}$$

The passengers of a rideshare are only subject to time costs for the period of travel time between their scheduled start and end locations. $PC(p_i, C)$ is the inconvenience cost of passenger $p_i \in S_{-d}$ for rideshare plan C.

$$PC(p_i, C) = T_i(t(\ell_{i,s}), t(\ell_{i,e})) \tag{7.6}$$

$v_i(C)$ represents the value of agent p_i for rideshare plan C. The cumulative value of a rideshare plan, $V(C)$, represents the value of agents in rideshare plan C switching to collaborative plan C from their individual plans.

$$\begin{aligned} v_i(C) &= PC^o(p_i) - PC(p_i, C) && (7.7) \\ V(C) &= \sum_{p_i \in S} v_i(C) && (7.8) \end{aligned}$$

Before leaving the discussion of preferences, it is important to note that there are subtle, yet potentially powerful psychological and social costs and benefits associated with sharing rides with others. There is opportunity in assessing and smoothly integrating key psychosocial factors as additional costs into the optimization used for generating plans. For instance, participants can be offered the option of providing preference functions that yield estimates of the cost of traveling with one or more people based on an established reputation and on social or organizational relationships. For example, preferences can be captured with utility functions that specify the costs of including people in a shared plan that are related to the participant via different types of organizational links or via increasing graph distances in a social network. Such additional costs would likely influence individual objective functions, and thus the overall behavior of the system, leading to modifications in the rideshare plans generated as compared to the output system that ignores psychosocial issues.

7.2.3 Plan Optimization as Search

Rideshare plan optimization seeks to identify the shared transportation plan for a group of agents S with the highest cumulative value. This optimization problem is a search problem over the universe of rideshare plans $\mathcal{C}(\mathcal{S})$ available for S, where the search dimensions of $\mathcal{C}(\mathcal{S})$ are the set of possible commute chains, set of possible stop locations for the passengers, trip start times, and potential routings between stop points. The optimization component performs geospatial search over the feasible paths that satisfy the constraints of a rideshare plan for S. Given the start\end locations of the assigned driver, the optimizer considers sets of updated routes by adding potential passenger stop points as waypoints and performing A^* search. The set of potential passenger stop points is selected from a radius around the initial stop points of the passenger. The magnitude of the radius is limited by the maximum distance the passenger is willing to diverge from the initial stop location to have a more efficient rideshare. The engine searches for the start time of the rideshare plan that minimizes the total cost.

The plan optimizer selects the plan $C^*(S)$ that offers the maximum cumulative value to agent set S, among all possible plans $\mathcal{C}(\mathcal{S})$. It provides $C^*(S)$ to the rideshare group optimizer.

$$C^*(S) = \arg\max_{C_j \in \mathcal{C}(\mathcal{S})} V(C_j) \tag{7.9}$$

7.2.4 Group Assignment as Set Cover

Given a set of agents P in the rideshare system, the rideshare group optimization finds the set of subset of P that covers all agents in P by offering the highest cumulative value. Thus, this optimization is identical to the well-known NP-hard set-cover problem.

Let us consider a set of agents, $P = \{p_1, \ldots, p_n\}$ willing to collaborate in a rideshare system. k is the capacity of a single vehicle, thus the maximum size of a collaborative rideshare group. A set cover for $SC_i = \{S_h, \ldots, S_m\}$ for agent set P is a set of subsets of P, such that for all subsets S_j; $|S_j| \leq k$, $\bigcup_{S_j \in SC_i} S_j = P$, and for any $S_j, S_k \in SC_i$ $S_j \bigcap S_k = \emptyset$. Thus, set cover SC_i represents a collection of rideshare groups and their best possible rideshare plans that cover all agents in the rideshare system without exceeding the capacity of a transportation vehicle. $\mathcal{SC}(\mathcal{P}) = \{SC_1, \ldots, SC_r\}$ denotes the universe of all set covers for set of agents P.

Valuation function $V(S_j)$ denotes the value generated by the rideshare plan offering the highest value to agent group S_j. The value of a set cover SC_i is calculated as:

$$V(S_j) = \begin{cases} 0 & |S_j| \leq 1 \\ V(C^*(S_j)) & otherwise \end{cases} \tag{7.10}$$

$$V(SC_i) = \sum_{S_j \in SC_i} V(S_j) \tag{7.11}$$

A set-cover solver returns the optimal set cover $SC^* = \arg\max_{SC_i \in \mathcal{SC}(\mathcal{P})} V(SC_i)$.

The dynamic, open-world nature of the rideshare domain requires the optimization to run efficiently, since agents may unexpectedly arrive, leave or change preferences which may result in running the optimization multiple times. However, solving the set-cover problem optimally takes exponential time in practice. Additionally, the optimization of rideshare plans requires calls to computationally expensive online traffic prediction and routing services to evaluate the value of each set cover, which makes the optimization calculations more expensive. Due to the infeasibility of applying optimal set-cover solvers in open-world settings, an approximate greedy set-cover algorithm is implemented in the ABC system to generate the rideshare groups (Li et al., 2005).

The rideshare optimization system ensures that no rideshare group is worse off by participating in the rideshare mechanism. The rideshare group optimizer considers single-agent subsets as well as rideshare groups in the set-cover optimization, thus selects individual (initial) trips for some of the agents rather than assigning them into rideshares should no beneficial rideshare plan be available. Thus, any rideshare group generated by the mechanism offers non-negative cumulative utility to the agents.

Ensuring non-negative utility to rideshare groups does not guarantee individual rationality or fairness among agents in the rideshare system. The system may incur additional costs to drivers while generating benefits for passengers. The next section investigates payment mechanisms that can fairly divide the collaborative benefit generated by the rideshare optimization component among participants of the mechanism.

7.3 Mechanism Design for Collaboration

The payment mechanism is a crucial component of ABC's operations as it promotes collaboration among people and directly influences the user behavior and the efficiency of the system. Sharing fuel costs among passengers is a simple but widely used payment mechanism in ridesharing. However this simple payment scheme is not suitable for a personalized ridesharing system, because it does not consider varying agent costs in payment calculations. Using such a payment scheme in ABC would make the system vulnerable to deceptive reporting of needs by individual agents with the goal of biasing rideshare plans to satisfy their individual preferences.

Designing the payment component of a dynamic and personalized ridesharing system is a challenging problem. As stated by the impossibility theorem, no exchange mechanism can be efficient, budget balanced and individually rational (Myerson and Satterthwaite, 1981). Moreover, computationally expensive payment calculations may not be feasible for a dynamic system. This work focuses on VCG-based payments as they promote truthful behavior and individual rationality and adapt to the changing preferences of agents, in contrast to simple payment methods such as basic cost sharing. This section presents the initial VCG-based payment mechanism and then explores the tradeoffs with applying the mechanism within the ABC prototype in terms of efficiency, computational complexity, budget balance and individual rationality.

7.3.1 VCG Payments for Rideshare

ABC's payment mechanism distributes VCG-based payments to promote truthful behavior, to ensure fairness and sustainability of the system, while maximizing the cumulative value of the collaboration (Clarke, 1971; Groves, 1973; Vickrey, 1961).

Agent p_i's VCG payment to the system, denoted as ρ_i, is calculated as below, given that V^*_{-i} is the collaborative value of the collection of rideshare plans SC^* to all agents except p_i, $(V_{-i})^*$ is the value of the collection of rideshare plans that would be generated when p_i is excluded from the ABC system:

$$\rho_i = (V_{-i})^* - V^*_{-i} \tag{7.12}$$

If the rideshare plans generated by the optimization component are optimal, the VCG payment mechanism is efficient–its output maximizes social value, is individually rational–all agents have positive utility by participating, and strategy proof–truth-telling is a dominant strategy.

VCG payments ensure truthfulness by aligning an agent's individual utility function to the group utility. If the participants of a collaborative activity receive VCG payments, they are incentivized to maximize their own utility by maximizing the collaborative utility. Thus, VCG payments is a mechanism for uniting self-interested agents under a joint utility function, as assumed by formal teamwork models and helpful behavior models presented in Chapter 3, and enables the use of these models in domains of self-interest.

7.3.2 Tradeoffs on VCG Based Payments

Pursuing the use of VCG payments to promote ridesharing immediately faces several challenges. First, the VCG payment mechanism is not budget-balanced and may result in a loss. Second, calculating VCG payments in a dynamic mechanism is computationally expensive. Third, VCG mechanisms require the computation of optimal outcomes to ensure truthfulness. Due to the complexity limitations of the dynamic rideshare system, the implementation of the system uses an approximate algorithm for computing rideshare group assignments and for generating rideshare plan routes. VCG payments computed based on these suboptimal rideshare plans no longer guarantee truthfulness among agents participating in rideshare plans (Nisan and Ronen, 2007).

The VCG payment scheme is modified to adapt to the dynamic requirements of the open-world ridesharing problem. To simplify the analysis, it is assumed that removing one agent from a rideshare group does not affect the rideshare allocation of agents outside of that group. Based on this assumption, VCG payments are computed locally; the payment of agent, denoted as p_i, is computed only among the agents that share the same rideshare plan as p_i. This assumption makes payment calculations significantly more efficient, as rideshare plan optimizations for payment calculations are done over a small subset of all agents. Calculating VCG payment locally offers an alternative for efficient calculation of VCG-based payments, by pointing out an important tradeoff for implementing expensive payments efficiently. However, the locality assumption for calculating VCG payments efficiently is not fundamental, does not affect the collaborative rideshare plans and can be ignored if sufficient computational power is provided to compute payments globally.

The local VCG-based payment scheme is tested on a large dataset of GPS trails that are described in detail in Section 7.4. The experimental results show that value distribution with local payments maintains 99.7% to 100% of individual-rationality among agents with varying fuel and time costs. However, the evaluation highlights the prospect of incurring a deficit with VCG-based payments. The study identifies that the system pays drivers more than it collects from the passengers. To sustain the rideshare system with local VCG-based payments, the system runs into a deficit in the varying amounts of 55% to 79% of the cumulative value generated with rideshare plans. The deficit of the system grows proportionally to the average time costs of the agents, as it gets harder to bring self-interested agents together when time cost is high.

Given the challenge of balancing the budget, the payment calculations are revised to use an alternate VCG-centric scheme based on previous work by Parkes et al. (2001) proposing a threshold-based mechanism that enforces budget-balance as a hard constraint on payment calculation. The local VCG-based payment scheme is updated as presented below to eliminate deficit, where V^* is the cumulative value of rideshare plans and Vickery discount $\Delta_{vick,i}$ denotes the non-negative portion of VCG payments.

$$\Delta_{vick,i} = V^* - (V_{-i})^* \tag{7.13}$$

For some parameter $C \geq 0$, threshold discount, denoted as $\Delta^t_{vick,i}$, is defined as below, and payment of agent i to the system, denoted as ρ^t_i, is redefined based on $\Delta^t_{vick,i}$. The threshold parameter C is calculated with linear programming based on local VCG-based payments and $\Delta_{vick,i}$ values.

$$\Delta^t_{vick,i} = max(0, \Delta_{vick,i} - C) \tag{7.14}$$

$$\rho^t_i = v_i(SC^*) - \Delta^t_{vick,i} \tag{7.15}$$

Experiments with the real-world commute dataset using the local VCG-based payments with the threshold rule demonstrate that the revised mechanism is able to eliminate the deficit for a range of time and fuel-cost values. The mechanism does not affect adversely either individual rationality or the efficiency of the ABC system.

The payment component of the ABC system is designed not to overburden people by inquiring about the utility of each potential rideshare assignment. Instead, valuations are generated by the system based on acquired preferences.

With threshold-based payments and suboptimal rideshare plans, the mechanism is not guaranteed to be truthful. Investigating the effect of using local payments and the threshold rule on the truthfulness of agents will require deeper investigations into the system. Parkes et al. (2001) shows that the threshold-based payment scheme has better incentive properties than other rules. The threshold-based local VCG payments proposed in this work promote truthful behavior as an agent's payment does not directly depend on its preference revelation. The payment scheme is hard to manipulate by bounded-rational agents given the incomplete information available to agents about other agents and the indirect effect of an agent's preferences on outcomes.

7.4 Real-World Commute Dataset

The ABC system is empirically evaluated on commute data gathered from 215 subjects over a five-year period (Krumm and Horvitz, 2005). These subjects included Microsoft employees and spouses who volunteered to place GPS receivers with logging in their cars over several weeks in return for participating in a lottery for a prize. Nearly all the subjects live in the Seattle, WA USA area. The GPS receivers were programmed to record GPS data only when the subjects are in motion. The dataset contains a total of 1,434,308 (latitude, longitude) points for an average of 6,671 points per participant.

As the initial goal of this research is to generate rideshare plans for daily commutes of users, the dataset is segmented into discrete trips. Any two consecutive GPS points that are either 5 minutes or more than 7 kilometers apart are identified to belong to two separate trips. The trips that are shorter than a threshold value are eliminated, which results in 7,377 individual trips. For each user, a pair of morning and evening trips is selected to capture daily commute patterns of the users, based on the following properties: (1) the regularity of the commutes on trip data of the user, (2) minimum divergence of the selected commutes from a round trip. 215 morning$\backslash$evening commute patterns were extracted with average durations of 26 mins for morning, 29 mins for evening, and average distances of 21km for morning and 24 km for the evening.

7.5 Empirical Evaluation

The ABC prototype provides options for offline, batch optimizations as well as for real-time simulations of incoming ride requests based on the dynamic queuing of travel needs and preferences. Statistics are maintained on multiple dimensions of cost and savings for gaining insights into the operation and sensitivity of the plan generation to different workloads and assumptions. The system also provides visualizations of routes and route plans on a city map.

The ABC prototype is tested on commute patterns extracted from the commute dataset. The results of the rideshare system are evaluated in terms of the *efficiency in number of commutes* (i.e., the reduction ratio on total number of commutes), the *efficiency in cost* (i.e., the reduction ratio on total cost), and the *reduction of CO_2 emissions*. These empirical evaluations explored the sensitivity of the analyses to variations in the fuel costs (i.e., from \$0.035/mile to \$0.14/mile) and the average costs of time (i.e., from \$0/hour to \$9.6/hour).

Figure 7.2 compares the individual commute plans with the collection of rideshare plans generated by the system. In the figure, thicker lines illustrate crowded routes. The figure on the left illustrates morning commutes without the ABC system. The figure on the right illustrates morning commutes with the ABC system. The thinner lines on main highways in the right figure indicates the positive effect of ridesharing on the morning commute traffic in the Seattle region. The rideshare system reduces the number of cars in the traffic significantly. When the fuel cost is set to \$0.07/mile, [2] and average time cost is set to \$4.8/hour, the ABC system is able to achieve 41% efficiency on number of commutes, 14% efficiency on total cost of transportation which results in 84.16 tons of CO_2 reduction

[2] \$0.07/mile is stated to be the average per mile cost of driving by http://www.commutesolutions.org/calc.htm.

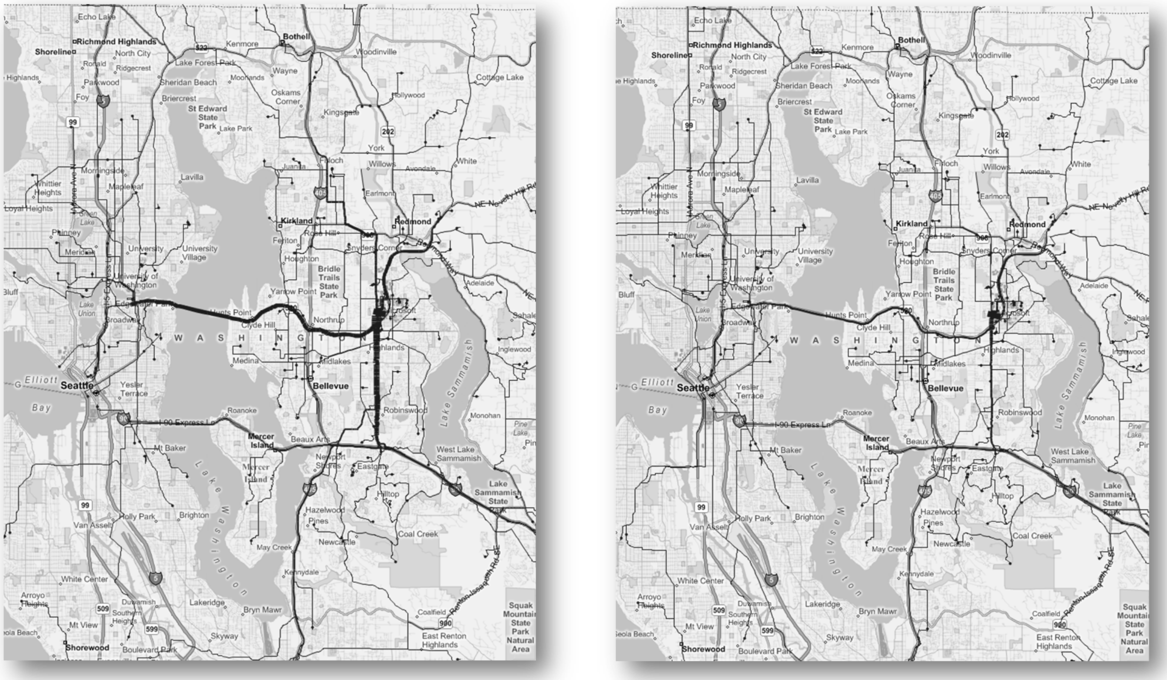


Figure 7.2: Seattle area map displaying the commute routes of study participants.

per year.

To investigate the influence of the cost of fuel on the value generated by the rideshare system, the system is tested over a range of fuel costs. As shown in Figure 7.3, the efficiency of the rideshare system on both the number of commutes and the total cost improves significantly with increases in the cost of fuel. These results indicate that increasing fuel costs can provide higher incentives for agents to collaborate. The willingness of agents to rideshare is expected to grow as fuel costs increase. The reduction in CO_2 emissions increases 25% as fuel costs increase from 0.035/mile to \$0.14/mile.

The influence of changes in the cost of time on the efficiency of the rideshare system is investigated by varying the average time cost of users as shown in Figure 7.4. As the cost of time increases, the efficiency of the optimization with regard to the number of commutes

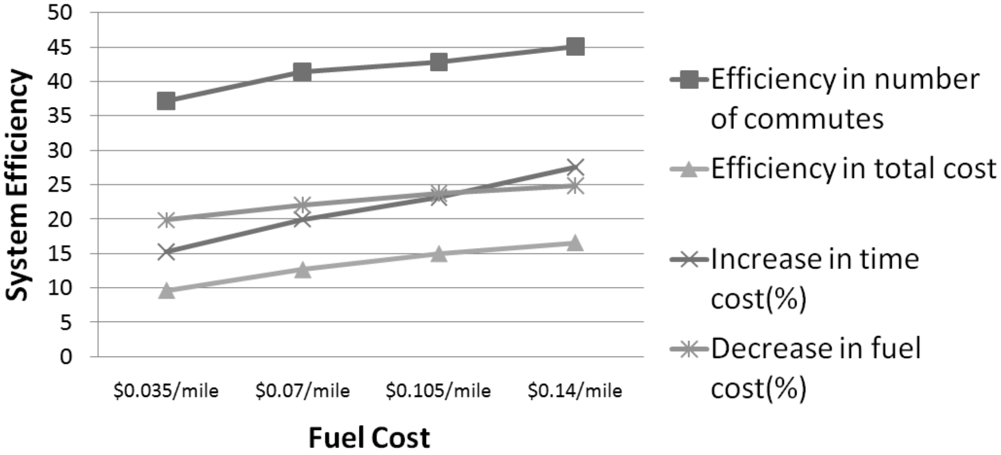


Figure 7.3: Effect of fuel cost on the efficiency of the ABC system.

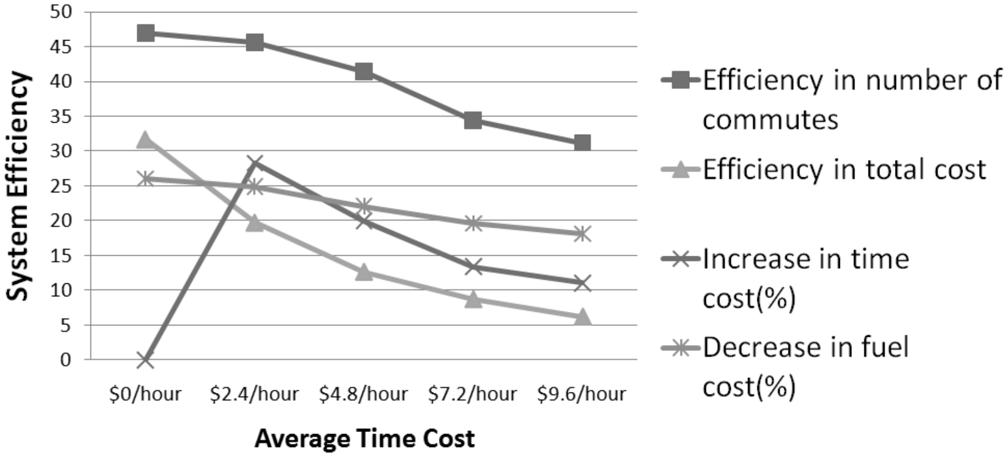


Figure 7.4: Influence of the average time cost on the efficiency of the ABC planning.

and total costs incurred drops significantly. The reduction in CO_2 emissions decreases by 29.6%. Increasing time costs makes it harder to bring self-interested agents together in rideshare plans.

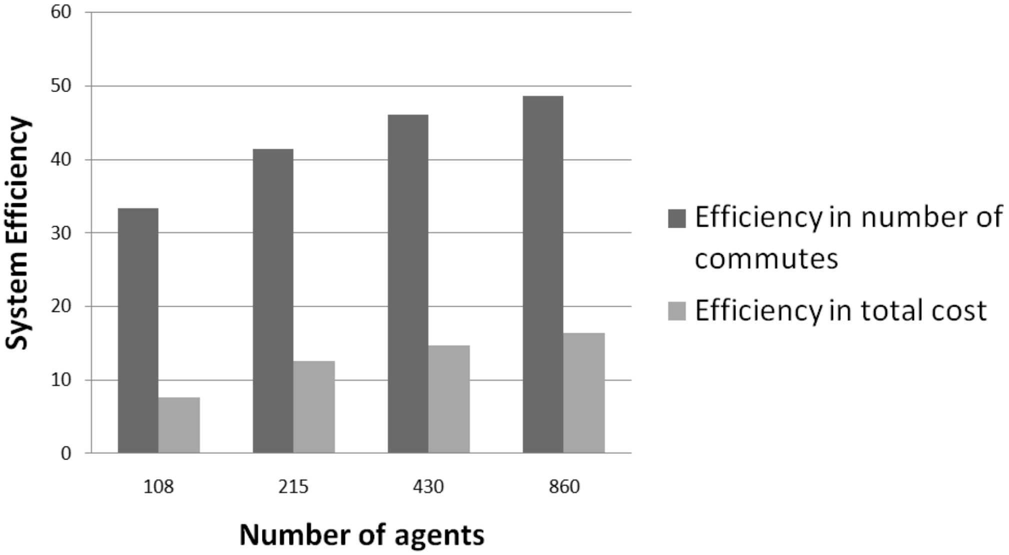


Figure 7.5: Effect of the size of agent set on the efficiency of the ABC planning.

The next set of experiments seeks to understand how the density of participants may change the behavior and overall savings generated by the system. To simulate the effect of increasing the number of agents in the system, the set commute patterns are populated with randomly created artificial commute patterns. The synthetic commuting requests are generated by pairing randomly selected start/end points from the commute dataset, with trip start times taken from a Gaussian distribution representing the start times of the commute patterns in the data. As illustrated in Figure 7.5, the efficiency of the system grows as a logarithmic function of the number of agents in the system. With more agents, the system is more likely to find better matches. Thus, the performance of the rideshare system is expected to improve with increasing numbers of agents.

7.6 Real-World Considerations

The work presented in this chapter focuses on computational methods for bringing self-interested people together in collaborative plans in a dynamic system. In particular, it uses the domain of ridesharing to demonstrate these methods on a real-world application. However, sustaining the success of a live collaborative mechanism such as ridesharing requires thinking about social factors and contingencies that may affect the mechanism. This section highlights and discusses major real-world considerations that may affect the success of a system in which people collaborate.

Providing fair and satisfactory incentives is crucial for the success of a collaborative activity. The participants of an activity may bear additional costs for doing a part of the activity that may benefit other participants and they must be satisfied with the compensation they receive. The payment component of the ABC system distributes incentives in the form of monetary payments. In the mechanism design literature monetary payments are mostly used in electronic commerce (Varian, 1995). When it comes to providing incentives in dynamic mechanisms that manage services (e.g., bandwidth management, p2p services), non-monetary incentives that affect the quality of service (e.g., improving data quality, increasing the bandwidth) have been proposed as an alternative (Dash et al., 2003). Non-monetary payments can be introduced to the rideshare domain in the form of better parking spots or carpool-lanes in highways. However, non-monetary payments may not compensate for extra gas or the time costs drivers may bear. On the other hand, monetary payments may cause unwillingness among the participants to join the system. Future user studies on the ABC system will provide more insight about the way human behavior is affected with respect to different types of incentives and payments. Understanding how people perceive different payment mechanisms and incentives may result in designing more successful systems in which people collaborate.

People participating in a collaborative system may be initially hesitant to give the complete control to a fully autonomous system to make decisions on their behalf and determine monetary incentives they receive. A good strategy may be to design a collaborative system as an interactive mechanism that continuously learns about people and asks for input from the participants of the system. For example, the current design of the ABC system is fully autonomous in the sense that the rideshare plans and payments are computed and dictated to users, under the assumption that human users completely agree with the decisions of the system. Nevertheless, the system may benefit from giving some control to users, especially if monetary payments are involved. Users may get notified about rideshare plans before their plans are finalized, or the system may ask for an approval before committing to a high payment. Sharing control with users may be particularly important during the trial period of the system to generate trust when users do not completely understand the way the system works. Moreover, user input may be beneficial to better understand user preferences (e.g., time and cognitive costs). Incorporating a mixed-initiative component that trades off the cost of interrupting the user with the benefit of this interaction may improve the performance of the system without overburdening users (Horvitz et al., 2004).

When people get involved in a collaborative system in real life, it is expected that there will be contingencies in which some participants willingly or unexpectedly fail to carry out their commitments. A collaborative system should be designed to recover from such contingencies and to disincentivize such events happening intentionally. In the ABC system, a driver may fail to pick up passengers, or a passenger may not show up. Such users are referred to as deviators. The system has built-in rules for recovering from these contingencies and for deterring users from failing to honor their commitments. The ABC payment component has a punishment module that determines how much a user needs to pay in case of failing a commitment. If a rideshare plan fails, the system runs the optimization component again by excluding the deviator, constructs updated rideshare plans, and notifies users. If no rideshare is available for some of the users, the system backs up to taxi/shuttle services. The punishment of a deviator is the difference of the utilities of all users excluding the deviator between the original and the updated rideshare plans. The deviator pays the additional burden on all other users for failing the commitment.

Although the ABC system currently has rules for punishing unwanted behavior, failures may be inevitable in a dynamic setting. Punishing participants harshly in case of an unintentional failure may hurt the willingness of agents to participate in the system. It is a challenge for future work to design punishment mechanisms that provide incentives to agents to keep their commitments without scaring them away.

The success of a collaborative system is likely to depend on multiple social and psychological considerations. Although the focus of this thesis is not on these issues, it is important to note that they are crucial for the wide deployment of such a system. A collaborative system may significantly benefit from social networks, trusted organizations and organizational membership to generate collaborative plans that participants are comfortable with. It may be possible to design special incentives that depend on the economics within organizations. For instance, the architecture of the ABC system can be further improved with the addition of a reputation mechanism that helps to distinguish reliable users from deviators. The payment mechanism can be further broadened to include organizations as participants of the system and by considering their organizational values for members collaborating on rideshare plans.

The design of the ABC system considers many social factors and contingencies that may affect the success of a collaborative activity. However, the successful design and implementation of a collaborative system in real-life with human agents requires understanding the way people interact with these systems and perceive different incentives. Future work is needed to investigate these issues in detail.

Chapter 8

Related Work

8.1 Formal Models of Teamwork

Collaboration is a special type of group activity in which participants work together toward a shared goal, typically the performance of a collective action. Modeling collaboration requires special attention. As stated by Grosz (1996), collaboration is more than the sums of individual plans. Agents need to form mutual beliefs and intentions to be in a collaborative activity, and formal models of collaboration need to be designed accordingly.

Several formal models of teamwork have been proposed to identify and model the special characteristics of collaborative activities. The *Joint Intentions* model (Cohen and Levesque, 1991; Cohen et al., 1997) defines joint commitment and joint intentions as a requirement for collaborative activity, and studies the way they are associated with individual intentions of agents. This model does not consider partiality of a collaborative plan in a comprehensive way and does not study its constituent components in detail.

Sonenberg et al. present a detailed specification of means-end reasoning components needed for forming collaborative plans, based on agents' mutual beliefs, joint goals, joint plans and joint intentions (Kinny et al., 1992). However, they do not study partial plans nor do they explicitly handle uncertainty or incomplete information in planning.

Jennings (1992) designed a testbed environment for cooperative multi-agent systems for the domain of electricity transportation management based on a formal model of joint intentions. This system involves a central planner for assigning agents to actions and choosing recipes accordingly. It allows limited partiality of collaborative plans in terms of delayed agent assignment and timing, but not in terms of the decomposition of actions.

In contrast to other prominent teamwork formalizations, the SharedPlan formalization (Grosz and Kraus, 1996) does not require a special kind of intention (joint intentions) among agents, but provides a specification of mental states of agents based on their beliefs, mutual beliefs and ordinary intentions. The formalization embraces the dynamic nature of the real-world and allows partial plans to get updated and completed over time. Due to this partiality, the specification is comprehensive in its handling of means-end reasoning and in its description of plan constituents.

8.1.1 Communication and Helpful Behavior Requirements

Jennings (1992) presents a set of experiments that examined the benefit of communication for collaborative activities. These experiments show that communication among team members helps to reduce wasted work and to recover from mistakes and unexpected events.

Consequently, several teamwork formalizations have axiomatized decisions about whether to communicate or to help. The Joint Intentions formalization defines communication as a crucial requirement for successful teamwork, and it presents strict axiomatic rules about when to communicate (Cohen and Levesque, 1991; Cohen et al., 1997). According to this formalization, whenever a joint goal is satisfied or found impossible to achieve, agents are required to commit to make this mutually known, leading to communication among agents. Similar to the Joint Intentions formalization, Kinny et al. (1992)'s definition of joint intentions leads to communication among agents in collaborative activities. During plan construction and execution, agents are required to broadcast the success or failure of a subaction. Fan et al. (2005) extend the set of logical axioms for communication to provide for proactive information exchange (informing and asking for information). However, these approaches do not consider the cost or benefit of communication nor do they provide mechanisms for helping actions that improve the utility of plans.

The SharedPlan (SP) formalization includes axioms that entail adopting intentions for helpful acts or lead to communication, based on certain kinds of intentions in the SP specification (Grosz and Kraus, 1996). These axioms allow more flexible behavior. Agents are not required to communicate whenever an action fails or succeeds, but communication can be used at will to make the situation mutually known. The axioms represent both the benefit of a helpful action to the group activity and the costs to the individual performing the helpful action. However, they do not handle uncertainty regarding the world or agents' capabilities. Furthermore, the specification provides no insight on how these axioms can be realized or implemented in agent design, whereas this thesis provides a decision-making mechanism.

STEAM, which drew on both the joint intentions (Cohen and Levesque, 1991; Levesque et al., 1990) and the SharedPlan (Grosz and Kraus, 1996) formalizations, supports the construction of agents able to collaborate in complex, real world domains of military training and robot soccer (Tambe, 1997). It includes a decision-theoretic mechanism for communication which models the cost-benefit trade-off associated with communicating information to the full group. In STEAM, a decision-tree is constructed for each agent every time a communication action is considered, with significant complexity costs for agents that need to consider many such actions. Consequently, the mechanism provided in Chapter 3 for decision-making has lower complexity requirements and is more general than these previous decision-tree approaches.

Work on decentralized approaches to multi-agent planning has provided models that consider the cost-benefit trade-off of communication among agents (Goldman and Zilberstein, 2004). As helpful behavior can emerge between any agents in a collaborative activity, helpful behavior (e.g., communication or helpful acts) needs to be directly embedded in the joint policy of the whole group of agents, making it exponential in the size of the history of agents' observations. Refining agents' plans in this setting means updating their entire policy every time a helpful action is considered, which is infeasible.

8.1.2 Applications

The various formalizations have been used as a foundation to build teamwork applications in various domains. Lochbaum (1994) presents a direct realization of the SharedPlan formalism as a dialogue system. Jennings (1995) used the Joint Intentions formalism as a basis for agent design in building the $GRATE^*$ system. These implementations demonstrate the usefulness of formally modeling agents' mental states in collaborative activities. They use axiomatic rules to specify the way agents should make decisions and act in the world, without explicitly modeling utilities, probabilities or uncertainties.

Rich and Sidner (1997) used the SharedPlan formalism to build the COLLAGEN system, which includes software interface agents for applications such as air-travel arrangement. This system does not explicitly model the way communication emerges from the intentions of agents nor does it apply SharedPlan's axiomatic rules for managing communication.

8.2 Decision-theoretic Models for Collaborative Decision-Making

The literature on artificial intelligence includes significant work on collaborative decision-making and planning. One important example is teamwork formalizations based on belief-desire-intention (BDI) models which have served as a foundation for multi-agent coordination (e.g., Cohen and Levesque (1991); Grosz and Kraus (1996)). Although these models emphasize the importance of communication and coordination decisions on the success of teamwork, they do not present decision-theoretic models that consider the uncertainty of real-world domains and the costs and benefits of such actions. There has been growing interest in multi-agent models of Markov Decision Processes (MDPs). These models include Multi-agent Markov Decision Processes (MMDPs) (Boutilier, 1999), Communicative Multi-agent Team Decision Problems (COM-MTDPs) (Pynadath and Tambe, 2002), Decentralized Markov Decision Processes (Dec-MDPs and Dec-POMDPs) (Bernstein et al., 2002). Among these models, Dec-MDPs have been widely used to model many multi-agent decision-making problems in which the world is uncertain and dynamic, and agents may have partial information about the world and each other. However, the complexity of solving Dec-MDPs is proven to be NEXP-hard and believed to take doubly exponential time to solve optimally, even for problems involving two agents (Bernstein et al., 2002). In practice, general solution techniques for Dec-MDPs are shown to be infeasible even for small size toy problems (Pynadath and Tambe, 2002). Moreover, the complexity of finding ϵ-approximate policies for Dec-MDPs is also shown to be NEXP-hard (Rabinovich et al., 2003).

Exploiting Structure

These discouraging results have motivated researchers to exploit characteristics of different domains and identify classes of decentralized decision-making problems that are easier to solve. The complexity of solving a transition-independent and reward-independent Dec-MDP is the same as the complexity of solving a single-agent MDP model. Transition-independent decentralized MDPs model a more comprehensive class of problems in which agents operate independently, but tied together with a reward structure that depends on the joint histories of agents. Becker et al. (2003) present an algorithm that solves transition-independent Dec-MDPs optimally. Subsequently the authors propose an approximate algorithm for solving Dec-MDPs that uses communication actions as a way to decompose joint decision making into individual policies by assuming that agents act individually between communication actions. This heuristic search algorithm converges to an optimal decomposition, under the assumptions that every time agents interact, they sync and share the complete state of the world with each other and the cost of communication is always fixed, independent of the state of the agents (Goldman and Zilberstein, 2008).

Seuken et al. (2008) define partially synchronized Dec-MDPs (PS-DEC-MDPs) that share similar independence properties on reward and transition functions as the ND-MDP model presented in this thesis. The independence properties of PS-DEC-MDPs emerge from agents being periodically inaccessible. PS-DEC-MDPs differ from ND-MDPs in that they do not distinguish actions agents perform jointly or independently, but they distinguish states in which agents are accessible. Thus, a policy generated for a PS-DEC-MDP determines how agents should act when they are inaccessible rather than deciding when and how they should act together.

Another method for exploiting the structure of a Dec-MDP is using factored representations. Factored representations distinguish the regions of the state space where agents act individually and where they need to coordinate with each other (Oliehoek et al., 2008; Roth et al., 2007). This approach provides an efficient solution if the domain of interest has conditional and context-specific independence properties. The algorithm for generating factored policies assumes free communication among team members (Roth et al., 2007). It uses query functions, not to capture value functions of other agents as proposed by the ND-DECOP Algorithm, but to capture unknown features of the world state. In subsequent research, heuristic algorithms have been proposed to determine when and what to communicate. These algorithms use greedy search to select communication actions with the highest expected benefit (Roth et al., 2005, 2006). They make the assumption that cost of communication depends on the length of communication, not on the joint state. These algorithms focus on managing interactions for robot soccer and bandwidth optimization domains, rather than interruption management in which communication decisions depend on cognitive states and performances of agents.

Modeling Communication Explicitly

Decentralized MDPs implicitly represent communication actions, as these actions can be included into the action space of agents. A Dec-MDP with Communication (Dec-MDP-Com) expands the general formalization with an explicit communication action space and a communication cost function that maps a communication action (message) to a real number (Goldman and Zilberstein, 2003). The Dec-MDP-Com formalism assigns varying costs for different communication actions, but assumes a fixed cost for a particular communication action regardless of the states of the agents. Dec-MDP and Dec-MDP-Com models are equivalent both in their expressiveness and complexity.

Shen et al. (2006) have shown that the complexity of multi-agent decision-making is highly correlated with the complexity of interactions between agents in a multi-agent setting. When communication is free, the complexity reduces to the complexity of solving single-agent problems, as agents can freely communicate at each time step and act as a single entity (Pynadath and Tambe, 2002). When communication is limited to a synchronization based communication protocol, in which all agents communicate simultaneously to unify their world views with a single communication cost, the complexity of finding an optimal policy for a transition-independent decentralized MDP is shown to be NP-hard (Goldman and Zilberstein, 2004). However this form of communication is unrealistic in many domains where bandwidth is limited or the cost of communication depends on the states of agents. Managing general communication among multiple agents in partially observable domains is shown to be NEXP-hard (Pynadath and Tambe, 2002).

Approximate Algorithms

Due to the infeasibility of generating optimal policies for Dec-MDPs, approximate algorithms have been proposed as an alternative. A polynomial-time algorithm presented by Beynier and Mouaddib (2005) approximates the joint policy by building individual agent policies that are connected with temporal and resource constraints. This algorithm assumes that individual agent models are fully observable and thus communication is not necessary among agents. Xuan et al. (2001) propose an approximate algorithm that uses heuristic functions to compute the expected benefit of communication by comparing information gain with the cost. These heuristic methods are empirically evaluated, but do not guarantee performance bounds.

Comparison of ND-DECOP with Previous Approaches

Modeling interactions between computer agents and people offers new challenges that are not addressed by the previous work in the Dec-MDP literature which focuses on homogeneous computer agent groups and assumes that communication is either free or associated with a fixed cost. When computer agents and people interact, the cost of communication (interaction) depends on the tasks both the person and the agent are performing, the person's cognitive state and attention level, and the effect of communication on their individual tasks. The ND-MDP model presented in this thesis is a special type of a Dec-MDP model that is designed for situations that may include groups of computer agents and people and the way they interact. The ND-DECOP algorithm offers a solution to this new communication problem that may include people, and focuses on managing joint actions based on the joint state, where the joint actions may also include communication actions.

An optimal policy generated by the algorithm initiates interactions if and when estimated to be most beneficial for the collaborative group based on the joint state.

The domains represented by the ND-MDP model (i.e., nearly-decomposable problems) have special characteristics that cannot be represented with transition-independent Dec-MDPs. In a nearly-decomposable problem, agents' transition functions are not fully independent, but conditionally independent. Their individual task performances depend on the joint actions that they perform together. However, for the cases in which they act independent, their tasks and decision making processes are independent.

The focus in this thesis has been on exact algorithms instead of approximate algorithms. A crucial requirement for the work presented in Chapter 6 is to be able to capture the outcome of an interaction correctly, so the outcome of the ND-DECOP and DECOP decision-making models can be used as a baseline in the empirical analysis of human behavior. It has been shown in this thesis that the policy generated by the ND-DECOP algorithm for a given ND-MDP is optimal, and the DECOP algorithm can compute the actual outcome of interruption accurately when an interruption is initiated for single-shot interruption scenarios. If this outcome is not correctly calculated, the differences between human responses and the decision-making algorithm might have been caused by the sub-optimality of the algorithm as well as the human characteristics of decision making, and this ambiguity might make it impossible to reach conclusive results about the characteristics of human behavior. Due to this necessity for optimal policies, this work focuses on building efficient optimal algorithms for special domains, rather than building approximate methods for solving general Dec-MDPs.

8.3 Interruption Management

The investigations presented in this thesis use interruption management as an example of a decision making capability needed for collaborative activities in which agents are distributed, conditions may be rapidly changing and decisions are made under uncertainty. Interruption Management has been studied extensively in different literatures including AI, psychology, HCI and management.

Negative Effects of Interruptions

Spontaneous communication among team members is useful for a collaborative activity, providing rich and up-to-date information about the tasks at hand (Dabbish and Kraut, 2004). However, communication leads to interruptions, which are inherently disruptive. If they are not managed and timed properly, they may negatively affect decision-making quality (Speier et al., 1999) or the emotional state and awareness of the user, and thus may reduce the overall task performance of the user and the system (Adamczyk and Bailey, 2004). Cutrell et al. (2001); Czerwinski et al. (2000) point out that even ignored interruptions can be disruptive for users. A study focused on software engineers shows that interruptions, at extreme cases, cause lost productivity and even set back production cycles (Perlow, 1999). Similar negative results have been reported for aviation (Dismukes et al., 1999) and office domains (Avrahami and Hudson, 2004; Cutrell et al., 2001).

People interact with a growing number of notification systems everyday (e.g., email, IM, phones, SMS). Consequently, they receive increasing number of interruptions which may cause prospective memory failure (i.e., forgetting a task that needs to be performed) and additional costs for task switching. Interruptions make it harder for people to manage their attention (Czerwinski et al., 2004; González and Mark, 2004). According to a field study by Iqbal and Horvitz (2007), an office worker on average is interrupted by 4 email alerts and 3 IM alerts per hour, and it takes 10 minutes to return to the suspended task after an interruption. Despite the negative effects of interruptions, another field study by Iqbal and Horvitz (2010) reports that office workers find notification systems useful.

Approaches for Managing Interruptions

Researchers have taken different perspectives in deciding whether and when to interrupt a user. Voida and Mynatt (2009) propose an activity-based computing approach that allows interruptions of relevance to the active task. This approach assumes that no interruption that is irrelevant to the task at hand is beneficial. Moreover, it does not consider the cost or benefit of an interruption while making interruption decisions.

Prior work on interruption management has addressed user needs and has focused mostly on the effect an interruption has on a person's cognitive state. McFarlane (2002) outlined four strategies for deciding when to interrupt; immediate (immediately delivering a notification), negotiated (user choosing the delivery time), mediated (an agent mediating in between), and scheduled (delivering notifications within a predetermined schedule). The results show that none of the proposed strategies is the single best approach,and thus that managing interruptions properly is a complicated problem that requires more sophisticated approaches.

Several research groups have built complex, computational models to predict the interruptibility of a user based on social and task-based features (Fogarty et al., 2005; Horvitz and Apacible, 2003; Horvitz et al., 2002; Hudson et al., 2003). Based on predictive models, decision-theoretic models have been proposed to time interruptions properly by considering the cost of interruption and the cost of delaying a notification (Horvitz et al., 1999). However these approaches perform a single-sided analysis of the benefit of interruption and ignore the benefit to the interrupter from the interaction.

In contrast, previous work on adjustable autonomy focuses on the system perspective. It identifies the points at which it is most suitable for the system to initiate interactions with a person, but does so without relating this decision to a user's mental state or the task being performed. Interruptions are driven solely by system needs and managed based on the benefit to the system (Scerri et al., 2003).

In contrast to traditional approaches in interruption management and adjustable autonomy literature, reasoning about interruptions in collaborative settings requires the ability to accurately estimate the costs and benefits of the interruption to all parties so that the outcome of the interruption positively affects group task outcomes. Few models have combined these two aspects into an integrated decision making mechanism (Fleming and Cohen, 2001), and none have done so in the kinds of rapidly changing domains with uncertainty considered in this thesis.

Human Perception of Interruptions

While there has been significant work on mixed-initiative system design, there has been little empirical work on how people perceive interruption utilities and make interruption decisions in human-computer interaction settings. Avrahami et al. (2007) investigated the differences between a person's self report of interruptibility and other people's predictions about that person's interruptibility. However, this work considered face to face human interaction, rather than human-computer interaction. Gluck et al. (2007) focused on designing notification methods to increase human perception of utility, whereas Bunt et al. (2007) showed that displaying system rationale to people may induce a person to trust a computer system more. In a separate study, it has been shown that the way a notification system is perceived at the initial stages of interaction is an important determinant for the success of further interactions (LeeTiernan et al., 2001). Survey studies in office domains show that interruptions on average are perceived to be more beneficial for interrupters (Kraut and Attewell, 1997; O'Conaill and Frohlich, 1995). Rudman and Zajicek (2006) point out that providing useful information is not necessarily enough to please users and thus computer agents should aim at initiating interactions that are perceived to be useful by users.

8.4 Collaboration of Self-Interested Agents

Altruism and reciprocity have been proposed as strategies to explain and promote cooperation among people when they interact repeatedly (Bowles and Gintis, 2005). However, in dynamic domains considered in this thesis collaborative plans may be generated on the fly, and the plans may change with respect to agents' dynamic preferences and the uncertainty in the world. For example, collaborative carpool plans may change with respect to changing preferences of users (e.g., trip start times, meeting schedules, cost of delay), and drivers may be paired with different set of passengers each day. Therefore, cooperation strategies such as altruism and reciprocity that require continuous interactions are not typically valid for dynamic domains.

Coalescing rational agents into groups of participants in rideshare plans is similar to the *initial-commitment decision problem* (ICDP) proposed by Hunsberger and Grosz (2000), as both problems aim to determine the set of tasks that agents need to commit in a collaboration. The methods employed in the ABC system are an extension to prior work on ICDP as a payment mechanism is included, which provides a rationale and incentives for self-interested agents to collaborate.

Several previous studies on set-cover optimization problems focus on mechanism design for cost sharing (Devanur et al., 2005; Li et al., 2005). The cost sharing problem focuses on dividing the cost of a service among self-interested agents in a fair manner, where the cost is independent of agents' preferences. The optimization used in ABC makes use of greedy optimization procedures similar to the approach taken in the earlier set-cover optimization efforts. However, the payment mechanisms employed in the past are not suitable for collaboration among self-interested agents. The domains in which self-interested collaborate do not have a distinction between service providers and receivers, and the cost is not independent of agents' preferences.

Mechanism design has been applied to the coordination of self-interested robots in sequential decision-making scenarios (Cavallo et al., 2006). The work presented in this thesis differs from the prior work in that both the joint plans and payments are based on combinations of dynamic and changing preferences of people about their daily habits including time, fuel, cognitive costs and travel preferences. This thesis also presents a detailed analysis of the optimization and payment mechanisms with respect to the computational issues that arise when they are evaluated on real-life data in a dynamic domain.

Chapter 9

Conclusion

This thesis presents theoretical and empirical investigations of representations and decision-making strategies needed by computer agents in settings in which they collaborate with people and with other computer agents. These representations and decision-making strategies contribute to the design of computer agents that are effective partners to people. The challenges that are addressed arise from the uncertain and dynamic nature of real-world domains and the fact that an agent participating in a collaborative activity may have partial information about the world and the way the activity is being accomplished. This thesis defines new representations for handling this uncertainty. It shows empirically that decision-theoretic models which make use of these representations to reason about the costs and benefits of doing an action on the collaborative utility helps to improve the success of a collaborative activity. It demonstrates the value of these collaborative teamwork ideas in a real-world domain and highlights the challenges that arise in applying these ideas in dynamic settings.

A major contribution of this thesis is Probabilistic Recipe Trees (PRT), a probabilistic representation for agents' beliefs about the way a collaborative activity is being accomplished. This representation formally incorporates costs, utilities and uncertainties into formal teamwork theories and enables decision-theoretic reasoning on them. The PRT representation is shown to be exponentially more compact than an exhaustive belief representation, and its modular structure enables efficient updates to reflect the changes in agents' beliefs. This thesis demonstrates the usefulness of the PRT representation for enabling decision-theoretic reasoning on teamwork models with a decision-theoretic mechanism that agents can use to decide whether to undertake helpful behavior. This mechanism is used by agents to make decisions about informing a partner about an observation, asking for information, doing a helpful action and abandoning commitment to perform an action. This thesis empirically investigates the performance of this mechanism in experiments that vary the cost of helpful behavior and the uncertainties of agents about the world and about their partners. The results show that agents using the decision-theoretic mechanism for deciding whether to help are able to perform better than the agents using axiomatic rules for all conditions.

A particular focus of this thesis is the settings in which computer agents collaborate with people and the way agents make decisions in such settings. The thesis presents efficient planning algorithms that evaluate the effect of actions on collaborative utility, and it investigates the way people perceive this utility as computed by the algorithms. It defines Nearly-Decomposable Markov Decision Processes (ND-MDPs). The ND-DECOP algorithm proposed for solving ND-MDPs distinguishes the cases in which agents act individually from those in which they act jointly, and it decouples multi-agent planning into individual models that are coordinated only when agents are acting jointly. The analysis of the algorithm demonstrates that it can achieve up to exponential savings in computation time while generating optimal policies for multi-agent planning problems that can be represented as ND-MDPs.

The way people perceive the fully rational collaborative utility as computed by planning algorithms is investigated in this thesis in a specially designed human-computer collaboration setting in which computer agents need to manage their interruption requests to humans. These investigations reveal that the actual benefit of interruptions to both computer agents and people is the major factor affecting the likelihood that people will accept interruption requests. However, for those cases in which the benefit of interruption is ambiguous, people prefer to accept those interruptions that originate from other people rather than computer agents.

Based on the empirical results showing that different factors may affect the way people make decisions, this thesis investigates if learning from human responses helps to better predict the way people interact with computer agents. It applies well-known learning algorithms to the data collected from the human studies to build predictive models of the way people respond to interruption requests. The empirical results show that learning improves the accuracy of computer agents' prediction of human responses when collaborative utility values as computed by the fully-rational computational models are provided as features for learning.

This thesis also introduces an application of collaborative teamwork ideas to a real-world domain of ridesharing. It presents a complete computational model that generates collaborative plans for self-interested users and computes incentives to guide them to collaboration. The computational model is tested on real-world data collected from a group of commuters. The empirical investigations show that collaboration offers significant value for individual users and the environment, and it reduces the number of vehicles on the road, the cost of transportation and gas emissions. They also highlight challenges and trade-offs that arise in applying teamwork ideas in real-world.

9.1 Future Work

The research presented in this thesis fills an important gap in formal models of teamwork by formally incorporating the notions of uncertainty, costs and utilities and integrating these formal models with decision-theoretic reasoning. The PRT representation of agents' beliefs about the way a collaborative activity is being done is the key to this integration. An interesting challenge for future work is the design of planning algorithms able to dynamically modify agents' beliefs on the PRT representation efficiently, including accommodating possible resource and temporal constraints that exist among constituent actions of a collaborative activity. Such algorithms would enable more sophisticated decision-theoretic mechanisms. A particular challenge that arises in the design of such algorithms is to determine the complexity of reasoning with PRTs with respect to the different types of dependencies that may exist among the constituent actions. Future studies are needed to understand the way different dependencies (e.g., temporal, resource constraints) can be introduced without drastically increasing the complexity of reasoning with PRTs.

The empirical investigations presented in this thesis are a first step in understanding human perception of collaborative utility in teamwork settings. Future studies are needed to investigate the effects of computational and cognitive complexity on the way people make short- and long-term decisions, focusing on the role of trust in the short and long term collaboration. Moreover, understanding the influences of social and organizational factors on collaborative teamwork will provide valuable guidance in designing collaborative systems for heterogeneous agent groups.

There exist numerous opportunities in the real-world for applying collaborative teamwork ideas to generate value for individuals and the society. This thesis presents an example application for the domain of ridesharing. The computational models for generating collaborative plans and providing incentives to collaborate can be generalized to many other areas, including sharing scarce resources such as energy among the members of the society based on their preferences; developing computational systems that can support collaboration among people to accomplish tasks that they cannot do individually; designing and developing computer agents that can reason about the activities being performed by people so that they can better interact and work with people and be effective partners for them to make everyday tasks easier. It is a challenge for future work to identify the issues that arise in applying teamwork ideas to any of these areas and to build on existing teamwork models, representations and algorithms accordingly.

A major challenge for real-world applications of teamwork is providing incentives to self-interested participants of a collaborative activity for the costs they burden for contributing to the activity. The properties of incentive mechanisms directly influences the way participants behave in a collaborative setting: whether they act truthfully and keep their commitments to the success of the activity. This thesis demonstrates the challenges and trade-offs that arise in implementing monetary payments in a real-world domain in terms of budget-balance, truthfulness and computational complexity. Designing a payment mechanism with the desired properties is even more challenging in dynamic domains in which some parts of the collaborative activity may stochastically fail. Future studies on payment mechanisms will provide valuable guidance to the design of collaborative systems involving self-interested agents. Moreover, in real-world collaborations among people, monetary incentives may not be the only factor affecting human behavior, but social and psychological factors may also play a role. Understanding the way people perceive different payment mechanisms and incentives may result in designing better collaborative systems for people.

Bibliography

P.D. Adamczyk and B.P. Bailey. If not now, when?: The effects of interruption at different moments within task execution. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 278. ACM, 2004.

D. Avrahami and S.E. Hudson. Balancing performance and responsiveness using an augmented instant messaging client. In *Proc. of ACM Conference on Computer Supported Cooperative Work, New York: ACM Press*. Citeseer, 2004.

D. Avrahami, J. Fogarty, and S.E. Hudson. Biases in human estimation of interruptibility: Effects and implications for practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 60. ACM, 2007.

T. Babaian, B.J. Grosz, and S.M. Shieber. A writer's collaborative assistant. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, pages 7–14. ACM New York, NY, USA, 2002.

R. Becker, S. Zilberstein, V. Lesser, and C.V. Goldman. Transition-independent decentralized Markov decision processes. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 41–48. ACM New York, NY, USA, 2003.

D.S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4): 819–840, 2002.

A. Beynier and A.I. Mouaddib. A polynomial algorithm for decentralized Markov decision processes with temporal constraints. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 963–969. ACM New York, NY, USA, 2005.

C. Boutilier. Sequential optimality and coordination in multiagent systems. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 478–485. Citeseer, 1999.

S. Bowles and H. Gintis. Can self-interest explain cooperation? *Evolutionary and Institutional Economics Review*, 2(1):21–41, 2005.

M.E. Bratman, D.J. Israel, and M.E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(3):349–355, 1988.

A. Bunt, J. McGrenere, and C. Conati. Understanding the utility of rationale in a mixed-initiative system for GUI customization. *User Modeling*, pages 147–156, 2007.

R. Cavallo, D.C. Parkes, and S. Singh. Optimal coordination of loosely-coupled self-interested robots. In *The Workshop on Auction Mechanisms for Robot Coordination, AAAI-06, Boston, MA*. Citeseer, 2006.

E.H. Clarke. Multipart pricing of public goods. *Public Choice*, 1971.

P.R. Cohen and H.J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.

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

P.R. Cohen, H.J. Levesque, and I.A. Smith. On team formation. *SYNTHESE LIBRARY*, pages 87–114, 1997.

W.J. Conover. *Practical nonparametric statistics*. Wiley New York, 1999.

E. Cutrell, M. Czerwinski, and E. Horvitz. Notification, disruption, and memory: Effects of messaging interruptions on memory and performance. *Proceedings of the IFIP Conference on Human-Computer Interaction*, pages 263–269, 2001.

M. Czerwinski, E. Cutrell, and E. Horvitz. Instant messaging and interruption: Influence of task type on performance. In *Proceedings of OZCHI*, pages 356–361, 2000.

M. Czerwinski, E. Horvitz, and S. Wilhite. A diary study of task switching and interruptions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 175–182. ACM, 2004.

L. Dabbish and R.E. Kraut. Controlling interruptions: Awareness displays and social motivation for coordination. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, pages 182–191. ACM New York, NY, USA, 2004.

R.K. Dash, N.R. Jennings, and D.C. Parkes. Computational-mechanism design: A call to arms. *IEEE intelligent systems*, 18(6):40–47, 2003.

E. Davis. *Representations of commonsense knowledge*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1990.

N.R. Devanur, M. Mihail, and V.V. Vazirani. Strategyproof cost-sharing mechanisms for set cover and facility location games. *Decision Support Systems*, 2005.

F. Dignum and H. Weigand. Modelling communication between cooperative systems. In *Advanced Information Systems Engineering*, pages 140–153. Springer, 1995.

K. Dismukes, G. Young, R. Sumwalt, J. McElhatton, P. Buchanan, C. Drew, K. Etem, and M. Patten. Cockpit interruptions and distractions. *Air Line Pilot*, 68:18–21, 1999.

X. Fan, J. Yen, and R.A. Volz. A theoretical framework on proactive information exchange in agent teamwork. *Artificial Intelligence*, 169(1):23–97, 2005.

M. Fleming and R. Cohen. A user modeling approach to determining system initiative in mixed-initiative ai systems. *User Modeling 2001*, pages 54–63, 2001.

J. Fogarty, S.E. Hudson, C.G. Atkeson, D. Avrahami, J. Forlizzi, S. Kiesler, J.C. Lee, and J. Yang. Predicting human interruptibility with sensors. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 12(1):119–146, 2005.

Y. Freund and R.E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

J. Gluck, A. Bunt, and J. McGrenere. Matching attentional draw with utility in interruption. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 50. ACM, 2007.

P. Gmytrasiewicz and P. Doshi. Interactive pomdps: Properties and preliminary results. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1374–1375, 2004.

C.V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 137–144. ACM New York, NY, USA, 2003.

C.V. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research*, 22:143–174, 2004.

C.V. Goldman and S. Zilberstein. Communication-based decomposition mechanisms for decentralized MDPs. *Journal of Artificial Intelligence Research*, 32:169–202, 2008.

V.M. González and G. Mark. Constant, constant, multi-tasking craziness: Managing multiple working spheres. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 113–120. ACM, 2004.

B. Grosz. Collaborative systems. *AI Magazine*, 17(2):67–86, 1996.

B.J. Grosz and L. Hunsberger. The dynamics of intention in collaborative activity. *Cognitive Systems Research*, 7(2-3):259–272, 2006.

B.J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

B.J. Grosz and S. Kraus. The evolution of SharedPlans. *Foundations of Rational Agency*, 14:227–262, 1999.

B.J. Grosz, S. Kraus, S. Talman, B. Stossel, and M. Havlin. The influence of social dependencies on decision-making: Initial investigations with a new game. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 782–789. IEEE Computer Society Washington, DC, USA, 2004.

T. Groves. Incentives in Teams. *Econometrica*, 1973.

E. Horvitz and J. Apacible. Learning and reasoning about interruption. In *Proceedings of the 5th International Conference on Multimodal Interfaces*, pages 20–27. ACM New York, NY, USA, 2003.

E. Horvitz, A. Jacobs, and D. Hovel. Attention-sensitive alerting. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, volume 99, pages 305–313, 1999.

E. Horvitz, P. Koch, C.M. Kadie, and A. Jacobs. Coordinate: Probabilistic forecasting of presence and availability. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2002.

E. Horvitz, P. Koch, and J. Apacible. BusyBody: Creating and fielding personalized models of the cost of interruption. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, pages 507–510. ACM New York, NY, USA, 2004.

E. Horvitz, P. Koch, R. Sarin, J. Apacible, and M. Subramani. Bayesphone: Precomputation of context-sensitive policies for inquiry and action in mobile devices. *User Modeling*, pages 251–260, 2005.

E. Horvitz, P. Koch, and M. Subramani. Mobile opportunistic planning: Methods and models. *User Modeling*, pages 228–237, 2007.

S. Hudson, J. Fogarty, C. Atkeson, D. Avrahami, J. Forlizzi, S. Kiesler, J. Lee, and J. Yang. Predicting human interruptibility with sensors: A wizard of Oz feasibility study. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 257–264. ACM New York, NY, USA, 2003.

L. Hunsberger and B. Grosz. A combinatorial auction for collaborative planning. In *Proceedings of the Fourth International Conference on MultiAgent Systems*, pages 10–12, 2000.

S.T. Iqbal and E. Horvitz. Disruption and recovery of computing tasks: field study, analysis, and directions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, page 686. ACM, 2007.

S.T. Iqbal and E. Horvitz. Notifications and awareness: A field study of alert usage and preferences. In *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*, pages 27–30. ACM, 2010.

N.R. Jennings. On being responsible. *Decentralized Artificial Intelligence*, pages 93–102, 1992.

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

L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

E. Kamar and B.J. Grosz. Applying MDP approaches for estimating outcome of interaction in collaborative human-computer settings. *Multi-Agent Sequential Decision Making in Uncertain Domains*, pages 25–32, 2007.

E. Kamar, E. Horvitz, and C. Meek. Mobile opportunistic commerce: mechanisms, architecture, and application. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 1087–1094. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

H. Kautz. A circumscriptive theory of plan recognition. *Intentions in Communication*, 134, 1990.

K. Keogh, L. Sonenberg, and W. Smith. Coordination in adaptive organisations: Extending shared plans with knowledge cultivation. In *Organized Adaption in Multi-Agent Systems: First International Workshop, OAMAS 2008, Estoril, Portugal, May 13, 2008. Revised and Invited Papers*, page 90. Springer, 2009.

R. Khardon and G. Wachman. Noise tolerant variants of the perceptron algorithm. *The Journal of Machine Learning Research*, 8:227–248, 2007.

D. Kinny, M. Ljungberg, A.S. Rao, E. Sonenberg, G. Tidhar, and E. Werner. Planned team activity. *Artificial Social Systems*, 1992.

R.E. Kraut and P. Attewell. Media use in a global corporation: Electronic mail and organizational knowledge. *Culture of the Internet*, pages 323–342, 1997.

W. Krauth and M. Mezard. Learning algorithms with optimal stability in neural networks. *Journal of Physics A: Mathematical and General*, 20:L745–L752, 1987.

J. Krumm and E. Horvitz. The Microsoft multiperson location survey. Technical report, MSR-TR-2005-103, Microsoft Research, 2005.

J. Krumm and E. Horvitz. Predestination: Inferring destinations from partial trajectories. *Ubiquitous Computing*, pages 243–260, 2006.

S. LeeTiernan, E. Cutrell, M. Czerwinski, and H. Hoffman. Effective notification systems depend on user trust. In *Proceedings of Human-Computer Interaction–Interact*. Citeseer, 2001.

H.J. Levesque, P.R. Cohen, and J.H.T. Nunes. On acting together. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 94–99. Boston, MA, 1990.

X.Y. Li, Z. Sun, W. Wang, and W. Lou. Cost sharing and strategyproof mechanisms for set cover games. *Journal of Combinatorial Optimization*, page 218, 2005.

Y. Li and P.M. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 46(1):361–387, 2002.

D.J. Litman and J.F. Allen. Discourse processing and commonsense plans. *Intentions in Communication*, pages 365–388, 1990.

K.E. Lochbaum. Using collaborative plans to model the intentional structure of discourse. *Computational Linguistics*, 24:525–572, 1994.

D. McFarlane. Comparison of four primary methods for coordinating the interruption of people in human-computer interaction. *Human-Computer Interaction*, 17(1):63–139, 2002.

R.B. Myerson and M.A. Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 29(2), 1981.

N. Nisan and A. Ronen. Computationally feasible VCG mechanisms. *Journal of Artificial Intelligence Research*, 29(1):19–47, 2007.

B. O'Conaill and D. Frohlich. Timespace in the workplace: Dealing with interruptions. In *Conference on Human Factors in Computing Systems*, pages 262–263. ACM New York, NY, USA, 1995.

F.A. Oliehoek, M.T.J. Spaan, S. Whiteson, and N. Vlassis. Exploiting locality of interaction in factored Dec-POMDPs. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 517–524. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, 2008.

D.C. Parkes, J. Kalagnanam, and M. Eso. Achieving budget-balance with Vickrey-based payment schemes in combinatorial exchanges. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1161–1168, 2001.

L. Perlow. The time famine: Toward a sociology of work time. *Administrative Science Quarterly*, 44(1):57–81, 1999.

M.E. Pollack. Plans as complex mental attitudes. *Intentions in Communication*, pages 77–103, 1990.

D.V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

Z. Rabinovich, C.V. Goldman, and J.S. Rosenschein. The complexity of multiagent systems: The price of silence. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1102–1103. ACM New York, NY, USA, 2003.

C. Rich and C.L. Sidner. COLLAGEN: When agents collaborate with people. In *Proceedings of the First International Conference on Autonomous Agents*, pages 284–291. ACM New York, NY, USA, 1997.

J.K. Rilling, A.G. Sanfey, J.A. Aronson, L.E. Nystrom, and J.D. Cohen. The neural correlates of theory of mind within interpersonal interactions. *Neuroimage*, 22(4):1694–1703, 2004.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

M. Roth, R. Simmons, and M. Veloso. Reasoning about joint beliefs for execution-time communication decisions. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 786–793. ACM New York, NY, USA, 2005.

M. Roth, R. Simmons, and M. Veloso. What to communicate? Execution-time decision in multi-agent POMDPs. In *The 8th International Symposium on Distributed Autonomous Robotic Systems*. Springer, 2006.

M. Roth, R. Simmons, and M. Veloso. Exploiting factored representations for decentralized execution in multiagent teams. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*. ACM New York, NY, USA, 2007.

D.M. Roy and L.P. Kaelbling. Efficient Bayesian task-level transfer learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, Hyderabad, India*, 2007.

P. Rudman and M. Zajicek. Autonomous agent as helper-helpful or annoying? In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 170–176. IEEE Computer Society, 2006.

D. Sarne and B.J. Grosz. Estimating information value in collaborative multi-agent planning systems. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, page 48. ACM, 2007.

P. Scerri, D. Pynadath, L. Johnson, P. Rosenbloom, M. Si, N. Schurr, and M. Tambe. A prototype infrastructure for distributed robot-agent-person teams. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 433–440. ACM New York, NY, USA, 2003.

S. Seuken, R. Cavallo, and D.C. Parkes. Partially-synchronized DEC-MDPs in dynamic mechanism design. In *Proceedings of the 23rd National Conference on Artificial Intelligence-Volume 1*, pages 162–169. AAAI Press, 2008.

J. Shen, R. Becker, and V. Lesser. Agent interaction in distributed POMDPs and its implications on complexity. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 529–536. ACM New York, NY, USA, 2006.

H.A. Simon. The architecture of complexity. *Proceedings of the American Philosophical Society*, pages 467–482, 1962.

C. Speier, J.S. Valacich, and I. Vessey. The influence of task interruption on individual decision making: An information overload perspective. *Decision Sciences*, 30:337–360, 1999.

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

H.R. Varian. Economic mechanism design for computerized agents. In *USENIX workshop on Electronic Commerce*, pages 13–21, 1995.

W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 1961.

S. Voida and E.D. Mynatt. It feels better than filing: Everyday work experiences in an activity-based computing system. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, pages 259–268. ACM, 2009.

P. Xuan, V. Lesser, and S. Zilberstein. Communication decisions in multi-agent cooperation: Model and experiments. In *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 616–623. ACM New York, NY, USA, 2001.

Appendix A

Proofs of Theorems in Chapter 4

Theorem A.0.1. *The joint value of the joint policy π for an ND-MDP as defined in Section 4.1.2 is the aggregate of the individual values of agents 1 and 2 for π:*

$$V^\pi(s_1^h, s_2^h) = V_1^\pi(s_1^h, s_2^h) + V_2^\pi(s_1^h, s_2^h)$$

where the value for agent 1 of choosing π in state $s^h = (s_1^h, s_2^h)$ is

$$V_1^\pi(s_1^h, s_2^h) = R_1(s_1^h) + \sum_{(s_1^{h+1}, s_2^{h+1})} T\big((s_1^{h+1}, s_2^{h+1}) \mid (s_1^h, s_2^h), \pi(s_1^h, s_2^h)\big)$$

$$\cdot V_1^\pi(s_1^{h+1}, s_2^{h+1})$$

Proof. By induction:

Basis: $h = H$, the end of the time horizon is reached. Starting with the original Dec-MDP value function given in Equation 4.1:

$$V^{\pi}(s_1^H, s_2^H) = R(s_1^H)$$

Applying reward independence property from Equation 4.3

$$= R_1(s_1^H) + R_2(s_2^H)$$

$$= V_1^{\pi}(s_1^H, s_2^H) + V_2^{\pi}(s_1^H, s_2^H)$$

Inductive step: For each $h + 1 < H$, assuming that:

$$V^{\pi}(s_1^{h+1}, s_2^{h+1}) = V_1^{\pi}(s_1^{h+1}, s_2^{h+1}) + V_2^{\pi}(s_1^{h+1}, s_2^{h+1}) \tag{A.1}$$

Building value function for time step h by applying equation 4.1 from Section 4.1.1 on Equation A.1:

$$V^{\pi}(s_1^h, s_2^h) = R(s_1^h, s_2^h) + \sum_{(s_1^{h+1}, s_2^{h+1})} T\big((s_1^{h+1}, s_2^{h+1}) \mid (s_1^h, s_2^h), \pi(s_1^h, s_2^h)\big) \cdot V^{\pi}(s_1^{h+1}, s_2^{h+1})$$

Applying semi-transition and reward independence properties of the ND-MDP formalization given in Equations 4.3 and 4.2

$$= R_1(s_1^h) + R_2(s_2^h) + \sum_{(s_1^{h+1}, s_2^{h+1})} T_1\big(s_1^{h+1} \mid s_1^h, \pi(s_1^h, s_2^h)\big)$$

$$\cdot T_2\big(s_2^{h+1} \mid s_2^h, \pi(s_1^h, s_2^h)\big) \cdot (V_1^{\pi}(s_1^{h+1}, s_2^{h+1}) + V_2^{\pi}(s_1^{h+1}, s_2^{h+1}))$$

$$= R_1(s_1^h) + \sum_{(s_1^{h+1}, s_2^{h+1})} T_1\big(s_1^{h+1} \mid s_1^h, \pi(s_1^h, s_2^h)\big) \cdot T_2\big(s_2^{h+1} \mid s_2^h, \pi(s_1^h, s_2^h)\big)$$

$$\cdot V_1^{\pi}(s_1^{h+1}, s_2^{h+1})$$

$$+ R_2(s_2^h) + \sum_{(s_1^{h+1}, s_2^{h+1})} T_1\big(s_1^{h+1} \mid s_1^h, \pi(s_1^h, s_2^h)\big) \cdot T_2\big(s_2^{h+1} \mid s_2^h, \pi(s_1^h, s_2^h)\big)$$

$$\cdot V_2^{\pi}(s_1^{h+1}, s_2^{h+1})$$

$$= V_1^{\pi}(s_1^h, s_2^h) + V_2^{\pi}(s_1^h, s_2^h) \tag{A.2}$$

□

Theorem A.0.2. *The policy π_c^* maximizing the ND-DECOP algorithm value function $V^{\pi_c^*}$ as given in Equations A.3 and A.4 is an optimal policy for a given ND-MDP.*

$$V^{\pi_c^*}(s_c^h) = \begin{cases} \max_{a \in A_c} \big[R_c(s_c^h) + \Sigma_{s_c^{h+1}} T_c(s_c^{h+1}|s_c^h, a) & \\ \cdot V^{\pi_c^*}(s_c^{h+1})\big] & \textit{if } h{<}H \\ R_c(s_c^h) + V^{\pi_q^*}(s_1^o \cup C^{0,H-1}) & \textit{if } s_c^h = s_2^H \cup C^{0,H-1} \cup s_1^o \end{cases} \tag{A.3}$$

$$V^{\pi_q^*}(s_q^h) = \max_{a_q : t(a_q) = t^h} \big[R_q(s_q^h) + \Sigma_{s_q^{h+1}} T_q(s_q^{h+1}|s_q^h, a_q) \cdot V^{\pi_q^*}(s_q^{h+1})\big] \tag{A.4}$$

Proof. The value function of an optimal policy for a Dec-MDP can be converted to the ND-DECOP value functions given above, thus policies computed by the ND-DECOP value functions are optimal for a given ND-MDP.

The action sets of agent 1 and 2, A_1 and A_2, are mapped to A_q and A_c as $A_q = A_1^I \cup A^J$, $A_c = A_2^I \cup A^J$. For a whole and succinct representation, a constraint function (C) is defined over A_q and A_c. The constraint function applied in sequence mimics the functionality of the type sequence.

Definition 1.

$$C(a_c^h) = \begin{cases} \{a_c^h\} & \text{if } a_c^h \in A^J \\ A_1^I & \text{otherwise} \end{cases}$$

Using the Constraint function, the ND-DECOP value functions given in Equations A.3 and A.4 can be rewritten as below:

$$
\begin{aligned}
V^{\pi_c^*}(s_c^0 = s_1^0 \cup s_2^0) = &\max_{a_c^0 \in A_c} \big[R_2(s_2^0) + \Sigma_{s_2^1} T_2(s_2^1 | s_2^0, a_c^0) \\
&\cdot \max_{a_c^1 \in A_c} \big[R_2(s_2^1) + \Sigma_{s_2^2} T_2(s_2^2 | s_2^1, a_c^1) \cdots \\
&\cdot \max_{a_c^{H-1} \in A_c} \big[R_2(s_2^{H-1}) + \Sigma_{s_2^H} T_2(s_2^H | s_2^{H-1}, a_c^{H-1}) \cdot R_2(s_2^H) \\
&+ \max_{a_q^0 \in C(a_c^0)} \big[R_1(s_1^0) + \Sigma_{s_1^1} T_1(s_1^1 | s_1^0, a_q^0) \cdots \\
&\cdot \max_{a_q^{H-1} \in C(a_c^{H-1})} \big[R_1(s_1^{H-1}) + \Sigma_{s_1^H} T_1(s_1^H | s_1^{H-1}, a_q^{H-1}) \cdot R_1(s_1^H)\big]\big]\big]\big]\big]
\end{aligned}
\tag{A.5}
$$

Proof by induction:

Basis: $h = H - 1$. Starting with the original Dec-MDP value function given in Equation 4.1:

$$
\begin{aligned}
V^{\pi^*}(s_1^{H-1}, s_2^{H-1}) = &\max_{(a_1^{H-1} \in A_1, a_2^{H-1} \in A_2)} \big[R(s_1^{H-1}, s_2^{H-1}) \\
&+ \sum_{(s_1^H, s_2^H)} T\big((s_1^H, s_2^H) \mid (s_1^{H-1}, s_2^{H-1}), (a_1^{H-1}, a_2^{H-1})\big) \cdot R(s_1^H, s_2^H)\big]
\end{aligned}
$$

Applying reward independence property from Equation 4.3

$$
\begin{aligned}
&= \max_{(a_1^{H-1}, a_2^{H-1})} \big[R_1(s_1^{H-1}) + R_2(s_2^{H-1}) \\
&+ \sum_{(s_1^H, s_2^H)} T\big((s_1^H, s_2^H) \mid (s_1^{H-1}, s_2^{H-1}), (a_1^{H-1}, a_2^{H-1})\big) \\
&\cdot \big(R_1(s_1^H) + R_2(s_2^H)\big)\big]
\end{aligned}
$$

Mapping action sets A_1 and A_2 to A_q and A_c and applying semi-transition property given in Equation 4.2

$$V^{\pi^*}(s_1^{H-1}, s_2^{H-1}) = \max_{(a_q^{H-1} \in A_q, a_c^{H-1} \in A_c)} \big[R_1(s_1^{H-1}) + R_2(s_2^{H-1}) + \sum_{(s_1^H, s_2^H)} T_1\big(s_1^H \mid s_1^{H-1}, a_q^{H-1}\big)$$

$$\cdot T_2\big(s_2^H \mid s_2^{H-1}, a_c^{H-1}\big) \cdot \big(R_1(s_1^H) + R_2(s_2^H)\big)\big]$$

Decoupling max operator using the Constraint function (mimicking the type sequence)

$$= \max_{a_c^{H-1}} \big[\max_{a_q^{H-1} \in C(a_c^{H-1})} \big[R_1(s_1^{H-1}) + R_2(s_2^{H-1}) + \sum_{(s_1^H, s_2^H)} T_1\big(s_1^H \mid s_1^{H-1}, a_q^{H-1}\big)$$

$$\cdot T_2\big(s_2^H \mid s_2^{H-1}, a_c^{H-1})\big) \cdot \big(R_1(s_1^H) + R_2(s_2^H)\big)\big]\big]$$

Applying distributivity principle and eliminating independent variables that sum up to 1.0 (e.g., Function R_1 is independent of function T_2 and variables s_2^{H-1} and s_2^H)

$$= \max_{a_c^{H-1}} \big[\max_{a_q^{H-1} \in C(a_c^{H-1})} \big[R_1(s_1^{H-1}) + R_2(s_2^{H-1}) + \sum_{s_1^H} T_1\big(s_1^H \mid s_1^{H-1}, a_q^{H-1}\big)$$

$$\cdot R_1(s_1^H) + \sum_{s_2^H} T_2\big(s_2^H \mid s_2^{H-1}, a_c^{H-1}\big) \cdot R_2(s_2^H)\big]\big]$$

Moving max operators according to the variables they are dependent to

$$= \max_{a_c^{H-1}} \big[R_2(s_2^{H-1}) + \sum_{s_2^H} T_2\big(s_2^H \mid s_2^{H-1}, a_c^{H-1}\big) \cdot R_2(s_2^H)$$

$$+ \max_{a_q^{H-1} \in C(a_c^{H-1})} \big[R_1(s_1^{H-1}) + \sum_{s_1^H} T_1\big(s_1^H \mid s_1^{H-1}, a_q^{H-1}\big) \cdot R_1(s_1^H)\big]\big]$$

$$= V^{\pi_c^*}(s_c^{H-1}), \text{ where } s_c^{H-1} = s_1^{H-1} \cup s_2^{H-1} \tag{A.6}$$

Inductive step: For each $h+1 < H$, assuming that

$$V^{\pi^*}(s_1^{h+1}, s_2^{h+1}) = V^{\pi_c^*}(s_c^{h+1}), \textit{ where } s_c^{h+1} = s_1^{h+1} \cup s_2^{h+1} \tag{A.7}$$

Building value function for time step h by applying the original Dec-MDP value function as given in Equation 4.1

$$V^{\pi^*}(s_1^h, s_2^h) = \max_{(a_1^h \in A_1, a_2^h \in A_2)} \big[R(s_1^h, s_2^h) + \sum_{(s_1^{h+1}, s_2^{h+1})} T\big((s_1^{h+1}, s_2^{h+1}) \mid (s_1^h, s_2^h), (a_1^h, a_2^h)\big)$$

$$\cdot V^{\pi^*}(s_1^{h+1}, s_2^{h+1})\big]$$

Applying the induction hypothesis from Equation A.7

$$= \max_{(a_1^h, a_2^h)} \big[R(s_1^h, s_2^h) + \sum_{(s_1^{h+1}, s_2^{h+1})} T\big((s_1^{h+1}, s_2^{h+1}) \mid (s_1^h, s_2^h), (a_1^h, a_2^h)\big)$$

$$\cdot V^{\pi_c^*}(s_c^{h+1})\big]$$

Applying reward independence property from Equation 4.3

$$= \max_{(a_1^h, a_2^h)} \big[R_1(s_1^h) + R_2(s_2^h) + \sum_{(s_1^{h+1}, s_2^{h+1})} T\big((s_1^{h+1}, s_2^{h+1}) \mid (s_1^h, s_2^h), (a_1^h, a_2^h)\big)$$

$$\cdot V^{\pi_c^*}(s_c^{h+1})\big]$$

Mapping action sets A_1 and A_2 to A_q and A_c and applying semi-transition property given in Equation 4.2

$$= \max_{(a_q^h \in A_q, a_c^h \in A_c)} \big[R_1(s_1^h) + R_2(s_2^h) + \sum_{(s_1^{h+1}, s_2^{h+1})} T_1\big(s_1^{h+1} \mid s_1^h, a_q^h\big)$$

$$\cdot T_2\big(s_2^{h+1} \mid s_2^h, a_c^h\big) \cdot V^{\pi_c^*}(s_c^{h+1})\big]$$

Applying Equation A.5 in return for $V^{\pi_c^*}(s_c^{h+1})$

$$V^{\pi^*}(s_1^h, s_2^h) = \max_{(a_q^h \in A_q, a_c^h \in A_c)} \big[R_1(s_1^h) + R_2(s_2^h) + \sum_{(s_1^{h+1}, s_2^{h+1})} T_1\big(s_1^{h+1} \mid s_1^h, a_q^h\big) \cdot$$

$$T_2\big(s_2^{h+1} \mid s_2^h, a_c^h\big)$$

$$\cdot \max_{a_c^{h+1} \in A_c} \big[R_2(s_2^{h+1}) + \Sigma_{s_2^{h+2}} T_2(s_2^{h+2} | s_2^{h+1}, a_c^{h+1}) \cdots$$

$$\cdot \max_{a_c^{H-1} \in A_c} \big[R_2(s_2^{H-1}) + \Sigma_{s_2^H} T_2(s_2^H | s_2^{H-1}, a_c^{H-1}) \cdot R_2(s_2^H)$$

$$+ \max_{a_q^{h+1} \in C(a_c^{h+1})} \big[R_1(s_1^{h+1}) + \Sigma_{s_1^{h+2}} T_1(s_1^{h+2} | s_1^{h+1}, a_q^{h+1}) \cdots$$

$$\cdot \max_{a_q^{H-1} \in C(a_c^{H-1})} \big[R_1(s_1^{H-1}) + \Sigma_{s_1^H} T_1(s_1^H | s_1^{H-1}, a_q^{H-1}) \cdot R_1(s_1^H)\big]\big]\big]\big]\big]$$

Decoupling max operator using the Constraint function

$$= \max_{a_c^h \in A_c} \big[\max_{a_q^h \in C(a_c^h)} \big[R_1(s_1^h) + R_2(s_2^h) + \sum_{(s_1^{h+1}, s_2^{h+1})} T_1\big(s_1^{h+1} \mid s_1^h, a_q^h\big)$$

$$\cdot T_2\big(s_2^{h+1} \mid s_2^h, a_c^h\big)$$

$$\cdot \max_{a_c^{h+1} \in A_c} \big[R_2(s_2^{h+1}) + \Sigma_{s_2^{h+2}} T_2(s_2^{h+2} | s_2^{h+1}, a_c^{h+1}) \cdots$$

$$\cdot \max_{a_c^{H-1} \in A_c} \big[R_2(s_2^{H-1}) + \Sigma_{s_2^H} T_2(s_2^H | s_2^{H-1}, a_c^{H-1}) \cdot R_2(s_2^H)$$

$$+ \max_{a_q^{h+1} \in C(a_c^{h+1})} \big[R_1(s_1^{h+1}) + \Sigma_{s_1^{h+2}} T_1(s_1^{h+2} | s_1^{h+1}, a_q^{h+1}) \cdots$$

$$\cdot \max_{a_q^{H-1} \in C(a_c^{H-1})} \big[R_1(s_1^{H-1}) + \Sigma_{s_1^H} T_1(s_1^H | s_1^{H-1}, a_q^{H-1}) \cdot R_1(s_1^H)\big]\big]\big]\big]\big]$$

Applying distributivity principle and eliminating independent variables

$$= \max_{a_c^h \in A_c} \big[\max_{a_q^h \in C(a_c^h)} \big[R_1(s_1^h) + R_2(s_2^h) + \sum_{s_2^{h+1}} T_2\big(s_2^{h+1} \mid s_2^h, a_c^h\big)$$

$$\cdot \max_{a_c^{h+1} \in A_c} \big[R_2(s_2^{h+1}) + \Sigma_{s_2^{h+2}} T_2(s_2^{h+2} | s_2^{h+1}, a_c^{h+1}) \cdots$$

$$
\begin{aligned}
&\cdot \max_{a_c^{H-1}\in A_c}\big[R_2(s_2^{H-1}) + \Sigma_{s_2^H} T_2(s_2^H|s_2^{H-1}, a_c^{H-1}) \cdot R_2(s_2^H)\\
&+ \sum_{s_1^{h+1}} T_1\big(s_1^{h+1} \mid s_1^h, a_q^h\big)\\
&\cdot \max_{a_q^{h+1}\in C(a_c^{h+1})}\big[R_1(s_1^{h+1}) + \Sigma_{s_1^{h+2}} T_1(s_1^{h+2}|s_1^{h+1}, a_q^{h+1}) \cdots\\
&\cdot \max_{a_q^{H-1}\in C(a_c^{H-1})}\big[R_1(s_1^{H-1}) + \Sigma_{s_1^H} T_1(s_1^H|s_1^{H-1}, a_q^{H-1}) \cdot R_1(s_1^H)\big]\big]\big]\big]\big]
\end{aligned}
$$

Moving max operators according to the variables they are dependent to

$$
\begin{aligned}
V^{\pi^*}(s_1^h, s_2^h) = &\max_{a_c^h\in A_c}\big[R_2(s_2^h) + \sum_{s_2^{h+1}} T_2\big(s_2^{h+1} \mid s_2^h, a_c^h\big)\\
&\cdot \max_{a_c^{h+1}\in A_c}\big[R_2(s_2^{h+1}) + \Sigma_{s_2^{h+2}} T_2(s_2^{h+2}|s_2^{h+1}, a_c^{h+1}) \cdots\\
&\cdot \max_{a_c^{H-1}\in A_c}\big[R_2(s_2^{H-1}) + \Sigma_{s_2^H} T_2(s_2^H|s_2^{H-1}, a_c^{H-1}) \cdot R_2(s_2^H)\\
&+ \max_{a_q^h\in C(a_c^h)}\big[R_1(s_1^h) + \sum_{s_1^{h+1}} T_1\big(s_1^{h+1} \mid s_1^h, a_q^h\big)\\
&\cdot \max_{a_q^{h+1}\in C(a_c^{h+1})}\big[R_1(s_1^{h+1}) + \Sigma_{s_1^{h+2}} T_1(s_1^{h+2}|s_1^{h+1}, a_q^{h+1}) \cdots\\
&\cdot \max_{a_q^{H-1}\in C(a_c^{H-1})}\big[R_1(s_1^{H-1}) + \Sigma_{s_1^H} T_1(s_1^H|s_1^{H-1}, a_q^{H-1}) \cdot R_1(s_1^H)\big]\big]\big]\big]\big]
\end{aligned}
$$

$$V^{\pi^*}(s_1^h, s_2^h) = V^{\pi_c^*}(s_c^h), \textit{ where } s_c^h = s_1^h \cup s_2^h. \tag{A.8}$$

□